On the Scalability of Routing Integrated Time Synchronization

János Sallai[1], Branislav Kusý[1], Ákos Lédeczi[1], and Prabal Dutta[2]

[1] Institute for Software Integrated Systems, Vanderbilt University,
2015 Terrace Place, Nashville, TN 37203, USA
{sallai,kusy,akos}@isis.vanderbilt.edu

[2] Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
prabal@cs.berkeley.edu

Abstract. Reactive time synchronization is becoming increasingly popular in the realm of wireless sensor networks. Unlike proactive protocols, traditionally implemented as a standalone middleware service that provides a virtual global time to the application layer, reactive techniques establish a common reference time base *post facto*, i.e. after an event of interest has occurred. In this paper, we present the formal error analysis of a representative reactive technique, the Routing Integrated Time Synchronization protocol (RITS). We show that in the general case, the presence of clock skews cause RITS to scale poorly with the size of the network. Then we identify a special class of sensor network applications that are resilient to this scalability limit. For applications outside this class, we propose an in-network skew compensation strategy that makes RITS scale well with both network size and node density. We provide experimental results using a 45-node network of Berkeley MICA2 motes.

1 Introduction

In a large class of sensor network (*sensornet*) applications, such as environmental monitoring [1], [2], target tracking [3], or countersniper systems [4], [5], sensor nodes are deployed in the environment to detect certain physical phenomena, or *events*. Typically, the sensed data is tagged with the location of the sensor node and the time of event detection. The location and time of event allow the sensornet to combine data from multiple sensors into high level information, that is, to perform *data fusion*, independently from the time when the data is received at a data fusion node. However, the data fusion can be only achieved if the time tags of events have a common time base across multiple sensors, or in other words, the sensors are *time-synchronized*.

The most common way to achieve time synchronization (*timesync*) is to use one of the many *proactive* timesync protocols [6], [7]. The term *proactive* is used because these protocols establish a virtual global time base in advance, namely,

before the sensornet application starts registering events from the environment. Commonly, proactive protocols use periodic message broadcasting to compensate for different sources of error (e.g. clock drifts, clock frequency noise, and clock glitches). The need for periodic message exchange, however, conflicts with the power constraints and lifetime requirements of sensornet applications.

The observation that the global virtual time of an event is *not* used at the node registering the event, but only at the data fusion node, together with the fact that proactive protocols suffer from severe messaging overhead, lead to the development of power-aware *reactive* timesync protocols [8], [7], [9]. The general idea of reactive approaches is *not* to synchronize the local clocks, but instead to timestamp the events using unsynchronized local clocks. Synchronization takes place *after* the event had been detected; henceforth, this approach is often called *post-facto* synchronization.

We concentrate on the analysis of the Routing Integrated Time Synchronization protocol (RITS) [8] that integrates post-facto timesync into a routing service. RITS, as well as reactive techniques in general, is superior to many proactive timesync protocols with respect to communication overhead. However, by decreasing the number of synchronization messages, we trade precision for power saving. While with a proactive protocol a node can frequently update its knowledge of the virtual global time, RITS is limited to using routing messages for synchronization.

In this paper, we show that the timesync error of RITS can significantly grow in the presence of clock skews and communication delays in message routing. We provide a formal analysis to explain the effects of error in such cases. The analysis of the components of RITS error shows that RITS is well-suited, without any enhancements, for applications where the sensor fusion algorithm works on time difference of arrival (TDOA) of events that are collocated in space and time. For the general case, we propose an in-network skew compensation strategy that can be adopted to improve the timesync error of RITS in particular, and reactive timesync protocols in general.

Based on the observation that information on clock skews of neighboring nodes is implicitly present in the timestamped messages they exchange, nodes can maintain a neighbor table storing skew information without any communication overhead. RITS converts the event timestamps from the local time of the sender node to the local time of the receiver node as the message is being passed from hop to hop along the routing path. We propose that the conversion of the timestamp includes compensation for the clock skew between the sender and the receiver at every hop.

It is imperative that sensornet applications be scalable not only with network size, but also with node density. In a dense network, however, it is not possible to store skew information for all neighbors. Our skew compensation approach addresses this requirement with a space efficient skew table maintenance strategy that operates with predefined table size. We store skew information only for a small selected subset of the neighbors. Unknown skews are estimated based on the locally available skew information.

Our experimental results, acquired from a 10-hop network of 45 MICA2 motes with artificially introduced routing delays, show that employing skew compensation reduces the timesync error of RITS from $29\mu s$ to $5.3\mu s$ on average.

We organize the paper as follows: Section 2 provides a detailed description of the leading reactive timesync protocols found in the literature. We formally analyze the timesync errors of RITS and discuss the implications on the general applicability of RITS in Section 4. To support sensornet applications in the general case, Section 5 presents an in-network skew compensation technique that improves the scalability of RITS with network size, communication delays, and node density. Finally, we offer our conclusions in Section 6.

2 Reactive Time Synchronization Protocols

Traditional *proactive* timesync protocols require the clocks of sensor nodes to be synchronized before an event happens. Because the clock rates of the nodes drift and vary in random and unpredictable ways, depending on the required timesync accuracy of sensornet applications, a non-trivial amount of system resources needs to be spent to keep the clock rate information accurate and actual. *Post-facto* (or *reactive*) timesync protocols propose to start the synchronization process after the event is detected to avoid performing timesync when it is not needed. This way nodes can be kept in a low-power sleep mode, conserving energy during periods of inactivity.

Post-facto synchronization was first suggested in [10] and later extended in [11]. The authors propose two forms of post-facto synchronization: *single-pulse synchronization* which requires advance calibration to be accurate, but reconstructs the global timescale quickly, and *post-facto synchronization with RBS* which takes longer to converge but does not require any a priori knowledge. The approach described in [9] transforms timestamps exchanged between nodes to the local time of the receiver, rather than adjusting the clocks to the global time base. The low message overhead of this method renders the protocol suitable for sensornets. Finally, the approach advocated in [8] claims to provide accurate and instantaneous timesync using no extra radio messages and requiring no a priori information. We delve into the details of these synchronization schemes in the remainder of this section.

Single-Pulse Synchronization. Single-pulse synchronization [11] requires a third party node, *a beacon*, to broadcast a synchronization pulse right after an event of interest was detected in the network. Nodes that receive the pulse use it as an instantaneous time reference and normalize their timestamps of the event detection to the synchronization pulse. This scheme works well for short distances (i.e. within the broadcast range of a single node) provided the stimulus timestamps are recorded close in time to the synchronization pulse. The three main error sources of this scheme were characterized as the receiver clock skew, variable delays in receivers, and propagation delay of the synchronization pulse. The clock skew error is the most significant source of error, therefore single-pulse

scheme works the best, if a priori calibration of clock frequencies is performed and clock skew estimates are used to correct the stimulus timestamps.

Post-facto Synchronization with Reference Broadcast. The second scheme proposed in [7] resolves the drawbacks of single-pulse synchronization scheme: it achieves timesync over large distances, and synchronizes nodes that have no mutually shared information. After a stimulus event is detected in the network, an algorithm estimating the clock skews between the nodes is executed and the resulting clock skew estimates are used to correct the stimulus event timestamps in the past. This scheme resolves the problems of the single-pulse, but also brings in some disadvantages: the RBS estimator needs multiple synchronization pulses to obtain clock skew estimates, so the timesync is not achieved instantaneously. If a long time passes between the stimulus event and skew estimator conversion, the clock skew estimates may significantly differ from the clock skews at the event detection introducing additional errors.

Time Synchronization with Timestamp Conversion. This protocol [9] proposes not to synchronize the local clocks of the devices, but instead to generate timestamps using unsynchronized local clocks. When the locally generated timestamps are passed from a node to node in the network, they are converted to the local time of the receiving device. Due to the limited precision of the timestamp conversion used, the algorithm uses time intervals as a lower and upper bound for the exact value. Comparison of timestamps relies on a special interval arithmetic, hence there are cases when the temporal ordering of timestamps cannot be determined. One distinctive feature of this approach is that the timestamp transformation has the following correctness property: the partial ordering of event timestamps in the local time of a give node reported by the algorithm is a subset of the total ordering of the times of the event in real time. This approach explicitly targets communication of timestamps over long distances, making it particularly suitable for multi-hop ad hoc networks.

3 The Routing Integrated Time Synchronization Protocol

RITS [8] is a reactive timesync protocol, which can be used to obtain times of event detections at multiple observers in the local time of the sink node(s). We provide a more detailed description of the protocol later when formally analyze the timesync errors it introduces.

From the application's point of view, RITS is an extension of the routing service with a well-defined interface. The interface defines commands to send and timestamp a data packet, a callback function to signal the reception of a packet, and a command to query the timestamp of a received packet. On detecting an event, the application on the sensor node generates a data packet containing the event information, and timestamps it with the value of the local time of detection. It forwards the packet with the timestamp to the routing service, which delivers it to the sink. RITS places no assumptions on the network topology or routing algorithm beyond those that are required by the application. Rather than performing explicit timesync after the event of interest is detected, RITS

performs inter-node time translation along the routing path from an observer node to the sink: as the data packet travels from node to node in the network, RITS converts the corresponding timestamp from the local time of the sender to that of the recipient. When the packet arrives at the sink, the routing service signals an event to the application layer that a packet has been received. The application can then query the routing layer for the timestamp of the received packet, which is returned in the local time of the sink.

The prototype implementation of RITS builds on the Directed Flood Routing Framework (DFRF) [12]. DFRF is a generic routing framework that supports rapid prototyping and implementation of large class of application specific routing protocols that are based on directed flooding. Integrating reactive timesync with the routing service has several benefits over a standalone timesync service:

- **Coupling of event data and event timestamps.** There is a tight logical coupling between event information and the corresponding timestamps. RITS retains this coupling: in a data packet, event data and timestamps are physically collocated. RITS thus implements *implicit* timesync, that is, the flow of time information is embedded in the flow of data. There are no pure timesync messages, hence RITS has virtually no communication overhead.
- **Network-transparent event timestamps.** As data packets propagate in the network, RITS converts the corresponding time stamp hop by hop to the local time of the recipient node. As a result, all data packets received by a given node contain event timestamps in the recipient node's local time, independently from where in the network the events originated.
- **Packet aggregation.** Packet aggregation helps decrease the number of message transmissions. In fact, not only does the number of radio messages decrease, but also the overall payload size. This is because in an aggregated radio message, n data packets (containing event information and event timestamp) share only one transmit timestamp.
- **Packet filtering.** Through packet filtering support, it is possible to discard outdated messages at intermediate nodes *enroute* to the destination, thus decreasing the message load.
- **Orthogonality to the routing policy.** DFRF allows for the customization of routing behavior via routing policies. RITS is orthogonal to the policies, that is, the same time conversion is used with different routing behaviors.

4 Analysis of the Error of RITS

The RITS protocol claims to achieve highly accurate instantaneous post-facto timesync without using extra radio messages [8]. RITS provides these properties only for a relatively small subset of sensornet applications, for which a particular set of assumptions is fulfilled. We formally express the error of RITS and derive the set of properties that RITS requires from sensornet applications.

As noted in both [11] and [8], reactive protocols are susceptible to multiple sources of error. The two most egregious ones are the error caused by different

clock rates of the nodes on the routing path, and the error in timestamping the radio message arrival.

We use the following notation: we have a set of N nodes that can be receivers r_i and/or senders s_i, $i \in \{1, \ldots, N\}$. Each node has its own local clock that measures the local time t_i. We denote a fictious universal time with u. The offset of the local time from the universal time can change over time because the clock rate of a node can differ from the rate of universal time, we express the relation of the local and universal time as

$$t_i = \alpha_i u + \beta_i. \tag{1}$$

The clock skews α_i are assumed to be constant in the time interval a packet spends at node i. This assumption is justifiable for a reasonably fast routing service, nevertheless the crystal clock rates, though slowly, do change in the real hardware. Furthermore, we assume that the clock skews α_i are independent random variables from a symmetric distribution with mean one (that is, the *universal time rate*). We impose no assumptions on the initial clock offsets β_i.

We express the synchronization mechanism of RITS as follows: receiver r_k synchronizes with the sender s_j by receiving a synchronization message m_i. We denote the sender timestamp of message transmission by y^s_{ij} and the receiver timestamp of message arrival by y^r_{ik}. Both y^s_{ij} and y^r_{ik} are known to the receiver. If e^s_{ij} and e^r_{ik} are timestamping errors of sender and receiver, respectively, and u_i is the universal time when the message was transmitted, then

$$y^s_{ij} = \alpha_j u_i + \beta_j + e^s_{ij} \tag{2}$$

$$y^r_{ik} = \alpha_k u_i + \beta_k + e^r_{ik}. \tag{3}$$

Similarly, if the i-th node records the local time of an event E, we denote this timestamp as y_{Ei}.

According to [6] it is assumed that both e^r_{ik}, and e^s_{ij} are independent identically distributed random variables with zero mean. Since low-power transceivers have limited communication range, we can further assume that the propagation delay between the sender and receiver is negligible, therefore the universal time of sending and receiving message m_i are the same (i.e. u_i).

If the receiver (r_k) wishes to transform a time of stimulus event E from the sender's (s_j) timeline to its own timeline, provided a radio message m_i has been sent, the receiver performs the following calculation:

$$y_{Ek} = y_{Ej} + y^r_{ik} - y^s_{ij}. \tag{4}$$

It is important to note that the timestamp conversion of RITS does not consider the clock skews. Henceforth, we expect that skew related errors will accumulate.

4.1 Error Along a Routing Path

Now let us apply the transformation iteratively as a message is being passed along routing path to get the following general result. Timestamps converted to

the local times of the second, the third node and the n-th node ($y^r_{01} = y_{E1}$) are:

$$\begin{aligned} y_{E2} &= y_{E1} + y^r_{12} - y^s_{11} \\ y_{E3} &= y^r_{23} - (y^s_{22} - y^r_{12}) - (y^s_{11} - y_{E1}) \\ y_{En} &= y_{E(n-1)} + y^r_{(n-1)n} - y^s_{(n-1)(n-1)} = y^r_{(n-1)n} - \sum_{i=1}^{n-1}(y^s_{ii} - y^r_{(i-1)i}). \end{aligned}$$

We denote the timestamping error introduced by the i-th node with e_i and define it as $e_1 = e^s_{11}$ and $e_i = e^s_{ii} - e^r_{(i-1)i}$ for $i > 1$ and use e_i along with Equations 2 and 3 to further rewrite y_{En}:

$$y_{En} = y^r_{(n-1)n} - \sum_{i=1}^{n-1} \alpha_i(u_i - u_{i-1}) - \sum_{i=1}^{n-1} e_i.$$

Furthermore, for the sake of simplicity, let us assume that for all i, $u_i - u_{i-1}$ is constant, that is, the message spends equal amount of time at each node along the routing path. Let us denote this constant with τ. This way we can separate the first summation into skew-independent and skew-dependent components:

$$y_{En} = y^r_{(n-1)n} - \tau(n-1) - \tau \sum_{i=1}^{n-1}(\alpha_i - 1) - \sum_{i=1}^{n-1} e_i, \tag{5}$$

where the first term is the time when the message arrives at the last node, the second term is the age of the packet, the third and fourth terms are errors introduced by the clock skews and the message timestamping, respectively.

4.2 RITS and TDOA Measurements

In an important class of monitoring applications, sensor fusion works with time differences of arrival (TDOA) of events. Let us assume that the event E was detected at time u_E by two nodes r_1 and r'_1, and the two time tags arrived to the data fusion node along two different paths P and P', such that $P = r_1, \ldots, r_n$ and $P' = r'_1, \ldots, r'_m$. We further know that the final node of both P and P' is the same (the data fusion node), so $\alpha_n = \alpha'_m = \alpha_{df}$. Without loss of generality we can assume that $n < m$. Consequently, we express the error that RITS introduces to the TDOA data when routing the timestamps to the data fusion node:

$$y'_{Em} - y_{En} = \tau \sum_{i=1}^{n-1}(\alpha_i - \alpha'_i) + \tau \sum_{i=n}^{m-1}(\alpha_{df} - \alpha'_i) + \sum_{i=1}^{n-1} e_i - \sum_{i=1}^{m-1} e'_i. \tag{6}$$

Due to the assumptions on the distribution of the skews and the message timestamping errors, the expected values of the first, third and fourth terms are zero. Interestingly, the expected value of the second term is $\tau(m-n-1)\alpha_{df}$. This means that the clock skew of the data fusion node introduces an error

proportional to the clock skew of the data fusion node and the difference of delivery times of the messages.

The variance of the two skew related terms sum up to $\tau[(n-1)+(m-1)]V(\alpha)$, meaning that the variance is proportional to the time the messages spend at a given hop and to the sum of the lengths of the paths. The message timestamping related error has a variance of $[(n-1)+(m-1)]V(e_i)$, which grows with the sum of the path lengths.

An important special case is when the two paths overlap. Without loss of generality, we assume that $P' = r_1, \ldots, r_j, \ldots, r'_{j+1}, \ldots, r'_m$. Using partially overlapping paths, the TDOA calculated by RITS changes as follows:

$$y'_{Em} - y_{En} = \tau \sum_{i=j+1}^{n-1} (\alpha_i - \alpha'_i) + \tau \sum_{i=n}^{m-1} (\alpha_{df} - \alpha'_i) + \sum_{i=1}^{n-1} e_i - \sum_{i=1}^{m-1} e'_i. \quad (7)$$

Since the skew related errors introduced by the nodes that are on both paths cancel out, the variance of the skew related error decreases to $\tau[(n-1)+(m-1)-j)]V(\alpha)$. That is, the variance of the skew related errors is proportional to the time the packet spend at the nodes and to the length of the disjoint regions of the routing paths. Another factor is that the event times need to be close to each other. Otherwise, the clock skew of the sender node introduces a large error. In particular, the events should be kept on a node for as short period of time as possible.

Platform	**Timestamping error**	**Clock skew error**
Mica2 external crystal (32 kHz, CC1000 radio)	30.5 μs (= 1 clock tick)	50ppm typically 30.5 μs per second
Mica2 internal oscillator (7MHz, CC1000 radio)	1.4 μs	50ppm typically $\leq$ 20μs per second
Telos internal oscillator (8MHz, CC2420 radio)	0.125 μs	50ppm

Table 1. Survey of timesync errors expected from timestamping and clock skews for common sensornet platforms [13], [14].

4.3 Implications of Theoretical Results

We provide typical expected timestamping errors and clock skews for the most common sensornet platforms to match the formal results of the previous section to real world hardware in Table 1. We further concentrate on the applicability of the RITS protocol and show an experimental test case, where the timesync error of RITS does not scale well with the number of hops. Consequently, we provide a set of properties that RITS requires from the sensornet applications. Finally, we use the error analysis results and suggest two improvements of RITS.

Applicability of RITS. The design of RITS was heavily influenced by the requirements of a specific sensornet application, acoustic event localization [4]:

– only time differences of arrivals are required to localize an acoustic event,

– fast routing to the sink is required, because the event source is mobile,
– preservation of temporal ordering of events is not a requirement, and
– detected events are close to each other in space and time.

These properties are important because they place bounds on the terms in Formula 6 that contribute to the error of RITS. Maximum RITS error of $80\mu s$ and the average error of $8\mu s$ were reported in [8] using Mica2 platform and 7MHz internal clock in a 10 hop network. However, only a small class of applications can achieve similar results, the main problem being the error introduced by the (uncompensated) clock skews. Table 1 shows that this error can become significant even over moderate time intervals.

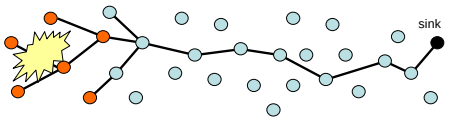

Fig. 1. Large scale network detects an event, the timestamps are then routed along a spanning tree to the sink node.

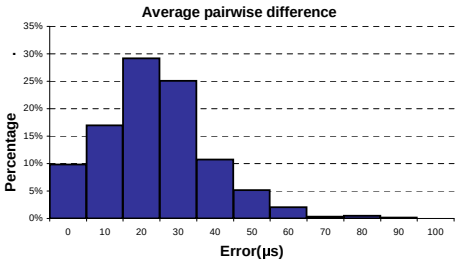

Fig. 2. The histogram of average errors for the experiment with simulated transmission delays of 5 seconds at each node.

We experimentally verified the poor scalability of RITS if the routing time to the sink node increases. We carried out an experiment similar to the one described in [8]: we used 45 Mica2 motes arranged in a grid forming a 10-hop network. Events that triggered nodes within a certain radius were periodically simulated at random points in the network. Each event was simultaneously detected at all triggered nodes, and the timestamps of these detections were sent to the data fusion node (sink), as shown in Figure 1. We introduced an artificial delay of five seconds between receiving and forwarding the message, thus inflating the time intervals it takes to route the event detection times. The maximum and average synchronization errors are computed as the maximum and average pairwise difference of all timestamps received by the sink that correspond to the same event detection. Compared to the non-delayed case [8], the measured maximum and average synchronization errors across 700 simulated events grew significantly from $80\mu s$ to $265\mu s$, and from $8\mu s$ to $29\mu s$, respectively. The histogram of average errors can be seen in Figure 2.

Mitigating the Error of RITS: Routing Strategies. Formula 6 shows that if we fix a routing path P, then the variance of the term $\tau \sum(\alpha_i - 1)$ grows with the increased routing time to the sink. This causes a large timesync error along the path P. The important observation is that this error is consistent, i.e. has a relatively small variance, as it is caused by the timestamping errors alone, since the skew related errors cancel out.

We verified this experimentally by deploying 50 Mica2 nodes logically arranged on a line, forming a 50-hop network. The first node, the coordinator, broadcasts a RITS packet with its current time. Other nodes retransmit the packet upon receiving it, until the packet reaches the last node. The coordinator overhears all retransmissions, uses RITS to convert the timestamp in the packet to its local time and records the error of the timestamp after each hop. This allowed us to study the error of RITS across multiple hops. Figure 3 shows the accumulated error after each hop, averaged over multiple rounds. To make the clock skew errors prevail over timestamping errors we introduced a delayed strategy for the RITS packet retransmission: a node waits 5 seconds after receiving the packet and only then retransmits it.

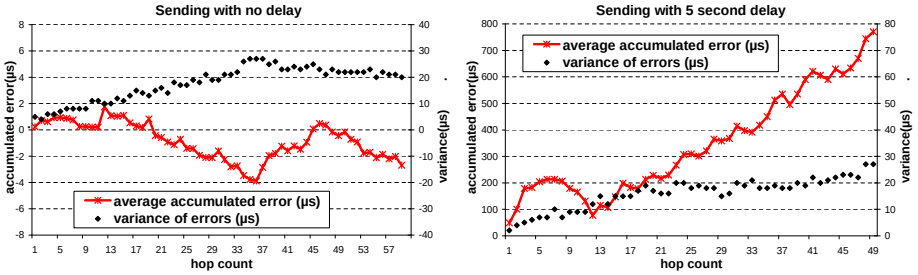

Fig. 3. 50 nodes arranged on a line experiment: errors of the RITS calculated time compared to a single-node time for the no-delay and 5 seconds delay strategies are shown. The accumulated errors are averaged over multiple runs and we plot the variance of these errors.

The promising fact is that the variance of the accumulated error is the same in both experiments, which means that the motes introduce significant, but consistent error. Consequently, we specify the applications requirements and propose improvements to the RITS protocol to make it scale for large networks:

- the original RITS protocol did not discuss the implications of using different routing strategies. We observe the advantage of overlapping routes, and suggest using spanning tree routing which ensures this property. In contrast, gossip or epidemic protocols which can result in dynamic routing paths, will not support scalability of RITS.
- the sink node introduces large error if the time between receiving two different events is long. This error is caused by the clock skew error of the sink and can be mitigated by synchronizing the sink with a high precision external clock source.

With these improvements, RITS scales well with the number of hops and communication delays, provided an application has the following properties:

- applications need to be TDOA based; we know that a routing path introduces large error which is unknown to the sink node, therefore, relating this time to any global time scale is not possible,

– the stimulus event detections must be located within a small neighborhood, as shown in Figure 1, so that the significant portions of the spanning tree paths overlap.

5 Skew Compensation

The main features of RITS were that it does not require any a priori information, does not need to know or maintain the skews of the nodes, and uses no additional timesync messages to achieve synchronization. The improvements to RITS discussed in the previous paragraph did not need to sacrifice any of these features. Supporting a general class of applications, however, drives us to drop one of these properties. We show that it is possible to estimate the clock skews without using additional timesync messages, provided that there exists a lower bound on the frequency of the stimulus events in the deployment area.

5.1 RITS with Clock Skew Compensation

Compensating for the skews between the clocks of the nodes along the routing path can significantly decrease the variance of the timesync error. Recall that RITS converts the event timestamps from the local time of the sender node to that of the receiver node as the message is being passed from hop to hop. Without skew compensation, this conversion is achieved by adding the offset of the clocks of the sender and the receiver nodes to the event timestamp in the local time of the sender to yield the event timestamp in the local time of the receiver.

When skew compensation is employed, the conversion is more involved. We do not assume that a node knows its clock skew from the nominal clock rate (referred to as the *absolute skew*), however, it is assumed that it knows the skews of their neighbors relative to its local clock rate (referred to as the *relative skew*). This means that there is no global clock rate to which the elapsed time at each hop could be converted.

The absolute skew of node i is defined as its skew relative to the nominal clock rate f_{nom}, that is, $\alpha_i = \frac{f_i}{f_{nom}}$. Relative skew of node i with respect to node j is defined as $\alpha_{i,j} = \frac{f_i}{f_j}$.

5.2 The Approach

The proposed skew compensation approach is the following. When receiving a packet – which includes event description and event timestamp in the sender's local time – the receiver node calculates the *age of the packet* in the sender's time:

$$age^s = y_m^s - y_E^s,$$

where y_m^s is the transmit timestamp of the message and y_E^s is the event timestamp, both in the sender's local time. Because a clock skew is present, the age of the

packet needs to be converted from the sender's clock rate to the receiver's clock rate, which is achieved simply by dividing it with the relative skew.

$$age^r = \frac{age^s}{\alpha_{s,r}}$$

Subtracting the converted packet age from the receive timestamp of the message yields the event timestamp in the receiver's local time.

$$y_E^r = y_m^r - age^r$$

Expressing the conversion in one formula gives

$$y_E^r = y_m^r - \frac{y_m^s - y_E^s}{\alpha_{s,r}}.$$

Eventually, when the packet reaches the sink node, the event timestamp is converted to the sink node's local time. In contrast with RITS without skew compensation, the conversion takes into account differences of *both offsets and skews* between the sink node and nodes registering the events.

5.3 Measuring the Relative Skew

Relative skew of a neighbor w.r.t. a given node is computed as the fraction of the number of ticks of the neighbor's clock and the number of ticks of the local clock during a reference time interval.

While determining the clock offset of neighboring nodes requires only one common reference point in time, acquiring their relative skew necessitates having two of them. The estimation of relative skews is based on a neighbor skew table, which contains records with the transmit and receive timestamps of the most recent message from the neighbor, as well as the most up-to-date relative skew to the neighbor, if known.

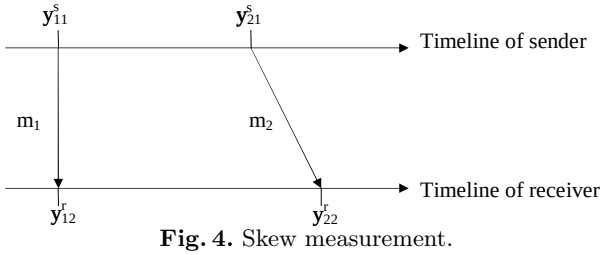

Fig. 4. Skew measurement.

Maintenance of the skew table is carried out as follows. When a message is received, we locate the sender's record in the skew table. The record contains the transmit and receive timestamps of the previous message from the sender. The difference of the actual and the previous transmit timestamps (in Figure 4 denoted by y_{21}^s and y_{11}^s, respectively) gives the time elapsed between the two messages in the sender's clock. Similarly, the difference of the actual and the previous receive timestamps, $y_{22}^r - y_{12}^r$, gives the time elapsed between the two

messages in the receiver's clock. The relative skew is the fraction of the two differences, which is exponentially averaged with the previously calculated skew value, and stored in the skew table.

Our approach employs an implicit skew measurement technique. All radio messages are timestamped by the sender and the receiver, regardless of message content. Since the relative skew information is implicitly carried in the message timestamps, measuring the skews requires no dedicated communication. This solution, however, has its caveat: since the clock frequencies of the devices are not stable, relative skews become outdated if nodes communicate rarely. The problem can be solved by periodically generating dummy messages, though RITS leaves this to the application layer. A method to find the optimal beaconing period has already been proposed by Ganeriwal et al. in [15] and [16].

5.4 The Challenge of Memory Constraints

Networked sensor nodes are severely constrained devices, where RAM is a precious resource. It is not unusual that the operating system, the middleware services (multi-hop routing, timesync, etc.) and the application layer have to share no more than 4kB of RAM. The neighbor table that augments skew measurements and stores the relative skew values contains records in the following structure.

```
typedef struct {
   uint16_t nodeID;          // ID of the neighbor
   float    skew;            // relative skew w.r.t the neighbor
   uint32_t lastTxTimeStamp; // Tx timestamp of last recvd message
   uint32_t lastRxTimeStamp; // Rx timestamp of last recvd message
} neighborRec;
```

The size of this record is 14 bytes, which might seem negligibly small, however, sacrificing a few hundred bytes for a neighbor table of one of the many middleware components might not always be a viable option.

We face two conflicting constraints here: small memory footprint versus scalability with node density. If the size of the skew table is too small, skew compensation will fail in dense networks, whereas a large neighbor table is not affordable because of the memory constraints.

5.5 Maintaining a Bounded Skew Table

Clearly, the need for scalability with network density necessitates limiting the size of the skew table. Furthermore, we may want to control which neighbors' relative skews we store, and we need to decide how to compensate for the skews of those neighbors for which no skew information is available. Our approach is that a node stores the relative skews only for a subset of its neighbors. Using the stored values, the node estimates its relative skew to the rest of the neighbors.

The most important property of this strategy is that the absolute skew of the node itself does not influence which neighbors are stored in the skew table: this

decision is made purely on the observed relative skew values of the neighbors. This way the skew compensation will work well even if the node itself has a significant absolute skew.

5.6 Estimation of Unknown Skews

An appealing strategy is to keep the skew information of neighbors having the *worst* absolute skews in the skew table, and not storing the relative skew information of the remaining *well-behaved* neighbors. When a packet is received from a *bad* neighbor, which would normally introduce a considerable skew related timesync error, the stored value is used for skew compensation. This way the worst timesync errors will be compensated for. When receiving a packet from a *good* neighbor, which has only minor contribution to the timesync error, we compensate with an estimate of the mean of the relative skews of the neighbors.

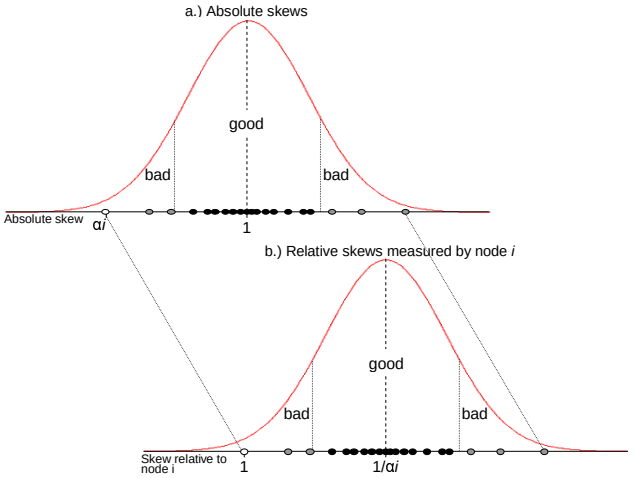

Fig. 5. Distribution of absolute and relative skews measured by an arbitrary node i. The white dot denotes node i, the grey and the black dots denote the *bad* and the *well-behaved* neighbors, respectively.

Since the absolute skews cannot be measured directly, the categorization of good and bad neighbors has to rely on the information carried in the relative skew measurements. The neighbor's relative skew values, as perceived by a node, are normally distributed with the same variance as the distribution of the absolute skews w.r.t. the nominal skew, but the values are centered around the reciprocal of the node's own absolute skew, not around 1. We can observe that the relative skews of the *good* neighbors fall close to the median of the measured values, while those of the *bad* neighbors are far from it.

If the bounded skew table is maintained such that the categorization of good and bad skews is based on their distance from the median of the measured skew values, the skew table will store the left and right tails of a random sample representing the relative skews of the neighbors (even if the clock rate of the local node significantly differs from the nominal rate). The values that are not

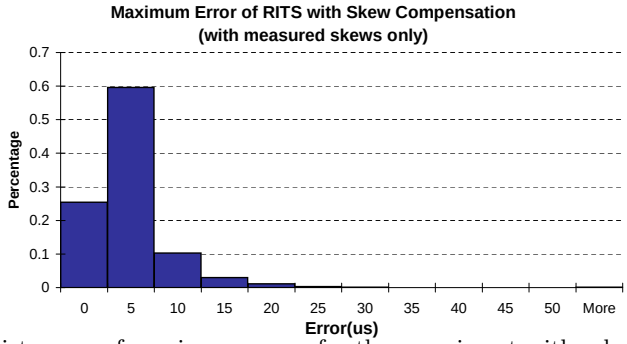

Fig. 6. The histogram of maximum errors for the experiment with a large skew table holding the skew values of all neighbors.

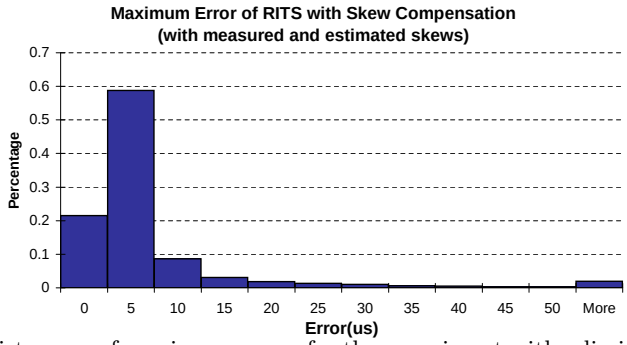

Fig. 7. The histogram of maximum errors for the experiment with a limited skew table holding the skew values of 6 neighbors.

stored must fall between the maximum skew of the left tail and the minimum skew of the right tail; from here, we estimate the unknown skews with the average of the two.

The corresponding table maintenance strategy is implemented as follows. The size of the skew table, denoted by n, is set to an even number. The skew records are sorted by the skew values. When the skew table is full, and a new skew measurement is completed, the new value is compared with the skews of the two records in the middle (at positions $\frac{n}{2}$ and $\frac{n}{2}+1$). If it is between the two values, it is discarded. If it is below (above) the two values, the record at position $\frac{n}{2}$ (at position $\frac{n}{2}+1$) is evicted, and the new measurement is inserted in the skew table. In a steady state, the two middle values give a lower and an upper bound on the skews of the neighbors that are not stored.

5.7 Experimental Results

We repeated the experiment described in Section 4.3 using RITS augmented with clock skew compensation. The 45 Mica2 motes were arranged in a grid forming a 10-hop network, using a sufficiently large neighbor table. As in the previous experiment, we introduced an artificial routing delay of five seconds at every hop, to allow skew related errors to manifest.

To test the performance of our skew compensation algorithm, we set the size of the skew table large enough to hold the skew information of all neighbors. As Figure 6 shows, employing in-network skew compensation drastically reduced the timesync error of RITS. Compared to the previous results (see Figure 2), the average synchronization errors decreased from $29\mu s$ to $2.8\mu s$. Not considering the bootup period of the skew compensation algorithm when the skew table is not populated, the maximum error decreased from $265\mu s$ to $44\mu s$.

In the next experiment, we limited the size of the skew table to hold only six records. As expected, the measured timesync errors increased compared to the fully compensated case. Although the maximum error we experienced was $258\mu s$, which is comparable to the non-compensated case, only 1% of the errors were above $100\mu s$. This can be attributed to the drastically small neighbor table. However, as Figure 7 shows, skew compensation with partial skew information is still a significant improvement over the non-compensated case, as the average synchronization error was only $5.3\mu s$.

6 Conclusions and Future Work

The Routing Integrated Time Synchronization protocol was specifically designed with a single application in mind [4]. Although it was successfully tested in a number of medium-scale deployments, our analysis found that its scalability with network size and communication delay limits its general applicability.

In this paper we investigated the reasons for the scalability problems and presented an error analysis of RITS. We showed that the variance in clock rates is responsible for the largest time synchronization errors. We found that the clock skew related errors do not manifest in a special class of sensornet applications, where only the time difference of arrival of the registered events are of concern, and where event detections are close to each other in space and time. We presented an in-network skew compensation technique that improves the scalability of RITS, making it suitable for a wide range of sensornet applications.

We further plan to investigate scalability limits of RITS augmented with skew compensation. Since the skew measurement errors and the skew estimation errors are expected to propagate in a multiplicative fashion, the precision of the skew measurements and a probabilistic upper bound on the estimation error have to be controlled. This can be achieved by setting a proper lower and upper limit on the time difference of reception of the two messages that are used to measure the skew and by choosing the size of the neighbor table large enough, such that the skews of the *good* neighbors are bounded by a relatively small interval. Finding the proper values of these constants demands further research.

7 Acknowledgments

The DARPA/IXO NEST program has partially supported the research described in this paper. We also wish to thank Miklós Maróti and anonymous reviewers for their valuable comments on our work.

References

1. Xu, N., Rangwala, S., Chintalapudi, K.K., Ganesan, D., Broad, A., Govindan, R., Estrin, D.: A wireless sensor network for structural monitoring. SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems (2004) 13–24
2. West, B.W., Flikkema, P.G., Sisk, T., Koch, G.W.: Wireless sensor networks for dense spatio-temporal monitoring of the environment: A case for integrated circuit, system, and network design. 2001 IEEE CAS Workshop on Wireless Communications and Networking (2001)
3. Wang, H., Elson, J., Girod, L., Estrin, D., Yao, K.: Target classification and localization in a habitat monitoring application. Proc of IEEE ICASSP (2003)
4. Simon, G., Maróti, M., Lédeczi, A., Balogh, G., Kusý, B., Nádas, A., Pap, G., Sallai, J., Frampton, K.: Sensor network-based countersniper system. SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems (2004) 1–12
5. Lédeczi, A., Nádas, A., Völgyesi, P., Balogh, G., Kusý, B., Sallai, J., Pap, G., Dóra, S., Molnár, K., Maróti, M., Simon, G.: Countersniper system for urban warfare. ACM Transactions on Sensor Networks **1** (2005) 153–177
6. Maróti, M., Kusý, B., Simon, G., Lédeczi, A.: The flooding time synchronization protocol. SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems (2004) 39–49
7. Elson, J., Girod, L., Estrin, D.: Fine-grained network time synchronization using reference broadcasts. SIGOPS Oper. Syst. Rev. **36** (2002) 147–163
8. Kusy, B., Dutta, P., Levis, P., Maroti, M., Ledeczi, A., Culler, D.: Elapsed time on arrival: A simple and versatile primitive for canonical time synchronization services. International Journal of Ad Hoc and Ubiquitous Computing **2** (2006)
9. Römer, K.: Time synchronization in ad hoc networks. Proceedings of MobiHoc 2001 (2001)
10. Elson, J., Estrin, D.: Time synchronization for wireless sensor networks. Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS-01) (2001)
11. Elson, J.: Time synchronization in wireless sensor networks. Ph.D. Thesis (2003)
12. Maróti, M.: Directed flood-routing framework for wireless sensor networks. Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware (2004) 99–114
13. Hill, J., Culler, D.: Mica: A wireless platform for deeply embedded networks. IEEE Micro **22** (2002) 12–24
14. Polastre, J., Szewczyk, R., Culler, D.: Telos: Enabling ultra-low power wireless research. Proceedings of the 4th Int. Conf. on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS) (2005)
15. Ganeriwal, S., Ganesan, D., Hansen, M., Srivastava, M.B., Estrin, D.: Rate-adaptive time synchronization for long lived sensor networks. Proceedings of the ACM international conference on Measurement and modeling of computer systems. (SIGMETRICS 2005) (Short Paper) (2005)
16. Ganeriwal, S., Ganesan, D., Sim, H., Tsiatsis, V., Srivastava, M.B.: Estimating clock uncertainty for efficient duty cycling in sensor networks. SenSys '05: Third ACM Conference on Embedded Networked Sensor Systems (2005)