A Case Against Routing-Integrated Time Synchronization

Thomas Schmid[†‡], Zainul Charbiwala[†], Zafeiria Anagnostopoulou[†], Mani B. Srivastava[†], Prabal Dutta[‡]

[†]Electrical Engineering Department
University of California, Los Angeles
Los Angeles, CA 90095
{zainul, zafeiria, mbs}@ucla.edu

[‡]Computer Science & Engineering Division
University of Michigan
Ann Arbor, MI 48109
{thschmid, prabal}@eecs.umich.edu

Abstract

To achieve more accurate global time synchronization, this paper argues for decoupling the clock distribution network from the routing tree in a multihop wireless network. We find that both flooding and routing-integrated time synchronization rapidly propagate node-level errors (typically due to temperature fluctuations) across the network. Therefore, we propose that a node chooses synchronization neighbors that offer the greatest frequency stability. We propose two methods to estimate a neighbor's stability. The first approach selects the neighbor whose Frequency Error Variance, or simply FEV, is smallest with respect to the local clock. The second approach selects the neighbor that reports the lowest FEV relative to its synchronization parent. We also propose the node-level time-variance FEV as an additive metric for selecting more stable clock trees than either naïve flooding or routing-integrated time synchronization can provide. We incorporate these techniques into FTSP, a widely-used time synchronization protocol, and show that the mean error in global time significantly improved (by a factor of five) when some nodes are warmed and others are not.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design, Network Protocols

General Terms

Algorithm, Design, Performance, Reliability, Experimentation, Measurement

Keywords

Sensor Networks, Time Synchronization, Clock Synchronization, Clock Drift, Multi-hop, Routing Integrated

SenSys'10, November 3–5, 2010, Zurich, Switzerland.

1 Introduction

Time synchronization is one of the most fundamental and widely used middleware services in distributed wireless sensor networks. Accurate and stable time estimates are essential for correlating distributed observations [3], decreasing communications energy [5], improving localization accuracy [12], increasing security [21], and improving coordination [2]. Modern sensornet time sync protocols, like the Flooding Time Synchronization Protocol (FTSP) [16] or its many optimizations [11, 13, 22, 23], are able to quickly establish network time, accurately estimate clock skews, steadily maintain global virtual time, and efficiently integrate with routing. Today, microsecond-level time synchronization can be readily achieved and steadily maintained under the kind of stable environmental conditions that one might find in the lab or in a temperature-controlled indoor setting.

Unfortunately for time synchronization, many modern sensor networks are being deployed in forests [24], data centers [14], and mixed indoor/outdoor settings [19] where the temperature field is neither uniform across the nodes nor fixed in time. When the sun shines at daybreak in a forest, the nodes at the top of a tree heat up faster than those in the middle or bottom. When a rack-mounted server's utilization increases suddenly, its exhaust temperature profile changes quickly. And when a network consisting of nodes located both indoors and out experiences a partly cloudy day, different nodes experience vastly different temperature profiles. These temperature differences lead to frequency errors which are then rapidly propagated across the network by routing-integrated or flooding-based time synchronization protocols, resulting in time synchronization errors.

To achieve more accurate global time synchronization, we argue for a logical decoupling of time synchronization from naïve flooding. In other words, we propose to decouple the clock distribution network from the routing tree in multihop wireless networks. Today's time synchronization protocols liberally integrate timing information offered by neighboring nodes into their own estimates. We argue that instead of blindly integrating such information, especially when offered by neighbors with dubious frequency stability on an ad hoc basis, that a node chooses the synchronization neighbor *that offers the most stable path from the root*. The challenge, then, becomes how nodes estimate and propagate path stability without access to a stable clock source – a seemingly circular problem.

In this paper, we propose two methods to estimate a neighbor's frequency stability and, by induction, path stability. Our key insight is that a node's time estimation errors, when expressed as a variance, provides an additive path metric by which time sync protocols can choose synchronization neighbors. Similar to additive metrics like hopcount in routing, and expected number of transmissions (ETX) in wireless, we propose a new metric called Frequency Error Variance or simply FEV. The FEV metric is additive, and we describe how it can be estimated locally by a node, or cooperatively by neighbors. Using FEV, selecting a synchronization parent is straightforward – a node simply selects the neighbor that offers the lowest FEV.

The Autonomous Time Information Routing Protocol (A-TIRP) selects the neighbor whose frequency error variance is smallest, with respect to the local clock. In this approach, the FEV is computed locally, and the additive property of frequency variance is expressed by the neighbors announcement of their global time estimates. This approach can be introduced incrementally and does not require changes to current protocols. However, it can result in deviant cliques of nodes forming timing loops (which are similar to routing loops) when nodes have frequency errors that are not independent or real-world quantization errors are considered.

The Cooperative Time Information Routing Protocol (C-TIRP) selects the neighbor that reports the lowest frequency error variance. When the parent and child's time estimates diverge, then either the parent or the child is experiencing clock instability. But even if these two values track, the nodes may still be experiencing correlated frequency errors. To detect this situation, we also require that the root of the network employ a stable clock and include a hopcount in its TIRP beacons. The hopcount is then used to break the timing loops that we discussed earlier.

This paper makes several contributions. First, we show how three basic sources of error – noise, quantization error, and frequency instability – conspire to affect network time synchronization. Second, we formulate the problem of *time information routing* as an optimization problem: the selection of a master clock by a slave in the presence of multiple clocks advertising themselves as being stable. We show that the variance of frequency skew is an additive property and we leverage this fact to propose the Frequency Error Variance (FEV) metric for routing time. Minimizing the FEV becomes the node-level objective function. We implement A-TIRP and C-TIRP using TinyOS and we compare the performance of the default FTSP implementation with and without our protocol. Our results show that by using FEV as the metric for clock tree construction, the accuracy of time synchronization is greatly improved, time error propagation is greatly attenuated and compartmentalized, and transients due to sudden temperature changes are quickly detected and corrected. Our design mitigates one of the major remaining sources of error in multihop wireless time synchronization systems, paving the way to better correlated distributed observations, reduced communication energy, improved localization accuracy, increasing security, and more tightly coordinated actions.

2 Estimating Clock Stability

The objective of a time synchronization protocol is maintaining clock accuracy at each node in the network with respect to a reference clock. It does this by nullifying the time offset and the frequency error of the local clock with respect to a global clock. In order to achieve this in a multi-hop wireless network, the protocol exchanges a sequence of synchronization messages between nodes. The synchronization messages originate at a master node that advertises its clock as being stable and is received by slave nodes that would like to stabilize their own clocks.

These messages are typically time-stamped transmissions that enable the synchronization algorithm on a slave node to estimate two quantities. One, by observing the difference in the transmission time reported in the message and the reception time measured locally, the slave node is able to compute the offset between clocks on the two nodes. Two, by observing the trend of offset measurements over a time window, the synchronization algorithm on the slave node can estimate the skew between the frequencies of the clocks. Applying these estimates to the local clock to cancel offset and skew, the slave node's clock accuracy can theoretically be equivalent to the master's. By performing this operation in succession from the root node to the rest of the network hop-by-hop, all nodes in the network can have a clock accuracy equivalent to the root node.

Three sources of error creep into this perfect system. First, timestamps are noisy because of jitter in processing delay during transmission at the master and reception at the slave. Second, since we are dealing with digital clocks, timestamp accuracy is limited by quantization error in time measurements. Quantization error is especially pronounced when the frequency of clocks is low due to a larger time ambiguity between consecutive ticks of the clock. Third, frequency error between master and slave clocks may change during the time window needed to collect sufficient offset measurements. An assumption made in practical synchronization algorithms is that the frequency error remains constant during the measurement period and thus the offsets can be fit to a linear function, the slope of which is the frequency error. When clock frequency changes, either at the master or the slave, it contributes to error in the offset measurement, which manifests as an error in the frequency error estimate. This error in frequency error estimation eventually leads to inaccuracy in clock synchronization.

The issue we would like to address in this paper is the selection of a master clock by a slave in the presence of multiple clocks advertising themselves as being stable. We term this problem time information routing to form an analogy with network routing, which addresses a similar issue - that of selecting the next hop to forward a packet toward its final destination based on wireless link estimations. In time information routing, the "next hop" for a slave node is the master clock it synchronizes to and the final destination is always the root clock in the system. In this way, time information routing is considered a tree routing problem: construct a time information tree rooted at the most stable clock in the system such that each clock in the network attains the highest accuracy possible.

As in network routing, the time information routing protocol must find the optimal tree with reasonable overhead cost, in a distributed manner, and be able to rapidly converge to a new tree if the topology of the network changes or the quality of clocks differs. A myriad of network routing protocols have been developed over the past three decades that each make distinct assumptions and trade offs in their design to realize these goals. A cornerstone of all routing protocols is the estimation of a cost metric, which quantifies the cost of a particular path to a destination. When this metric is monotonic along the path, there exist algorithms that compute the optimal path in polynomial time. For network routing, these gradient cost metrics are typically link delay, packet reception ratio or ETX [4]. For time information routing, we propose the use of Frequency Error Variance FEV as a cost gradient metric. The following section describes FEV and shows why it is an appropriate time information routing metric.

2.1 Frequency Error Variance as Routing Metric

In order to illustrate the concept behind FEV, we first analyze the effects of errors introduced in measurements using a simplified three node, two hop, linear network - $N_0 \rightarrow N_1 \rightarrow N_2$. Without loss of generality, we let node N_0 be the root node and assume that it has a perfect clock, i.e. its offset and frequency error with respect to a universal clock is always zero. From synchronization messages that N_0 broadcasts, N_1 can compute the offset c_1 between its local clock and N_0. For the i-th synchronization message, the offset can be modeled as:

$$c_1(i) = o_1(i) + e_1(i) \tag{1}$$

Here, $o_1(i)$ corresponds to the true offset between the clocks at the i-th message exchange and $e_1(i)$ is the error in the offset measurement introduced due to the three factors mentioned above. An additional factor could be included to account for the offset that builds up during the message exchange itself, but this error is short enough for wireless propagation times to be neglected in our analysis. Let the interval between two periodic synchronization messages be denoted as T. One could model the true offset at message i as:

$$o_1(i) = o_1(0) + \frac{iT\delta_1}{f_0} \tag{2}$$

where, $o_1(0)$ is the true initial offset between the clocks and δ_1 is the frequency error of N_1 with respect to N_0. f_0 is the nominal frequency of the N_0 master clock. The second term in the above relation is the drift in the offset between the clocks due to a difference in their frequencies. If f_1 is the frequency of N_1's clock, $\delta_1 = f_1 - f_0$.

It is the objective of the time synchronization algorithm to estimate offset $o_1(0)$ and frequency error δ_1 from a sequence of messages that provide noisy offset measurements, c_1. Since Equation (2) has an affine form, we consider the use of a least squares linear regression model to estimate both parameters. Using a window of w offset measurements, the estimates can be computed using [10]:

$$\hat{o}_1(0) = \bar{c_1} - b_1\bar{w} \tag{3}$$

$$\bar{c_1} = \frac{1}{w}\sum_{i=1}^{w} c_1(i) \tag{4}$$

$$\bar{w} = \frac{1}{w}\sum_{i=1}^{w} i = \frac{w+1}{2} \tag{5}$$

$$b_1 = \frac{\sum_{i=1}^{w}(c_1(i)-\bar{c_1})(i-\bar{w})}{\sum_{i=1}^{w}(i-\bar{w})^2} \tag{6}$$

$$\hat{\delta}_1 = \frac{f_0 b_1}{T} \tag{7}$$

Assuming that the offset measurement error $e_1(i)$ is independent across messages and is normally distributed $\sim \mathcal{N}(0,\sigma_1^2)$, the variance in the estimates $\hat{o}_1(0)$ and $\hat{\delta}_1$ is given by [10]:

$$\mathrm{Var}(\hat{o}_1) = \frac{1}{w}\left[1+\frac{\bar{w}^2}{\sigma_w^2}\right]\sigma_1^2 \tag{8}$$

$$\mathrm{Var}(\hat{\delta}_1) = \frac{f_0^2}{w\sigma_w^2 T^2}\sigma_1^2 \tag{9}$$

$$\sigma_w^2 = \frac{1}{w}\sum_{i=1}^{w}(i-\bar{w})^2 \tag{10}$$

The final relationships in Equations (8) and (9) map the variance at the input of the time synchronization algorithm in terms of offset measurement noise, σ_1^2, to a variance at the output in the offset and frequency error estimates.

After the reception of w messages, N_1 is considered synchronized with N_0 and the estimates from Equations (7) and (3) are applied to the local clock to cancel offset and frequency error. Since these estimates are not perfect (unless $\sigma_1^2 = 0$), the local clock at N_1 will not be an exact replica of the master clock at N_0. Nevertheless, once N_1 considers itself synchronized, it commences broadcasting messages too. N_2 hears these messages and repeats the process after receiving w of them to estimate $\hat{o}_2(0)$ and $\hat{\delta}_2$. However, as the clock at N_1 is not as accurate as the master clock, the noisy offset measurements at N_2 will now include additional terms:

$$c_2(i) = o_2(i) + e_2(i) + e_{o1} + \frac{iT}{f_0}e_{\delta 1} \tag{11}$$

$$e_{o1} = o_1(0) - \hat{o}_1(0) \tag{12}$$

$$e_{\delta 1} = \delta_1 - \hat{\delta}_1 \tag{13}$$

where, $o_2(i)$ is the true offset of the clock at N_2 with respect to the clock at N_0, $e_2(i)$ is the offset measurement error at N_2 and e_{o1} and $e_{\delta 1}$ are the errors introduced in offset measurement due to imperfect offset and frequency error estimation at N_1 respectively for message i.

One may perceive the error terms as being a combined error, the variance for which, after receiving w messages, is given by:

$$\sigma_2'^2 = \mathrm{Var}(e_2 + e_{o1} + \frac{wT}{f_0} e_{\delta 1}) \quad (14)$$

$$= \mathrm{Var}(e_2) + \mathrm{Var}(e_{o1}) + \frac{w^2T^2}{f_0^2}\mathrm{Var}(e_{\delta 1}) + \frac{2wT}{f_0}\mathrm{Cov}(e_{o1}, e_{\delta 1}) \quad (15)$$

$$= \mathrm{Var}(e_2) + \mathrm{Var}(\hat{o_1}) + \frac{w^2T^2}{f_0^2}\mathrm{Var}(\hat{\delta}_1) + \frac{2wT}{f_0}\mathrm{Cov}(\hat{o_1}, \hat{\delta}_1) \quad (16)$$

$$= \sigma_2^2 + \frac{\sigma_1^2}{w}\left[1 + \frac{\bar{w}^2}{\sigma_w^2}\right] + \frac{w^2T^2}{f_0^2} \cdot \frac{f_0^2\sigma_1^2}{w\sigma_w^2T^2} - \frac{2wT}{f_0} \cdot \frac{f_o\bar{w}\sigma_1^2}{Tw\sigma_w^2} \quad (17)$$

$$= \sigma_2^2 + \frac{\sigma_1^2}{w}\left[1 + \frac{\bar{w}^2}{\sigma_w^2} + \frac{w^2}{\sigma_w^2} - \frac{2w\bar{w}}{\sigma_w^2}\right] \quad (18)$$

$$= \sigma_2^2 + \sigma_1^2\left[\frac{1}{w} + \frac{(w-\bar{w})^2}{w\sigma_w^2}\right] \quad (19)$$

$$= \sigma_2^2 + \kappa\sigma_1^2 \quad (20)$$

Equation (15) is derived by expanding Equation (14) assuming independence between the error terms at N_1 and N_2. This assumption is reasonable because they are deemed physically independent as well. Equation (17) uses Equations (8) and (9) and results from [10] for the covariance term. Simplifying the resulting terms gives the final result in Equation (19). σ_2^2 is the inherent offset error variance that would be perceived at N_2 if it received synchronization messages directly from N_0. If one were to compute the variance of the frequency error estimate (FEV) at N_2 from the above, we would obtain:

$$\mathrm{Var}(\hat{\delta}_2) = \frac{f_0^2}{w\sigma_w^2T^2} \cdot (\sigma_2^2 + \kappa\sigma_1^2) \quad (21)$$

$$\kappa(w) = \frac{1}{w} + \frac{(w-\bar{w})^2}{w\sigma_w^2} \quad (22)$$

We glean a few observations from the above derivation. First, since all terms in Equation (22) are positive, $\kappa > 0$. This means that FEV is additive in nature along a time information path. Second, κ is the factor by which variance of offset measurements at the input of the master clock time synchronization algorithm influences FEV at the slave. Interestingly, κ is only dependent on the window size w of samples used. The variation of κ vs. w is plotted in Figure (1), which shows that as the window size is increased, this factor reduces. For FTSP [16], $w = 8$, which corresponds to $\kappa = 0.417$. This indicates that 41% of the offset error variance at a master clock is passed on to the slave clock. However, the length of w is bounded by the assumption that the frequency error has to be constant over the whole length of w, in effect limiting the maximum length of the synchronization window.

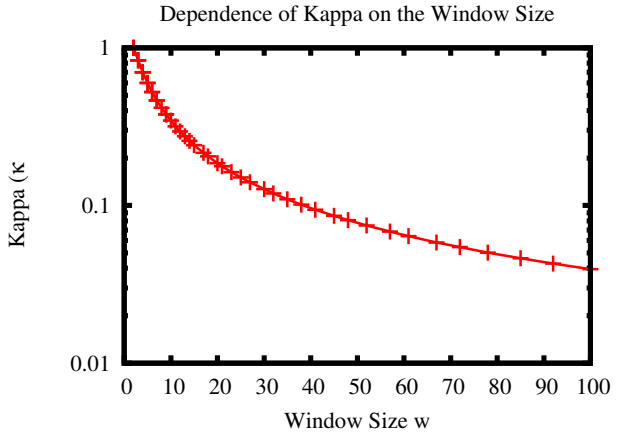


Figure 1. A plot of κ, the factor by which variance at the master clock is introduced in the slave.

Third, comparing Equations (9) and (21) for the case when $\sigma_1^2 = \sigma_2^2 = \sigma^2$, we see that FEV at N_2 is higher than that at N_1. Recall that σ^2 is a contribution of three factors: processing jitter, quantization error and frequency error variation. The case of $\sigma_1^2 = \sigma_2^2 = \sigma^2$ represents a steady state condition for a homogeneous network of nodes when there is no change in the environment and thus no change in frequency error. This implies that, in steady state, the variance of the frequency error estimate at a node that is two hops away from the root clock is higher than that at a node that can hear messages from the root clock directly. This is intuitive because a slave node that uses another node as the master clock must have a lower accuracy if they have equivalent offset measurement noise.

The above argument and analysis can be extended to a larger network to show that FEV is a monotonically increasing quantity along paths to a root in a time information tree. The FEV of a node k in the tree is given by the recurrence relation:

$$\mathrm{Var}(\hat{\delta}_k) = \frac{f_0^2\sigma_k^2}{w\sigma_w^2T^2} + \kappa \cdot \mathrm{Var}(\hat{\delta}_{\mathrm{master}(k)}) \quad (23)$$

Where, master(k) is the node a slave synchronizes to and $\mathrm{Var}(\hat{\delta}_{\mathrm{root}}) = 0$. We can now outline how this could be used practically to construct the time information tree itself.

Let there be another node, N_3, that is one hop away from the root and is synchronized to it. Let N_3 also be in communication range of N_2. Since N_2 can hear synchronization messages from both N_1 and N_3, it must decide which one to pick as master. If N_1 and N_3 could advertise their σ_1^2 and σ_3^2 respectively, N_2's obvious choice would be to pick the node with lower offset error variance. However, since σ_1^2 and σ_3^2 are unknown to N_1 and N_3 themselves, they can compute and advertise their FEV $\mathrm{Var}(\hat{\delta}_1)$ and $\mathrm{Var}(\hat{\delta}_3)$ respectively. For nodes one-hop away from the root, such as N_1 and N_3, $\mathrm{Var}(\hat{\delta})$ maps directly to σ^2 through Equation (9). Thus, by picking the node with lower $\mathrm{Var}(\hat{\delta})$, N_2 picks the one with lower σ^2.

If the topology of the network is unknown, however, N_2 does not know whether N_1/N_3 are in direct communication with the root. This would mean that Equation (9) may not apply, but rather Equation (23). The synchronization objective of attaining the highest clock stability is met when $\text{Var}(\hat{\delta}_2)$ is minimized. As all other terms except $\text{Var}(\hat{\delta}_{\text{master}(k)})$ for Equation (23) are constant at N_2, it can still pick the node that advertises the lowest $\text{Var}(\hat{\delta}_{\text{master}(k)})$ to ensure lowest $\text{Var}(\hat{\delta}_2)$. Scaling this concept to a large multi-hop network scenario, by selecting a master based on their advertised FEV, each node in the network is assured the highest stability clock possible and the routing tree constructed would be optimal for disseminating time information.

Interestingly, this process can be accomplished directly from timestamp messages without master nodes explicitly advertising their $\text{Var}(\hat{\delta})$. This makes the protocol robust to erroneous advertisements and allows backward compatibility to existing time synchronization protocols. Observe that N_2 receives synchronization messages from both potential masters N_1 and N_3, even after it has selected one of them. Using these messages, N_2 can continuously and simultaneously compute $\text{Var}(\hat{\delta}_{2/1})$ and $\text{Var}(\hat{\delta}_{2/3})$ – FEV at N_2 with respect to N_1 and N_3 respectively. This is implemented by computing the statistical variance of the output of Equation (7). Referring Equation (23) again, we see that with all other terms held equal, $\text{Var}(\hat{\delta}_{2/1})$ and $\text{Var}(\hat{\delta}_{2/3})$ differ only in the FEV of the master clock, $\text{Var}(\hat{\delta}_1)$ or $\text{Var}(\hat{\delta}_3)$. Therefore, selecting the master clock by computing and comparing $\text{Var}(\hat{\delta}_{2/x})$ for each potential master node x is equivalent to learning FEV from each node explicitly. The trade-off to this process is the delay in acquiring sufficient $\hat{\delta}_{2/x}$ samples for maintaining statistical significance of $\text{Var}(\hat{\delta}_{2/x})$.

The above procedure can not only be used to construct the time information tree but also to preserve it against clock frequency changes due to environmental variation. Frequency error changes at clocks close to the root will propagate further down the tree so a time information path should try to circumvent nodes that show high FEV. In order to track this error, we track changes in $\text{Var}(\hat{\delta}_{2/1})$ and $\text{Var}(\hat{\delta}_{2/3})$ over time for N_2, for example. From Equation (23), we learn that $\text{Var}(\hat{\delta}_{2/x})$ could fluctuate due to two factors – σ_2^2 and $\text{Var}(\hat{\delta}_x)$. When the frequency error at N_2 increases, due to local changes in temperature, say, σ_2^2 increases. This manifests as an increase in both $\text{Var}(\hat{\delta}_{2/1})$ and $\text{Var}(\hat{\delta}_{2/3})$ equally, indicating a local problem. Whereas, if the frequency error at either N_1 or N_3 increases, only one of $\text{Var}(\hat{\delta}_{2/1})$ or $\text{Var}(\hat{\delta}_{2/3})$ will increase, indicating that N_2 must switch masters to the one with a lower variance. In this way, by continuously monitoring the relative variation of $\text{Var}(\hat{\delta}_{2/x})$ across potential master nodes, N_2 will always select the best clock to synchronize to. This notion is easily extendible to the rest of the network, resulting in distributed preservation of the optimal time information tree in spite of environmental changes causing clock instability.

2.2 Root Clock Selection

It is expected that the base station node is equipped with a high stability time keeping device so that the time information tree is rooted at the same node as the data collection tree. When this is not the case, one may like to extend the master clock selection mechanism per node based on frequency error variance to select the root clock for the entire network. The root clock must be the most stable clock in the network. It would seem that if every node picks a master with the lowest frequency error variance, that a root will be selected consequently and that this root would be the most stable one in the network. This is untrue.

When a node k computes its frequency error variance $\text{Var}(\hat{\delta}_{k/j})$, it does so from the output of Equation 7 with respect to another node j. Because FEV is always relative, there is at least one unknown in the system of equations that is unsolvable. To see this, consider our three node linear network once again, but this time make no assumption about the stability of N_0's clock. Each node computes $\text{Var}(\hat{\delta}_{k/j})$. Thus, each node now has two FEV values and one can form a system of equations based on Equation 23 to solve:

$$\text{Var}(\hat{\delta}_{k/j}) = \frac{f_0^2}{w\sigma_w^2 T^2}\cdot\sigma_k^2 + \kappa\cdot\text{Var}(\hat{\delta}_j) \quad (24)$$
$$\forall k \in \{0,1,2\}\ j \in \{0,1,2\}\backslash k$$

This leads to six equations in six unknowns – σ_k^2 and $\text{Var}(\hat{\delta}_k)$ for $k \in \{0,1,2\}$ but the system is under-determined. The reason is that when a node k computes $\text{Var}(\hat{\delta}_{k/j})$, it does so to decide which of the other two j nodes it should pick, but not itself. Since it is impossible to compute frequency error without a reference, it is impossible for a clock to know its own stability and pick itself as the root. Thus, while the metric of FEV works well for master selection from a set of candidates, it does not work when one of the candidates is the node itself.

2.3 Timing Loops

A key assumption we used in deriving frequency error variance as a gradient time routing metric was that $\sigma_1^2 = \sigma_2^2 = \sigma^2$, which implies a steady state homogeneous network. When conditions in the network change, however, this assumption is violated and it could result in timing loops in the network akin to forwarding loops in network routing. To show why this might happen, compare Equations (9) and (21) that model the FEV at N_1 and N_2 respectively. If another node, say N_4, can hear messages from both N_1 and N_2, it would seem intuitive to expect N_4 to select N_1 as its master since it is closer to the root and N_2's clock is derived from N_1 anyway. However, N_4 selects N_1 as master only if:

$$\text{Var}(\hat{\delta}_1) < \text{Var}(\hat{\delta}_2) \quad (25)$$
$$\sigma_1^2 < \sigma_2^2 + \kappa\cdot\sigma_1^2 \quad (26)$$
$$\sigma_1^2(1-\kappa) < \sigma_2^2 \quad (27)$$

In the ideal limiting case, we would like $\kappa \to 0$. Thus, N_4 selects N_1 as master over N_2 only if : $\sigma_1^2 < \sigma_2^2$. Since the reverse is a likely possibility considering variations in the network, N_4 might choose N_2 at times and N_1 at other times.

This behavior causes route oscillations. A worse situation can occur if at some instant later $\text{Var}(\hat{\delta}_4) < \text{Var}(\hat{\delta}_1)$, even temporarily. If at this point, N_2 can hear synchronization messages from N_1 and N_4, N_2 will pick N_4 as its master. Coupled with the fact that N_4 could have chosen N_2 as master itself, N_4 and N_2 will form a timing loop picking each other as master clocks. They will drift away from the rest of the network and will never pick N_1 again as master since it will always be worse in terms of FEV. We will present a solution to the timing loop problem in Section 3.4.

3 The Time Information Routing Protocol

Section 2 outlined two approaches on how to use the frequency variance as a routing metric. In this section, we will go into details of these two approaches while presenting the Time Information Routing Protocol (TIRP).

The goal of TIRP is to optimally disseminate the global time established at a root synchronization node within a multi-hop network. The key function behind TIRP is the measurement of frequency variance with respect to a frequency reference clock. Previous work [16] proposed to select the synchronization root as the node with the lowest node id. This is not ideal for TIRP because it does not guarantee that the node with the lowest id has a stable clock frequency. Ideally, we would like to choose the clock with the lowest frequency variance within the whole network. But as we showed in Section 2.2 this can not be accomplished.

We assume that the synchronization root has access to a stable clock source, like a GPS receiver, or a temperature compensated crystal oscillator (TCXO). This assumption fits well within the common system architecture of wireless sensor networks for data collection. The gateway node or sink node is usually a more capable platform with access to a larger energy storage. We can equip this node with a more stable time source, even if it consumes more energy.

In TIRP, a node chooses its synchronization parent within its radio neighborhood according to a statistical measurement of frequency stability. We showed in Section 2.1 that the estimated frequency variance $\text{Var}(\hat{\delta})$ has the right properties, i.e., it is monotonic along the path. As described earlier, there are two ways how a node k with neighborhood V_k retrieves these measurements.

1. Nodes j in V_k advertise their $\text{Var}(\hat{\delta}_{j/\text{master}(j)})$ inside the synchronization beacons
2. Node k calculates $\text{Var}(\hat{\delta}_{k/j})$, $\forall\ j \in V_k$ based on the received synchronization beacons.

Since these two systems are significantly different, we will call them *Cooperative - TIRP* (C-TIRP) and *Autonomous - TIRP* (A-TIRP). In both cases, node k minimizes the synchronization objective of attaining the highest clock stability, i.e., it chooses master(k) from $j \in V_k$ such that

$$\text{master}(k) = \arg\min_j \text{Var}(\hat{\delta}_{j/\text{master}(j)}), \quad \forall j \in V_k, \qquad (28)$$

for C-TIRP or

$$\text{master}(k) = \arg\min_j \text{Var}(\hat{\delta}_{k/j}), \quad \forall j \in V_k, \qquad (29)$$

for A-TIRP respectively.

There are several trade-offs between C-TIRP and A-TIRP. C-TIRP has a smaller memory and calculation footprint since every node tracks only one frequency statistics, while in A-TIRP, a node has to track the frequency statistics of every potential synchronization neighbor. Nevertheless, in C-TIRP a node has to trust the variance calculated by their neighbors, while in A-TIRP a node makes its own decisions. This adds robustness to erroneous or even malicious advertisements since a node can not fake a lower frequency stability.

We implemented and evaluated both the C-TIRP and A-TIRP algorithm and the reminder of this section will go into some more protocol details. For both implementations, we assume that each node has a local clock source and a mechanism for low-level precision message time stamping. These two mechanisms allow a node to measure clock offsets $o(\cdot)$ between the local clock $c(\cdot)$ and the global clock advertised by a radio neighbor $r(\cdot)$.

3.1 Calculating Frequency Error Variance

Algorithm 1 Frequency Variance Calculation

```
regressionTbl[n%N] = (c(n), o(n))
n += 1
if n > 1 then
   δ̂ = calculateFrequencyError(regressionTbl)
   VarTbl[m%M] = δ̂
   m += 1
   if m > 1 then
      var = calculateVariance(VarTbl)
   end if
end if
```

Algorithm 1 gives a high level overview on how a node estimates the frequency error variance of the local clock with respect to a remote clock. The algorithm consists of three steps:

1. Collect pairs of *local time* $c(n)$, *remote time* $r(n)$ timestamps from synchronization beacons and store $c(n)$ and the offset $o(n) = r(n) - c(n)$.
2. Once enough pairs are collected, calculate the frequency error $\hat{\delta}$ using a regression algorithm on the $(c(n), o(n))$ pairs and store it in memory.
3. Once a significant number of frequency errors are collected, calculate the statistical variance for the frequency error measurements.

Note that the $(c(n), o(n))$ pairs are stored in a circular buffer of size N, and the frequency error estimates in a circular buffer of size M. Thus, after an initial phase of filling the buffers, every new $(c(n), o(n))$ pair will result in a new frequency error estimate, and thus a new value for the frequency error variance. The size of the circular buffers depends on the clock frequency, and resynchronization rate. They should be chosen such that the variance becomes statistically significant.

3.2 Cooperative - TIRP

C-TIRP resembles the architecture of wireless collection routing protocols, where every node advertises its current path metric. A node chooses the neighbor with the best path metric as its routing parent. In C-TIRP, this path metric is the calculated frequency variance with respect to the nodes synchronization master $\mathrm{Var}(\hat{\delta}_{k/\mathrm{master}(k)})$.

Message Format: A C-TIRP synchronization message contains the *timestamp* $r(\cdot)$, the *root id*, a *hop count*, and a compressed representation of the frequency variance $Y(\mathrm{Var}(\hat{\delta}_{k/\mathrm{master}(k)}))$. The *timestamp* $r(\cdot)$ contains the global time estimated by the transmitting node. The *root id* field contains the id of the synchronization root that provides the global time, while the *hop count* is the number of intermediate hops between the transmitting node and the root. This field is used to avoid time loops, which will be discussed in Subsection 3.4.

The compressed variance $Y(\cdot)$ is an 8-bit representation of the locally calculated variance. C-TIRP calculates the variance using floating point math, and converts it to an 8-bit representation using a mapping function. A linear mapping with constant resolution is not sufficient as the resolution for smaller variances is inadequate. Therefore, we give more resolution on small variances, and less resolution the bigger the variance becomes. The intuition is that small changes at low variance are more important to resolve than small changes at high variance. If a node exhibits high variation in its frequency error, we know that the node itself is bad, and thus can ignore it.

One mapping following this requirement is of the form

$$Y(\mathrm{var}) = Y_{\max} \cdot \left(1 - e^{-\frac{\mathrm{var}}{\mathrm{var}_0}}\right), \tag{30}$$

where $Y_{\max} = 255$, and var_0 represents the variance which will indicate 63.2% of $Y_{\max}$.

Before a node calculates its own frequency variance, it must select a synchronization parent from its radio neighborhood. At initialization, only the synchronization root can start to send out synchronization messages and thus the first nodes calculating their frequency variance will be the immediate root neighbors at 1-hop distance. Once these nodes collected enough information for statistical significance of their variance, the nodes become synchronized and start themselves sending out beacons. Thus the 2-hop nodes will start calculating their variances before they start sending out their own beacons. This repeats until all the nodes in the network can overhear another node's beacons, and the whole network becomes synchronized.

3.3 Autonomous - TIRP

The main idea behind A-TIRP is reduced reliance on other nodes' calculations. A-TIRP exploits the fact that the time beacons themselves indirectly include the accumulated path-frequency changes of all the nodes in the synchronization branch, i.e., the second term of Equation (23). A node receiving the synchronization beacons can extract the path variance by calculating the frequency error variance of the beacon time with respect to the local clock. This is drastically different from regular network routing, where the routing metric can not be extracted from the routing beacons.

Message Format: A-TIRP uses the same message format as C-TIRP, except that it does not need the frequency variance field $Y(\cdot)$. The rest of the fields are identical.

The major difference between A-TIRP and C-TIRP is the amount of storage and calculation a node has to perform. While in C-TIRP, a node tracks only one frequency error variance, in A-TIRP the node needs to calculate the frequency error variance for every radio neighbor. Thus, the storage and computation requirements for A-TIRP increase linearly with the size of the radio neighborhood.

Memory Considerations: We can imagine that for dense wireless networks a node limits itself to a subset of radio neighbors. For example, assume a node can only track a maximum of 10 nodes. Initially, the node would start calculating the frequency variance of the first 10 neighbors it overhears. Once the node calculated the variance of each neighbor, it could choose the 5 nodes with highest frequency variance and replace them with 5 new neighbors. This would allow the node to eventually go through all the radio neighbors finding the most stable nodes.

3.4 Time Loops

A common problem in wireless routing protocols are routing loops, where messages are sent around in a loop of nodes. A similar phenomena in TIRP are *time loops* as was described in Section 2.3, where nodes synchronize to each other in a loop. These time loops have to be avoided as they introduce large global synchronization errors for the nodes participating in the loop.

A common technique in routing to detect loops is duplicate packet inspection, i.e., routers actively search for identical packets. This is often achieved by means of the Time To Live (TTL) value of a packet. In TIRP, we decided to use an incrementing *hop* variable that gets incremented by one at each hop. This is similar to the THL field in the Collection Tree Protocol [9].

A node detects a time loop by observing the *hop* count of the beacon message. If the *hop* count keeps increasing with every beacon received, then there exists a loop and the beacon messages should be ignored. At the same time, if this particular neighbor was the chosen synchronization parent, then we should switch to a different neighbor.

4 Evaluation

We implemented A-TIRP and C-TIRP in TinyOS for the TelosB sensor network platform by extending the Flooding Time Synchronization Protocol [16]. The choice of platform limits our time resolution to a 32kHz clock, since the TelosB is not equipped with a high frequency crystal oscillator. At the same time, the timestamping accuracy of the Texas Instruments CC2420 radio chip employed on the TelosB has a latency between transmit and receive timestamp of 3.15 μs, and a variance of about 45 ns [23]. This is well below the 1-tic resolution of the 32kHz clock, and thus a uni-directional synchronization process, as is used in FTSP, achieves the same synchronization accuracy as a bi-directional synchronization exchange. Nevertheless, we show in our evaluation how temperature can introduce large time errors even at these low frequencies, and how TIRP prevents these errors from propagating through the network.

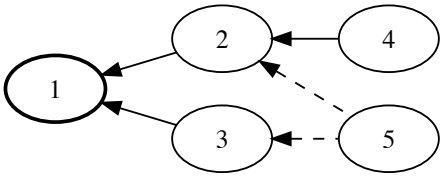


Figure 2. Experimental node connectivity. Node 5 has the possibility to switch its master from 2 to 3 if necessary, while node 4 was forced to use node 2 exclusively. This allows to show the difference between a node that switches its master, and a node that does not switch, in case the master becomes unstable.

4.1 C-TIRP vs. Routing Integrated

The experimental scenario to test C-TIRP involved a small 5 node network. Figure 2 illustrates the communication links within the network. Node 1 is the dedicated time root, and nodes 2 and 3 are the 1-hop synchronization nodes. In order to show the disadvantages of a routing integrated time synchronization protocol, we forced node 4 to always synchronize using node 2's beacons. Node 5 was free to choose between nodes 2 and 3 and is allowed to switch dynamically between the two nodes, depending on their advertised FEV. To induce a change in node 2's frequency, we started to heat node 2 about 400 seconds into the experiment. This simulates an abrupt change in environmental temperature that will impact the synchronization accuracy of node 2, but not the communication network itself.

We used the same mapping function as was proposed in Equation (30) to map the calculated frequency variance to a 1 byte value, and set the 63.2% mark to $var_0 = 4 \cdot 10^{-12}$. This represents a standard deviation of 2 ppm, i.e., if the standard deviation of the frequency error is measured to be 2 ppm, then the *var* field in the C-TIRP message will be set to 161. It is to note that a standard deviation of 2 ppm is bad for the purpose of synchronization, and that if the temperature environment is stable, the standard deviation for this experiment was closer to 0.1 ppm.

Figure 3 shows the effect that heating node 2 has on nodes 4 and 5. Shortly after node 2 heats up, node 2's synchronization accuracy becomes worse. Since node 4 uses node 2 as synchronization parent, it too incurs this inaccuracy, even though node 4 is in a stable temperature environmental. On the other hand, node 5 which is also at a 2-hop distance from the synchronization root does not change its accuracy at all.

Figure 4 depicts the explanation of node 5's behavior. While in the beginning, both node 4 and 5 use node 2 as synchronization parent, node 4 was forced to stay with node 2. Node 5 was however free to change its parent according to the C-TIRP algorithm. As we can see on Figure 4, node 2's stability sharply increases once the node gets heated, while node 3's stability stays low since it is away from the heat source. Thus, about 450 seconds into the experiment, node 5 switches its synchronization master from node 2 to node 3, successfully circumventing the instabilities it would else have experienced.

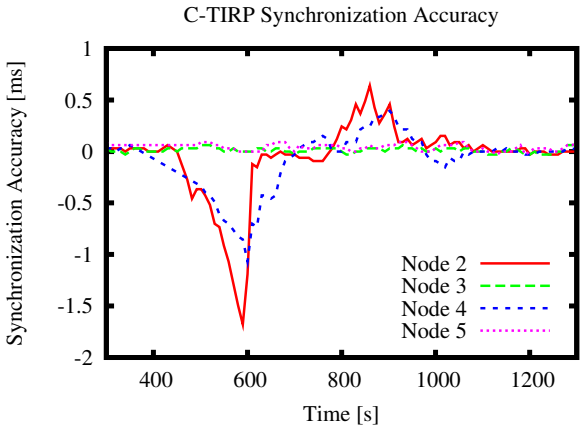


Figure 3. C-TIRP runtime evaluation. 400 seconds into the simulation of the network depicted in Figure 2, a hot air blower started to heat node 2. We can see how node 4 gets affected by this change, while node 5 switches its master to node 3 (see Figure 4).

4.2 A-TIRP vs. Flooding

We implemented A-TIRP in TinyOS by modifying the Flooding Time Synchronization Protocol implementation. While FTSP throws away synchronization messages with the same sequence number, A-TIRP uses them to calculate the per-neighbor FEV.

We performed two experiments to evaluate the performance of A-TIRP. For both experiments, we deployed two identical networks, one running FTSP, the other running A-TIRP, where nodes with the same id were collocated. To avoid collisions, we chose a different radio channels for each network. Running both networks concurrently and collocating the nodes allows us to make a comprehensive comparison between the two algorithms because the nodes experience the same temperature environment.

Figure 5 illustrates the network connectivity. Each node in the eleven node network can communicate with two neighbors of lower id, and two with a higher id. This network architecture allows the dynamic routing around a node with a high FEV value.

Node 1 was a specially modified TelosB mote with a Maxim DS32kHz TCXO [18]. Figure 6 shows a picture of the modified node. Two such nodes, one for the FTSP, the other for the LK-TIRP network, provided a temperature invariant and stable time source.

We performed two different experiments. The first experiment, described in Subsection 4.2.1, performs a controlled heating of one node in the network. This allows a detailed look at what happens if a node within a network becomes unstable. While this is a controlled environment, it allows us to rule out other effects of inaccuracies in the synchronization process.

In the second experiment, described in Subsection 4.2.2, we investigate how A-TIRP and FTSP handle a mixed indoor-outdoor node deployment. We observed the network time synchronization accuracy over a period of 5 days. This

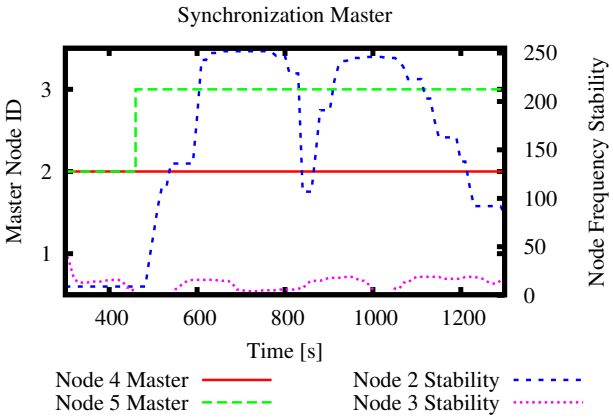


Figure 4. C-TIRP master selection example. Node 2's frequency stability becomes worse as a heat source is applied to it after 400 seconds. Therefore, node 5 switches its master to node 3, which is not affected by the heat source.

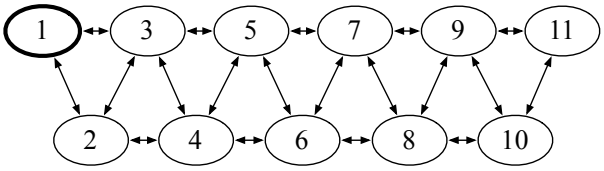


Figure 5. Experimental network architecture. Node 1 features a TCXO and acts as synchronization root.

experiment illustrates what happens if some nodes are exposed to a more variable climate, while other nodes are not. This experiment directly shows how a real deployment would perform, where a mixed environment of shaded, sunny, windy, and indoor/outdoor nodes can be found.

4.2.1 Controlled Heating

The change in environmental temperature is the most significant cause of changes in local clock frequency. Thus, heating a node using a hot air source will inevitably change the crystal frequency of an embedded system.

For this first experiment, we set the resynchronization rate to 30 seconds. This is the same value as the authors of FTSP [16] used in their evaluation. While longer resynchronization rates would be appreciated in actual deployments, we show that even at 30 second intervals we observe significant temperature stability problems.

Figure 7(b) illustrates the effect of heating node 3 in the two experimental networks depicted in Figure 5. Since both networks run concurrently, and the nodes 3 of FTSP and TIRP are collocated, they both experience the same temperature changes. We can see this in the synchronization error accumulating in node 3 for both networks. However, in FTSP the instability of node 3 starts to propagate down the network to the nodes further away from the root. Figure 7(a) depicts this as an inaccuracy ripple. TIRP protects itself against such errors by excluding node 3 from propagating time information.

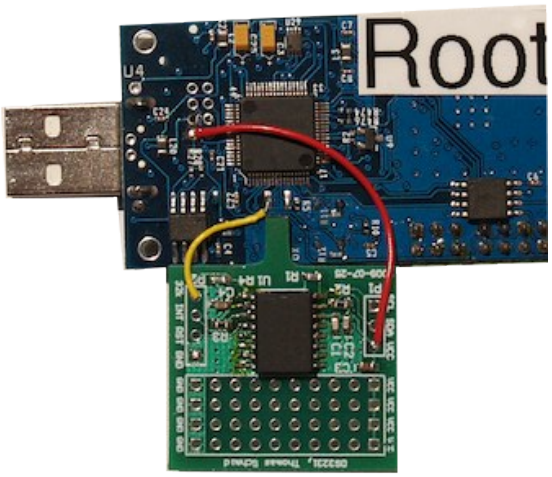


Figure 6. Modified TMote Sky used as a stable time reference. We replaced the regular 32 kHz tuning fork crystal with a precision TCXO from Maxim [18].

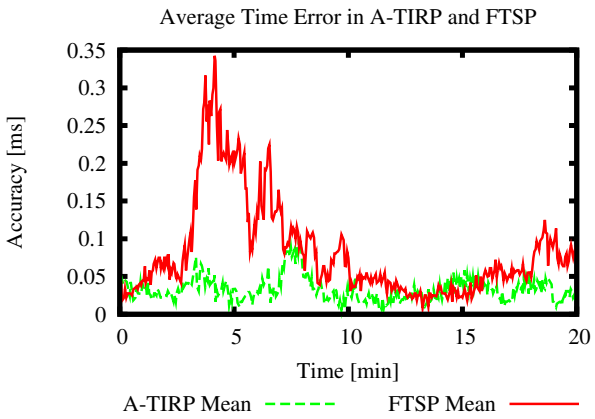
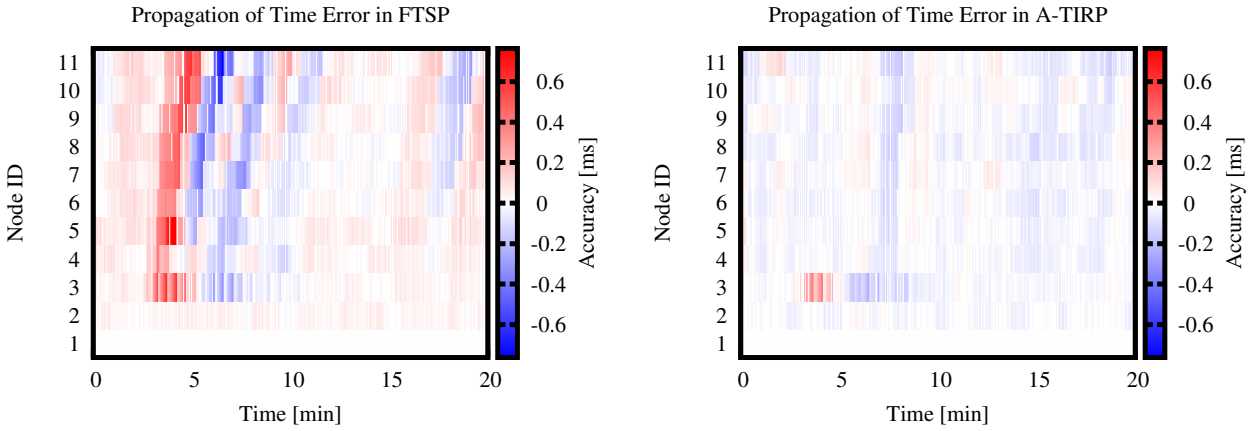


Figure 8. Average network synchronization error of the controlled heating experiment. The change in just one node impacts the whole network by increasing the average synchronization accuracy of FTSP.

We can also look at the average and maximum network synchronization accuracy. Figure 8 shows this for both networks. We observe that the average synchronization accuracy of TIRP stays below 100 μs, even though node 3 becomes desynchronized. The error introduced through node 3 in the FTSP network however propagates to the other nodes, decreasing the average synchronization accuracy to above 300 μs, or more than 3$\times$ that of TIRP.

4.2.2 Long-Term Indoor - Outdoor Experiment

The artificial heating of a node using a hot air source can be seen as a simulation of an industrial setup, where nodes might be placed close to machinery that exhaust air or fumes. One example are servers in a data center. While idling, the machines stay cold. But when the utilization suddenly increases, hot air will exhaust from the vents and heat a potential temperature sensing node.



(a) Accuracy FTSP (b) Accuracy A-TIRP

Figure 7. Controlled heating experiment. After 2 minutes, a hot-air source gets applied to node 3 in the network from Figure 5. We can see how the error propagates in the FTSP network in Figure (a), while A-TIRP in Figure (b) avoids any further error propagation.

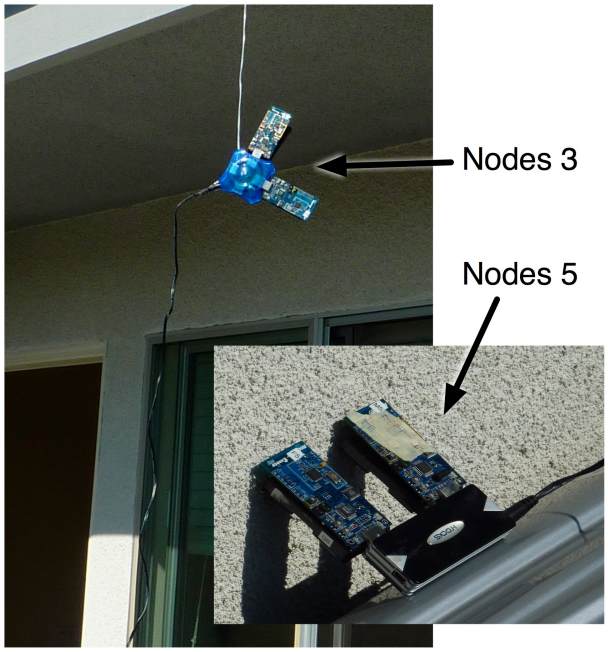


Figure 9. Collocated outdoors nodes 3 and 5. Nodes 3 were wind and sun exposed, while nodes 5 were wind protected.

While industrial applications can see drastic changes in environmental temperature, similar changes can be observed in nodes performing environmental sensing. Sunlight exposed nodes for example can exhibit drastic changes in temperature due to sun/shade transitions.

To test the performance of TIRP in such a scenario, we partitioned the two experimental setups into indoor and outdoor nodes. More specifically, we placed two of the 11 nodes outdoors, while the rest remained indoors. Both the FTSP and A-TIRP nodes 3 from the network depicted in Figure 5 were placed in a sunlight exposed area where wind was able to reach them. Both nodes 5 were placed in a sunlight exposed area which was wind protected. Figure 9 depicts their collocation setup. The rational behind this particular constellation was to test if wind introduces more temperature fluctuations, or if it helps cooling the nodes to reduce drastic temperature changes.

We observed the mixed indoor – outdoor network for several days using the same 30 second resynchronization interval as used in the controlled heating experiment from Subsection 4.2.1. Figure 10 shows the average network synchronization error for every 15 minutes over the period of 7 days of both the FTSP and A-TIRP network. While the observed errors are not as big as in the controlled heating experiment, there is a clear rise in the synchronization error during day time. A-TIRP is not fully immune to changes in temperature because the nodes that experience the change will be affected by them. However, A-TIRP's average synchronization accuracy rises only slightly during daytime, while FTSP's error almost triples.

The effects of changing temperature on synchronization accuracy becomes more pronounced as the resynchronization rate increases. While already at 30 seconds an increased error in FTSP is clearly visible, we redeployed the same network with a resynchronization rate of 60 seconds. Figure 11 shows a detailed per-node error analysis of 9 hours of the 18 hour deployment.

As expected, the synchronization error significantly increases, now reaching 1 ms in the worst case. Figure 11 shows as in the controlled heating experiment, that the two unstable nodes 3 and 5 are the source of synchronization error in FTSP. In A-TIRP, the two nodes still have higher synchronization errors than the other nodes. But the other nodes only rarely choose them as synchronization masters since their FEV is unstable.

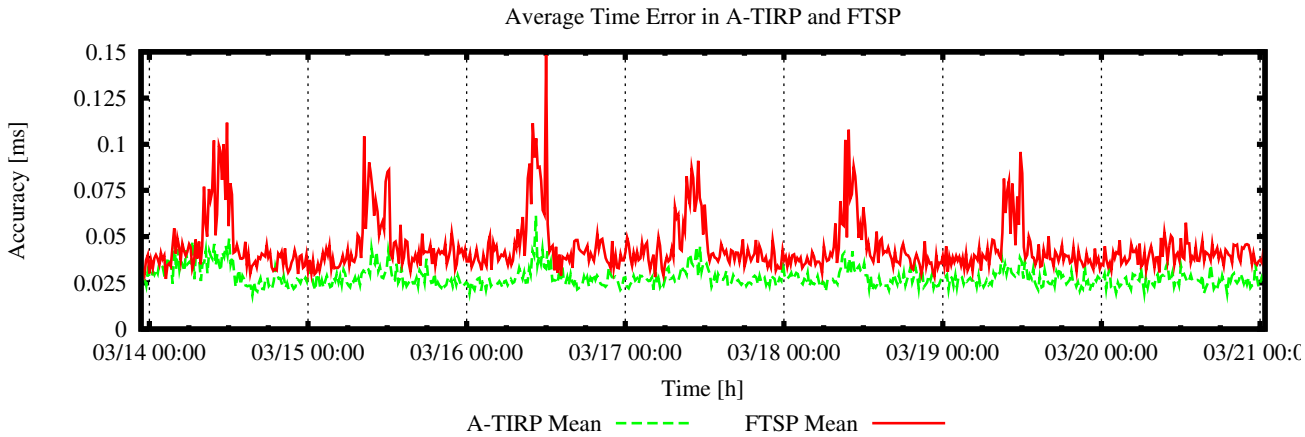


Figure 10. Average network synchronization accuracy of the mixed indoor-outdoor deployment. The average was taken over all nodes and for every 15 minutes. Even though the synchronization rate was 30 seconds, we can see effects of temperature impacting the FTSP network during day time. This behavior becomes more pronounced the longer the synchronization interval.

Another way of looking at the same data is by generating the synchronization graph depicting which node chooses which neighbor as synchronization master. Figure 12 depicts this information in two different ways for the times before 9h30, and the time after 9h30. We can observe that during the night, the network makes heavy use of node 3 and 5 as synchronization master. The cause is a stable night time temperature that keeps the crystals at a stable frequency. But as the day progresses, and temperatures start to fluctuate, node 3 gets chosen less often, and node 5 gets almost completely ignored as synchronization parent. This shows the effectiveness of A-TIRP and how it routes information around unstable nodes.

5 Discussion and Future Work

There are several knobs that can tweak the performance and accuracy of TIRP. In the following sections, we discuss three such improvements that address changes to TIRP for higher frequency clocks, how we can avoid time-loops and thus remove the hop-count field, and how a node can reduce its own frequency variance alternatively changing its resynchronization rate.

5.1 High-Frequency Clocks

The current implementation of TIRP relies on a 32 kHz crystal and a uni-lateral communication scheme. This limits the synchronization accuracy to about 30.5 μs or 1 jiffy. If we want to get higher precision, a higher frequency clock is imperative [23]. But at higher frequencies, a uni-lateral communication scheme will introduce new error sources as time of flight becomes more significant.

The solution is to move to a bi-lateral synchronization exchange as used in IEEE 1588 [6] or proposed in TPSN [8]. However, bi-lateral synchronization exchanges necessitate a tree structure since a synchronization exchange now can no longer be performed as a broadcast as in FTSP, but needs to be two unicast messages between the node and its selected synchronization master.

C-TIRP can directly apply a bi-lateral synchronization exchange since it only needs to overhear neighbor traffic, and extract the FEV metric from them. A-TIRP is slightly more complicated as the neighbor messages are used to locally compute their FEV metric. But fortunately, the frequency error estimation is immune to constant offset errors, like errors introduced through radio latency or time of flight. Therefore, overhearing neighbor's unicast messages with their synchronization masters is enough to extract their FEV metric.

5.2 Removing Hop-Counts

As Section 2.3 showed, the current choice of FEV has significant limitations and the potential of time loops. In order to remove the hop-count field we need a truly monotonically increasing time routing metric.

If we could measure the individual per link frequency variance, then a node could just add the link variances to the FEV the synchronization master reports. This metric would be truly additive along the time information path since no κ would decrease the additive variance.

However, in order to get the per link frequency variance we will have to significantly change the structure of the synchronization message, and thus TIRP would not be backwards compatible anymore. In addition to the global time timestamp, a node would also have to send the unmodified local time together with the FEV of the synchronization master. Using these three fields, a node can now calculate the per link variance by using the unmodified local time, but still get a global time reference from the global time timestamp.

5.3 Adjusting the Synchronization Interval

As we can observer from Equation (9) and (23) decreasing the resynchronization rate T will also decrease the frequency error variance. Thus, a node has two possibilities to increase the measurement stability. First, it can switch to a different, more stable master, or second, it can simply increase the synchronization interval.

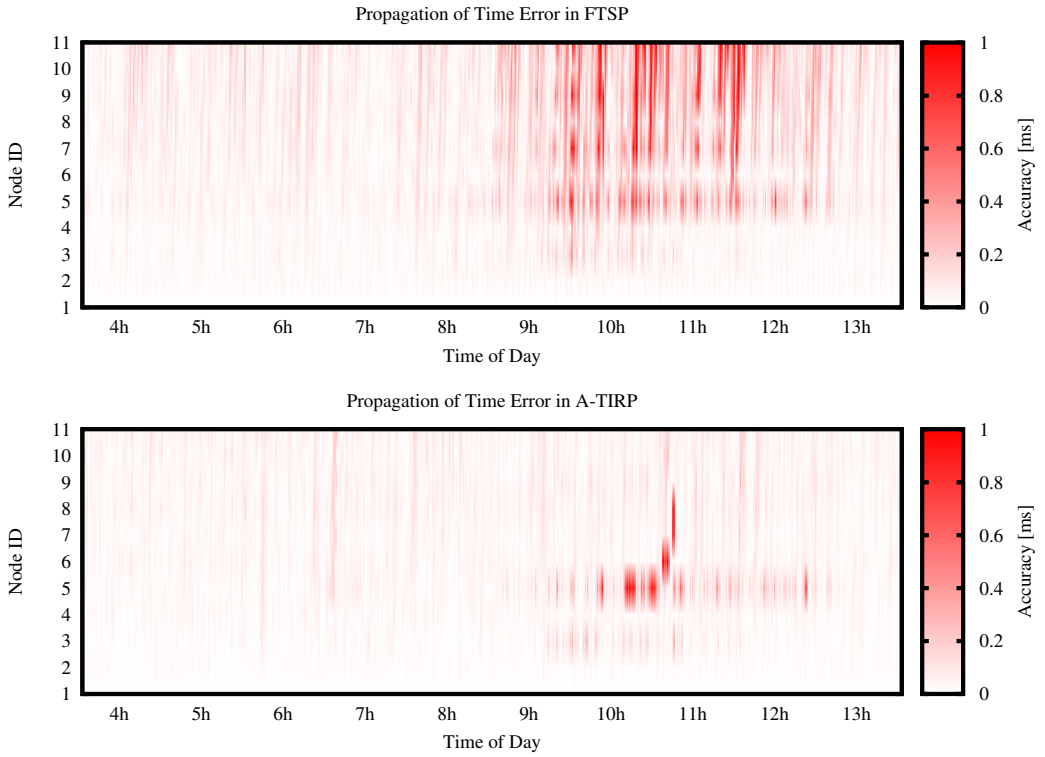
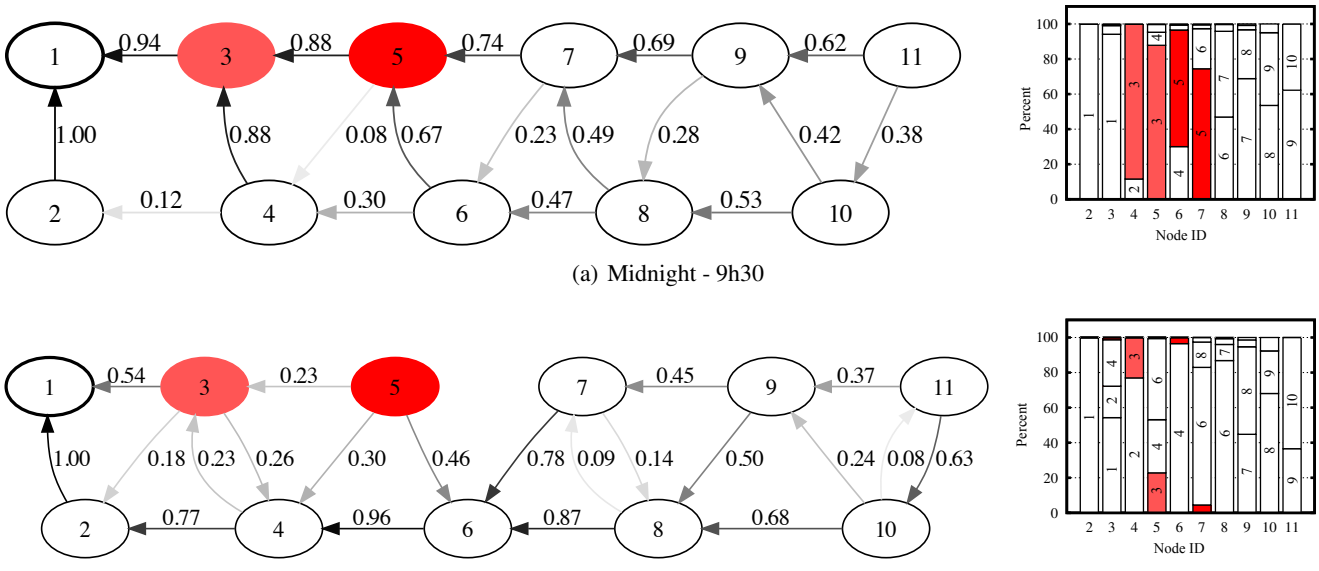


Figure 11. Propagation of time errors in A-TIRP and FTSP in the mixed indoor-outdoor deployment and a synchronization rate of 60 seconds. Node 3 and 5 were both sun exposed, while node 3 was also wind exposed. This helped to cool the node and thus decrease the change in frequency error. We can see how starting at 6h (sunrise), small time ripples start to manifest both in FTSP and A-TIRP. By 9h30, the changes in temperature become significant and FTSP starts to have large time ripple problems, even though only two nodes were in an unstable environment.



(a) Midnight - 9h30

(b) 9h30 - Noon

Figure 12. Time information routing graph before and after 9h30. The edges indicate the probability with which a certain path was taken. The stacked bar graph represents the same data. Note that for clarity, edges with <5% were removed. We can see how after 9h30, node 3 and 5 are avoided as synchronization masters in order to mitigate time error propagation.

Switching to a different master is more desirable since increasing the synchronization interval increases the communication overhead of the synchronization process. However, there are situations where increasing the synchronization interval is the only solution to increased stability. For example, if the node itself is the cause of instability due to local changes in temperature, changing the master node will not help in increasing the frequency stability. Only decreasing the synchronization interval will work in that case.

6 Related Work

Using clock characteristics to chose a synchronization master is not entirely new. IEEE 1588 [1], the Precision Time Protocol for measurement and control systems, relies on clock characterizations provided by a manufacturer of an instrument. Every clock in an instrument gets assigned to a class. This clock class is comparable to the Stratum definition in NTP. Furthermore, each clock estimates its clock accuracy. The clock accuracy indicates the expected accuracy of a clock when it is the master, or in the event it becomes a master. The last measurement is a scaled log variance representing the characteristics of the local clock as measured by a perfect clock. The standard specifies two possible ways of calculating this value:

1. A static constant determined by the manufacturer
2. Computed based on measurements and the environment

While the first case is a static assignment, the second is similar to our approach. However, the standard does not explain how the measurements get collected or how the environment gets integrated into these measurements.

Additionally to metrics for the master node, IEEE 1588 provisions for a synchronization slave to measure the variations within its parents clock. However, these fields are optional and don't get transmitted to other nodes. Therefore, they become purely informational and can not be used in a multi-hop scenario.

For NTP, clock frequency variation is less important than path latency estimation. The multi-hop nature of NTP and difference in path length between the synchronization host and the multiple synchronization servers introduces large error in the offset measurements. These errors come from changing packet queue lengths in intermediate routers and switches. NTP uses Marzullo's algorithm [17] to derive a consensus between the different measurements coming from different servers. While wireless sensor networks disseminate the time information over multiple hops, intermediate nodes participate in the synchronization process. This removes the hard to predict queue latencies because nodes timestamp the reception and transmission of messages.

The Flooding Time Synchronization Protocol [16] (FTSP) takes a different approach towards time information dissemination. In order to reduce redundant information, FTSP considers only the first message it receives from a flood, and drops all the others. The intuition is that the first message will come from the node that most recently heard from the root node, and is thus the most recent time estimate. The disadvantage is that if the first node overheard is an unstable node, then our synchronization accuracy will suffer and the errors propagate through the network.

Earlier work showed [22], and more recent work formalized [13], the need for rapid dissemination of time information in a multi-hop network in order to minimize the propagation of frequency errors. While FTSP considers only the first overheared message of a flood, the rebroadcasting of this information is an independent timer with potential significant different phase. Thus, significant time can pass until a node rebroadcasts the new time information.

Lenzen et. al [13] propose with PulseSync to modify FTSP to re-broadcast time information as soon as it has been received. By minimizing the on-node dwell time of timing information, PulseSync reduces the introduced inaccuracies of local clock instability. While PulseSync solves the same symptoms TIRP addresses, it relies on rapid flooding of the network. Rapid flooding in low duty-cycle wireless networks is difficult [15]. While the end result of TIRP and PulseSync are similar, TIRP works even if the time information is disseminated in an asynchronous fashion. However, the approaches are orthogonal and could be combined for more robustness.

Recently, Sallai et al. [22] and Neto et al. [20] proposed to piggy-back the time synchronization messages on the routing beacons. This is motivated by the reduction of communication overhead and the ensuing gains in energy efficiency. However, it ignores the difference between the clock distribution and the routing tree. This can have harmful effects on the synchronization accuracy as nodes are forced to use synchronization neighbors according to the network routing tree, and not according to the best clock source available.

The Reference-Broadcast Synchronization (RBS) algorithm is used to synchronize a set of receivers within a single broadcast domain. Elson et al. [7] describe how the RBS system can scale to a multi-hop architecture by using boundary clocks that translate from one broadcast domain into the next. Using these boundary clocks Elson describes how a routing tree can be established between the different broadcast domains. He even proposes a different routing metric using the residual error (RMS) of the linear fit to the broadcast observations. Thus a minimum error conversion can be found between any two nodes of the network using an algorithm like Dijkstra or Bellman-Ford. However, RBS fails to implement this strategy and show its effectiveness, especially in a scenario with a dynamic environments.

7 Conclusion

Much of the prior work on integrating time synchronization with routing has focused on the efficiency gains from doing so. In this paper, we show that naïve integration of time synchronization and routing extracts a harmful accuracy penalty. Since such errors are critical for applications like acoustic source localization, which require accurate and precise time synchronization, reducing these errors improves application performance. We show that by decoupling the clock distribution tree from the routing tree, much of this error can be reduced. This decoupling is not literal – messages from many surrounding neighbors will still be received by each node – but rather logical – in that we wish to better select which of the many neighbors to use as the parent for time synchronization purposes.

We show the best choice for a parent – the one that minimizes error with respect to the root – is the neighbor that offers the smallest accrued frequency variance along the path. We formalize this variance as the critical path metric for time synchronization, much like hopcount is critical to distance-vector routing protocols. We propose two methods to estimate this variance. Our results show that by using variance as the metric for clock tree construction, the accuracy of time synchronization is greatly improved, time error propagation is greatly attenuated and compartmentalized, and transients due to sudden temperature changes are quickly detected and corrected. Although these results are promising, there are a number of other ways of estimating and propagating the variance metric at the heart of this work, but we leave a more exhaustive exploration of these techniques for future work.

Acknowledgments

Special thanks to Younghun Kim, Tian He, and the anonymous reviewers for their insightful and constructive comments. This material is supported in part by the U.S. ARL and the U.K. MOD under Agreement Number W911NF-06-3-0001, NSF Award Number CCF-0820061, #964120 (”CNS-NeTS”), and by UCLA’s NSF Science and Technology Center for Embedded Networked Sensing. Additional NSF support was provided under Grant #1019343 to the Computing Research Association for the CIFellows Project, and a fellowship from the Swiss National Science Foundation. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the listed funding agencies. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

8 References

[1] IEEE standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages c1 –269, 24 2008.

[2] I. Akyildiz and I. Kasimoglu. Wireless sensor and actor networks: research challenges. *Ad hoc networks*, 2(4):351–367, 2004.

[3] M. Ceriotti, L. Mottola, G. Picco, A. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta, and P. Zanon. Monitoring heritage buildings with wireless sensor networks: The torre aquila deployment. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks:w*, pages 277–288. IEEE Computer Society, 2009.

[4] D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.

[5] P. Dutta, D. Culler, and S. Shenker. Procrastination might lead to a longer and more useful life. *HotNets-VI*, 2007.

[6] J. Eidson. *Measurement, Control, and Communication Using IEEE 1588*. Springer, 2006.

[7] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, 36:147–163, 2002.

[8] S. Ganeriwal, R. Kumar, and M. Srivastava. Timing-sync protocol for sensor networks. *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, 2003.

[9] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 1–14. ACM, 2009.

[10] M. Golberg and H. Cho. *Introduction to regression analysis*. Southampton: WIT Press, 2004.

[11] B. Kusy, P. Dutta, P. Levis, M. Maroti, A. Ledeczi, and D. Culler. Elapsed time on arrival: a simple and versatile primitive for canonical time synchronisation services. *International Journal of Ad Hoc and Ubiquitous Computing*, 1(4):239–251, 2006.

[12] Y. Kwon and G. Agha. Passive localization: Large size sensor network localization based on environmental events. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 3–14. IEEE Computer Society, 2008.

[13] C. Lenzen, P. Sommer, and R. Wattenhofer. Optimal clock synchronization in networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 225–238. ACM, 2009.

[14] C. Liang, J. Liu, L. Luo, A. Terzis, and F. Zhao. RACNet: a high-fidelity data center sensing network. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 15–28. ACM, 2009.

[15] J. Lu and K. Whitehouse. Flash flooding: Exploiting the capture effect for rapid flooding in wireless sensor networks. *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2009.

[16] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, 2004.

[17] K. Marzullo and S. Owicki. Maintaining the time in a distributed system. *SIGOPS Oper. Syst. Rev.*, 19(3):44–54, 1985.

[18] MAXIM. DS32kHz, 32.768khz temperature-compensated crystal oscillator. `http://datasheets.maxim-ic.com/en/ds/DS32kHz.pdf`, April 2010.

[19] R. Musaloiu-E, A. Terzis, K. Szlavecz, A. Szalay, J. Cogan, and J. Gray. Life under your feet: A wireless soil ecology sensor network/. In *Proc. 3rd Workshop on Embedded Networked Sensors (EmNets 2006)*, 2006.

[20] J. B. B. Neto, F. M. M. Neto, D. G. Gomes, P. F. R. Neto, and R. M. C. Andrade. Integration of routing and time synchronization protocols for wireless sensor networks. In *EATIS ’08: Proceedings of the 2008 Euro American Conference on Telematics and Information Systems*, pages 1–4, New York, NY, USA, 2008. ACM.

[21] M. Pajic and R. Mangharam. Anti-jamming for embedded wireless networks. In *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, pages 301–312. IEEE Computer Society, 2009.

[22] J. Sallai, B. Kusy, A. Ledeczi, and P. Dutta. On the scalability of routing integrated time synchronization. *3rd European Workshop on Wireless Sensor Networks (EWSN 2006)*, February 2006.

[23] T. Schmid, P. K. Dutta, and M. B. Srivastava. High-resolution, low-power time synchronization an oxymoron no more. In *the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. ACM, 2010.

[24] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, et al. A macroscope in the redwoods. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, page 63. ACM, 2005.