Incorporating Energy Heterogeneity into Sensor Network Time Synchronization

Zhenjiangi Li, *Member, IEEE*, Wenwei Chen, *Student Member, IEEE*, Mo Li, *Member, IEEE*, and Jingsheng Lei

Abstract—Time synchronization is one of the most fundamental services for wireless sensor networks. Prior studies have investigated the clock stability due to environmental dynamics. In this paper, we demonstrate by experiment that in spite of the surrounding environment, time synchronization is unavoidably impacted by in-network energy heterogeneity, which may incur up to 30-40 ppm clock uncertainty. We mathematically analyze the root cause of such clock uncertainty and propose a protocol called EATS. Sensor nodes with EATS can intelligently select the best synchronization parents that minimize the negative impact of the energy heterogeneity. The selection is robust to multiple impacting factors in the network and provides fine-grained synchronization accuracy. In addition, nodes can make use of local energy information and further calibrate the clocks. In light of this, the logic time maintained among different nodes is more consistent and the synchronization can be performed with a longer re-synchronization interval and less energy consumption. We implement EATS with TelosB motes and evaluate the effectiveness and efficiency of our design through extensive experiments and simulations.

Index Terms—WSN, time synchronization, energy heterogeneity

1 Introduction

Time synchronization is one of the most fundamental services in wireless sensor networks (WSNs). A common notion of time maintained in the network enables a variety of important WSN-based applications, e.g., data collection [1], skeleton extraction [2], MAC-layer time-slot alignment [3], network control and monitoring [4], [5], etc. Limited by the manufactural technique, the frequency of oscillators is not stable. For example, the most widely used CMOS crystal oscillators generally suffer from fluctuations ranging from 5 to 100 ppm (parts per million) [6]. Therefore, retaining a consistent and stable time estimate is crucial in WSNs.

A number of solutions have been proposed for synchronizing the entire network to a common timing reference (e.g., the clock time of the sink node). Representative examples include RBS [7], TPSN [8], FTSP [9] and so on. In addition to those protocol-level solutions, recent studies observe that the stability of oscillators is also determined by environmental factors, e.g., temperature, humidity, and pressure. Countermeasures are introduced to further improve the synchronization accuracy against environmental dynamics [10], [11], [12]. In this paper, we find that in spite of the surrounding environment, time synchronization is still *unavoidably* impacted by a set of in-network factors that may incur up to 30-40 ppm clock uncertainty, while they draw little attention in prior literatures.

To facilitate the system management, sensor nodes are usually powered by batteries. Different nodes may have different initial energy budgets. On the other hand, according to recent measurement studies [13], traffic load usually distributes unevenly and varies in the network, leading to heterogenous energy consumption rates. As the traffic load directly affects the energy draining rate of each node, the available power budgets (i.e., the supply voltages) tend to become *heterogenous* as time elapses. Such an energy heterogeneity phenomenon significantly affects the synchronization performance. According to our measurements, the supply voltage can incur 30-40 ppm uncertainty to clocks, which must be carefully eliminated. This problem may get even worse for energy harvesting sensor nodes [14], [15], [16], which can result in a more dynamic energy distribution in the network.

To cope with the energy heterogeneity issue towards the synchronization performance, following challenges must be carefully addressed. Sensor nodes in today's time synchronization protocols liberally integrate timing information from neighboring nodes into their own time maintenance. Instead of coupling such integrating with (data transmission) routing, a node should selects a synchronization parent that minimizes the impact of the energy heterogeneity. In addition, for those low-duty-cycle sensor networks where sensor node sleep may lead to long re-synchronization interval (e.g., 10-30 minutes), the energy distribution may dramatically change due to uneven traffic burden or energy harvesting. In any re-synchronization interval, clocks of sensor nodes advance independently and the offset accumulates. To achieve consistent network clocks and prolonged duty-cycle interval, the clock of each individual node should be carefully compensated according to its energy

- *Z. Li, W. Chen and M. Li are with the School of Computer Engineering, Nanyang Technological University, Singapore, 639798. E-mail: zhenjianglee@gmail.com, {chen0746, limo}@ntu.edu.sg.*
- *J. Lei is with the School of Computer and Information Engineering, Shanghai University of Electric Power, China, 200090. E-mail: jshlei@shiep.edu.cn.*

Manuscript received 3 May 2013; revised 16 Oct. 2013; accepted 5 Dec. 2013. Date of publication 23 Feb. 2014; date of current version 5 Dec. 2014. Recommended by J. Chen.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2014.2307890

drainage. To the best of our knowledge, an instrument to solving such issues is still lacking.

Although prior studies [10], [11], [12] suggest to select the robust synchronization neighbor for temperature changes as well, those proposed solutions do not directly concern the energy dynamics, which are strongly associated with internal network operations and thus quite different from external environment dynamics. We thoroughly study the time synchronization problem caused by the energy heterogeneity in this paper. The contributions of this paper are as follows. We unveil the clock uncertainty with respect to its supply voltage and observe that such uncertainty can significantly degrade the accuracy of time synchronization. Aiming at solving the problem in general, we mathematically analyze the root cause of the clock uncertainty and propose a protocol called EATS in this paper. EATS contains two major modules: *synchronization parent selection* and *skew compensation*. In the former module, we propose to adopt the linear regression technique to select synchronization parents that minimize the impact of the energy heterogeneity. In the latter module, each sensor node utilizes a prediction technique to compensate/calibrate its own clock. Based on the voltage-skew table introduced in EATS, the offset accumulation rate can be dramatically reduced, and as a result re-synchronization intervals are largely prolonged to save energy. We implement EATS on TelosB motes and evaluate its effectiveness and efficiency through extensive experiments and simulations.

The rest of this paper is organized as follows: Related works are reviewed in Section 2. In Section 3, we introduce paper's preliminary information and unveil the clock uncertainty w.r.t. the energy heterogeneity by experiments. In Section 4, we analyze the root cause of such clock uncertainty, propose EATS, and examine its performance on the TelosB platform. We further evaluate the network-wide performance of EATS through trace-driven simulations in Section 5. Finally, we conclude in Section 6.

2 Related Work

Great efforts have been made for achieving common time notion across the networking systems. RBS [7] eliminated the sender-side delay for synchronization. TPSN [8] further canceled out the propagation delay. By using the MAC-layer stamping technique, FTSP [9] largely increases the synchronization accuracy. However, the authors in [17] find that errors among different clocks exponentially increase with the network diameter. [18] proposes a fast flooding scheme and [19] studies the synchronization accuracy in low-duty-cycle sensor networks. [20] estimates clock uncertainty for duty-cycled sensor networks. The authors propose ATS and MTS in [21] and [22] respectively, which compensate both clock skews and offset, via message exchanges among neighboring nodes. In fact, EATS is compatible with above designs. EATS can be integrated with them to further improve their synchronization performance. On the other hand, some other works also focus on addressing the clock uncertainty and reducing the time synchronization cost. ACES [23] suggested to track skew by using Kalman filter and ODS [6] introduces the on-demand accuracy synchronization. [10] explores the temperature compensated scheme to mitigate the clock skew. In [19] and [24], authors introduce to maintain a common time by using two clocks on each individual node with different accuracies. To minimize the synchronization cost, another merging type of clock calibration schemes has been proposed by using the external signal source with a stable period. ROCS [25], WizSync [26], and FLIGHT [27] exploit to use FM-radio, Wi-Fi beacons, and fluorescent lighting for calibration, respectively. The authors also propose to use power lines in [28]. Different from this work, those existing literatures mainly focus on proposing protocol-level solutions without considering the impacts from both internal and external network factors.

Recently, there have been attempts made to battle the environmental factors to improve the synchronization accuracy in WSNs. In [10], the authors propose two methods to estimate a neighbor's stability with respect to the temperature changes. EACS is further proposed in [11] to better capture the temperature dynamics using an additional information aided multi-model Kalman filter (AMKF). From a theoretical perspective, the authors in [12] analyze the clock synchronization performance with a deterministic accuracy guarantee. Different from those studies that mainly consider environmental impacts, in this paper, we focus on the energy dynamics, which are strongly associated with internal network operations and thus quite different from external environment dynamics. Even the impact of the environment has been eliminated completely, the energy heterogeneity can still affect the performance of time synchronization, which thus must be well addressed.

3 Problem Specification

In this section, we elaborate preliminary concepts, experiment results, motivation, and design challenges of EATS.

3.1 Clock Skew and Clock Synchronization

Each sensor node i is equipped with a hardware clock (e.g., a quartz oscillator or a RC circuit) that generates periodic signals ticking at a certain frequency f_0 (e.g., 32.768 KHz). In addition, each node maintains a logic clock that is used by upper-layer applications and advances as follows:

$$C_i(t) = \int_0^t \frac{f_i(\tau)}{f_0} \mathrm{d}\tau + C_i(0), \tag{1}$$

where $C_i(t)$ and $f_i(t)$ are the logic time and the instant frequency value of node i at time t, respectively. Limited by the manufactural constraints, the actual frequency values of different sensor nodes are hardly the same even they claim the same frequency. The frequency value of each sensor node actually varies. To capture such a dynamic property, $f_i(t)$ can be rephrased as:

$$f_i(t) = \overline{f}_i + \alpha_i(t), \tag{2}$$

where $\overline{f}_i$ is the average value of the clock frequency and $\alpha_i(t)$ indicates its variance. As we will see soon, due to the supply voltage dynamics, $\overline{f}_i$ is not necessarily equal to f_0 either. Eqs. (1) and (2) imply that for any pair of nodes, even the initial logic times are the same, their logic clocks will inevitably tick towards divergency and the term *clock skew*

Fig. 1. The experimental environment.

is used to measure the speed of offset (difference) accumulation between two logic clocks. To eliminate the inconsistency caused by the clock skew among different nodes, time synchronization must be performed.

The general synchronization process is as follows. The logic time of the sink is treated as a global timing reference and each node calibrates its own clock consistent to the global reference. To this end, the sink node sends out messages including time stamps to its neighbors for canceling their initial offsets. Afterwards, those 1-hop neighbors send out messages to further synchronize their own neighbors (those who are not synchronized yet). Such a working paradigm repeats until the entire network is processed. We find that the in-network energy issue continuously impacts the synchronization performance even the ambient environment is stable. In the next section, we report our observations obtained from experiments in detail.

3.2 Clock Uncertainty Due to Supply Voltage

In this section, we conduct a set of experiments to examine the clock uncertainty caused by the supply voltage. We adopt the TelosB platform as a vehicle to present the experimental results. TelosB motes are usually equipped with Citizen CMR200T crystal oscillator to generate 32.768 KHz onboard signals. Motes can be powered by batteries, solar cells or capacitances [14], [15], [16]. According to [29], the working voltage range is between 2.1 and 3.6 V. Although the study [30] observes that MHz oscillators are robust to supply voltage dynamics (e.g., the oscillator in CC2500 suffers from only 1 ppm clock skew when the supply voltage changes from 1.8 to 3.6 V). Those oscillators result in tremendous energy consumption and high manufacturing cost [31]. As a result, KHz (instead of MHz) oscillators are adopted on the main board to generate logic clocks [31].

Fig. 1 depicts the experimental environment. In the experiments, we use three sensor nodes (i.e., *A*, *B*, and *C*). To investigate the impact of the supply voltage on clock uncertainty, node *B* is directed powered by the USB port with a stable 3.0 V supply voltage and node *C* is connected to a direct current electrical source (DCES) whose supply voltage can be changed. To minimize measurement errors due to uncertain delays in packet preparation, channel access, packet transmission, and packet propagation at the sender side [9], we avoid nodes *B* and *C* sending time stamps to each other directly. Instead, we introduce node *A* to mimic the RBS protocol [7]. Node *A* serves as a beacon node periodically broadcasting beacon messages. Upon

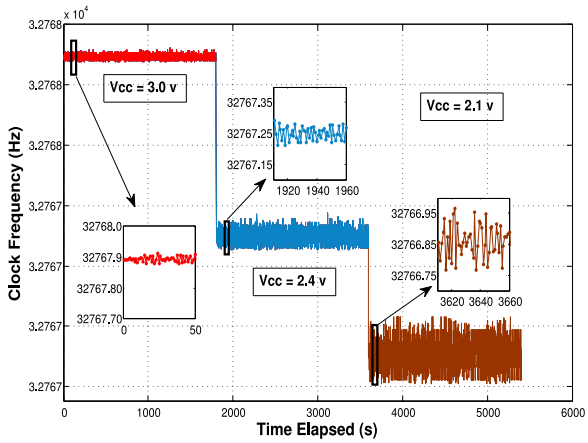


Fig. 2. Clock frequency under different supply voltages.

receiving a beacon message, nodes *B* and *C* record their local time and further transmit the recorded time information to the base station. By analyzing the logs of nodes *B* and *C*, we obtain the clock uncertainty of node *C* with respect to node *B* under different supply voltages. We conduct the experiments in a stable indoor environment to eliminate the influences from environmental factors, e.g., temperature and humidity, as much as possible.

We vary the supply voltage of sensor node *C* from 2.1 to 3.0 V with a step size 0.1 V (due to the hardware constraint, we cannot change its supply voltage continuously). For each supply voltage, the experiment is lasted for around 30 minutes. In Fig. 2, we plot the clock frequency of node *C* under three typical voltages, 3.0, 2.4, and 2.1 V. From the figure, we can see that the clock frequency under a higher supply voltage is closer to the claimed frequency value (i.e., 32.768 KHz) of the oscillator. On the other hand, a higher supply voltage yields a much more stable frequency. From statistics, the clock stability with the 3.0 V supply voltage is about two and four times higher than that with the 2.4 and 2.1 V supply voltages, respectively.

As node *B* is powered by USB (3.0 V) with a stable clock frequency, we further calculate node *C*'s clock skew and the corresponding variance with respect to node *B*. Fig. 3 depicts the clock skew under different supply voltages. From the figure, we observe that as the supply voltage decreases, the clock skew grows faster and the frequency variance increases as well. Fig. 3 implies that for most of the time (e.g., when supply voltage < 2.7 V), a sensor node will exhibit 20 ppm (parts per million) uncertainty at least due to the supply voltage.

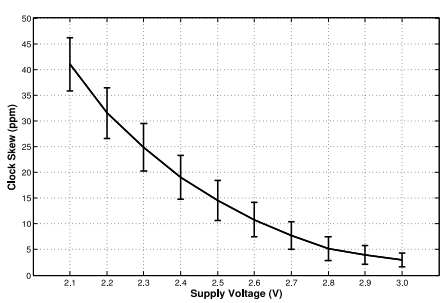


Fig. 3. Clock skew and variance of node *C* in the measurement.

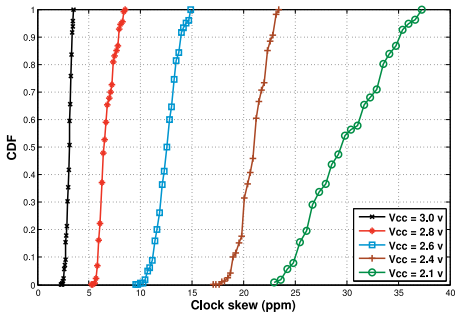


Fig. 4. CDF of avg. clock skew for 15 different motes synchronizing to node *B*.

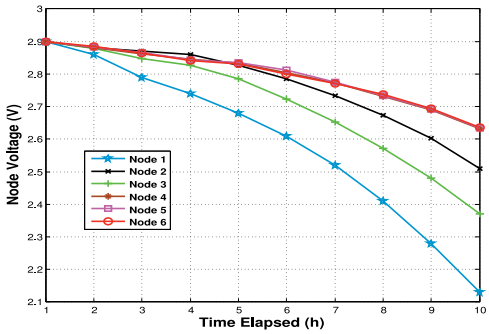


Fig. 5. Supply voltage changes of six nodes over time.

To avoid the measurement bias due to individual sensor motes, we further repeat the experiment 15 times and replace node C with a new TelosB mote each time. In Fig. 4, we plot the CDF of clock skews for all 15 motes under several typical supply voltages. Fig. 4 indicates that when the supply voltage is high (e.g., 3 V), clock is stable and the clock skew is smaller than 5 ppm. When the supply voltage decreases (e.g., 2.6 V), clock skew increases to 14 ppm for 90 percent motes. Clock skew can further increase to 23 ppm when the supply voltage becomes 2.4 V. Hence, for each pair of nodes that aims at being mutually synchronized, they will suffer from around 30-40 ppm inaccuracy with different supply voltages. Such uncertainty can result in up to 24 ms offset for every 10 minutes, which cannot be ignored in most applications [25], [26], [27].

Since clock skew solely determines the clock offset between any pair of nodes, the experiments in this section indicate that the impact of the energy supply needs to be carefully considered in the synchronization protocol design. As we will show in the next subsection, real sensor networks usually experience a heterogenous distribution of supply voltages across different sensor nodes, and such a distribution can vary with time.

3.3 Supply voltage heterogeneity

To examine the heterogeneity of sensor network supply voltage, we conduct a concrete experiment in the data collection application, which is one of the important networking services in WSNs.

Seven sensor nodes are used in this experiment with IDs assigned from 0 to 6. Node 0 is the sink node collecting data from all other nodes via the CTP protocol [32]. The topology of those seven nodes is given by Fig. 6. Each node periodically transmits a data packet back to the sink node following the formed data collection tree. In the experiment, we record the supply voltage of each node as time elapses. During the data collection, for the sensor nodes that are relatively far away from the sink node (e.g., node 6), their energy is mainly consumed for their own packet transmissions. On the contrary, the sensor nodes close to the sink node (e.g. node 1) not only generate their own traffic, but also relay the traffic for other nodes to the sink. As a consequence, the traffic burdens are not evenly distributed in the network and the energy consumption rates are heterogeneously distributed over the network. In Fig. 5, we plot the instant supply voltage of each node as time elapses. The result shows that the supply voltage is indeed not uniformly distributed. In particular, the energy draining rates of nodes 1 and 3 are $3 \sim 4$ times faster than other nodes on average in the experiment. Such a problem can become even worse in practical scenarios with larger network size, uneven traffic load pattern due to user demands, event detection heterogeneity, etc. Although sensor nodes in Fig. 6 are configured to be always-on, we also conduct the experiment in the low-duty-cycle context and observe similar phenomenon. Due to the page limitation, we omit the experimental result with the low-duty-cycle setting.

On the other hand, the traffic load is not the only factor that determines the supply voltage distribution in WSNs. As reported by recent works [14], [15], [16], rechargeable power supplies, e.g., solar cells and capacitances, exhibit great advantages in WSNs and thus they have been widely used in real systems and applications. In addition to the traffic load impact, the supply voltages of sensor nodes with a rechargeable power supply can further diverge during the recharging and energy leakage processes. According to [16], the complete charging and discharging durations (between 0 and 3 V) of solar cells are 45 and 15 min respectively and the energy harvesting operation is dynamically scheduled based on application requirements. Therefore, the rechargeable batteries lead to a more dynamic distribution of supply voltages, which further challenges the design of time synchronization protocols.

3.4 Design Overview

WSNs are usually connected in a multi-hop fashion. To synchronize all sensor nodes, synchronization messages are spread from the sink node to the entire network through a routing structure. For instance, traditional synchronization

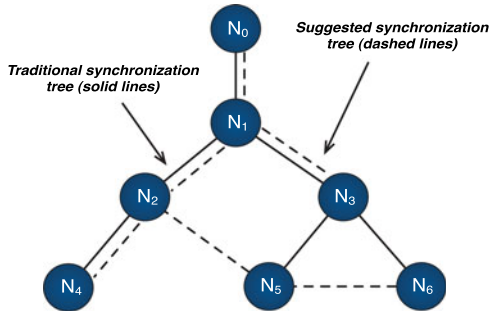


Fig. 6. Topology of six sensor nodes performing data collection.

protocols, like FTSP [9] and PulseSync [17], use the CTP tree to flood the synchronization information. Each node synchronizes its own clock to its parent node in the tree and such a parent is called *synchronization parent*. For example, in Fig. 6, node 3 is the synchronization parent of node 5. Due to the clock uncertainty and the measurement inaccuracy, the synchronization error may get accumulated hop by hop [17]. As supply voltages are usually not evenly distributed in the network and may vary as time goes by, we find that the synchronization structure used in traditional protocols may not always produce the best performance. In the example given by Fig. 6, nodes 5 and 6 will select node 3 as the synchronization parent. However, with the consideration of supply voltage, they may choose nodes 2 and 5 as their synchronization parents respectively, resulting in smaller synchronization errors. In addition, due to the traffic dynamics and the power charging/leakage behaviors, the optimal synchronization routing structure should be updated online according to available energy budgets.

A nature solution to cope with such dynamics is to frequently launch synchronization protocols. Existing protocols, however, rely on heavy intercommunications between nodes and lead to excessively high communication overhead and significant power drain in the system. To prolong the network lifetime, the time synchronization is only performed occasionally, while the supply voltage distribution can still vary during two consecutive synchronizations, even within one synchronization process. As a result, the design of EATS will focus on *how to select the best synchronization parent for each node* and *compensate the clock skew according to the supply voltage heterogeneity and its dynamics with long re-synchronization intervals*.

4 Protocol Design

In this section, we first mathematically analyze the impact due to the clock uncertainty in time synchronization. Based on the insights gained from the analysis, we elaborate the design details of EATS.

4.1 Uncertainty Analysis in Time Synchronization

We consider an arbitrary pair of nodes i and j conducting time synchronization using communication based protocols [6], [7], [9]. Without loss of generality, we assume node i is the synchronization parent of node j. Based on the time stamps recorded in messages from both sides (i.e., the parent side and the child side), child nodes can estimate the clock differences with respect to their parents. When node i transmits its first synchronization message, it records its local time t_{i0} in the message. Upon receiving this message, node j records its instant local time t_{j0}. The difference between t_{i0} and t_{j0} can be viewed as a sample of their clock offset. More precisely, at any time t, the offset between nodes i and j, i.e., $O_{ij}(t) = C_i(t) - C_j(t)$, is calculated by

$$\begin{aligned} O_{ij}(t) &= \left(\int_0^t \frac{f_i(\tau)}{f_0}\,\mathrm{d}\tau + C_i(0)\right) - \left(\int_0^t \frac{f_j(\tau)}{f_0}\,\mathrm{d}\tau + C_j(0)\right), \\ &= \frac{\Delta\overline{f}_{ij}}{f_0}t + E_i - E_j + O_{ij}(0), \end{aligned}$$

where $\Delta\overline{f}_{ij} = \overline{f}_i - \overline{f}_j$, $O_{ij}(0) = C_i(0) - C_j(0)$, $E_i = \int_0^t \alpha_i(\tau)\mathrm{d}\tau$, and $E_j = \int_0^t \alpha_j(\tau)\,\mathrm{d}\tau$. In particular, $\Delta\overline{f}_{ij}$ is the clock skew between node i and node j and $O_{ij}(0)$ refers to their initial clock offset. E_i and E_j are the errors caused by the communication jitter, quantization inaccuracy and clock uncertainty. Those errors are independent across different messages. According to [6], [10], E_l follows a normal distribution $\mathcal{N}(0, \sigma_l^2)$, where l is i or j.

Each obtained $O_{ij}(t)$ from the communication can be used to eliminate the offset between nodes i and j. On the other hand, after receiving sufficient amount of $O_{ij}(t)$ samples, node j applies the linear regression to estimate the clock skew $\Delta\overline{f}_{ij}$. By knowing their mutual clock skew, node j can compensate the advancing speed of its logic clock such that the offset accumulated with a much slower speed. If the clock skew can be accurately estimated, the time interval between two consecutive synchronizations can be prolonged significantly, which results in much reduced communication overhead and energy cost. By using the linear regression, the clock skew is estimated as follows:

$$\begin{aligned} \Delta\overline{f}_{ij} &= \frac{f_0}{T}\frac{\sum_{k=1}^{W}(O_{ij}(kT) - \overline{O}_{ij})(k - \frac{W+1}{2})}{\sum_{k=1}^{W}(k - \frac{W+1}{2})^2}, \\ &= \frac{12\sum_{k=1}^{W} kO_{ij}(kT) - 6W(W+1)}{W(W^2-1)}, \end{aligned} \tag{3}$$

where W is the window size of linear regression, T is the synchronization interval, and $\overline{O}_{ij} = \sum_{k=1}^{W} O_{ij}(kT)/W$. Based on [10], the variance of the estimated $\Delta\overline{f}_{ij}$ is given by

$$Var(\Delta\overline{f}_{ij}) = \frac{f_0^2}{\left(T^2\sum_{k=1}^{W}(k - \frac{W+1}{2})^2\right)}\left(\sigma_i^2 + \sigma_j^2\right). \tag{4}$$

As shown by Eq. (4), the value of $Var(\Delta\overline{f}_{ij})$ is mainly determined by the clock variances of the synchronization parent (node i) and the node itself (node j). Since a higher $Var(\Delta\overline{f}_{ij})$ results in a worse estimation of the clock skew $\Delta\overline{f}_{ij}$, node j should select a parent with the smallest variance of the clock frequency. As the clock re-synchronization interval T can be relatively long (e.g., 30 minutes), the supply voltage distribution could vary during an interval. Such dynamics should also be addressed when the synchronization routing structure is constructed and maintained. On the other hand, the skew variance in Eq. (4) is related to node j's clock uncertainty as well and we find that it is possible to compensate such uncertainty at node j's side to further improve the accuracy of the skew estimation. In the next two subsections, we introduce *synchronization parent selection* and *skew compensation* two modules to address above issues in EATS.

4.2 Synchronization Parent Selection

The synchronization parent selection module in EATS aims at choosing a stable synchronization parent for each node within the next synchronization interval. As aforementioned in Fig. 3, a higher supply voltage yields to a more stable clock that suffers from a small variance of clock frequency. Thus, to select a synchronization parent, any node j should consider the current supply voltages of its neighbors.

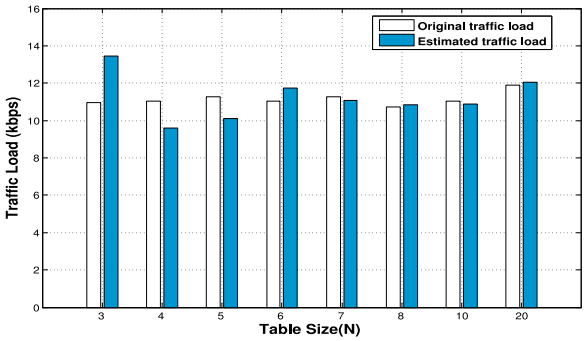


Fig. 7. Traffic load prediction with different table sizes.

On the other hand, the instant supply voltage value of each neighbor may not properly characterize the clock skew performance in an entire re-synchronization interval. Their supply voltages can change due to traffic load, recharging operations, etc. In many deployed sensor networks [13], network-wide traffic load is relatively stable as nodes periodically generate packets. We find that traffic loads in the previous N intervals immediately before the current synchronization can well approximate the traffic load in the next interval. By using the weighted average on the traffic loads in the previous N intervals, we can estimate the traffic burden in the near future as shown by Eq. (5). Although the prediction is linear, it turns out to be quite effective and computationally efficient in practice.

$$\bar{l}_i = \sum_{k=1}^{N} w_k \cdot l_i(k), \tag{5}$$

where $\bar{l}_i$ is the predicted traffic load within the next re-synchronization interval of node i and $l_i(k)$ indicates its traffic load in the kth past interval. In addition, w_k is a weighting factor where $\sum_{k=1}^{N} w_k = 1$ and w_1 corresponds to the most recent interval in the past. In the current EATS, we assign w_1 to the largest weight 0.4 and w_2 to the second largest weight 0.2. All other factors equally share the remaining 0.4 fraction. The rationale behind is that recent intervals should contribute more to the estimated result. To validate our weight assignment, we test the prediction accuracy using the network given by Fig. 6. We vary N from 3 to 20 and compare the predicted traffic load with the real traffic load in Fig. 7. From the figure, we can see that the prediction accuracy increases as N becoming larger. In particular, when $N > 7$, Eq. (5) can fairly approximate the traffic load in the near future. Considering the computation cost, we select N to be 7 in EATS. So far, we consider a normal scenario, i.e., relatively stable traffic load in the network. In case of bursty traffics, it may decrease the traffic approximation accuracy. Fortunately, the approximation error is not accumulated and can be corrected prior to the next re-synchronization.

With $\bar{l}_i$, the supply voltage reduction of node i in the next re-synchronization interval ΔV_i equals $\bar{l}_i \cdot \bar{r}_i$, where $\bar{r}_i$ is the voltage reduction rate (with respect to traffic load) in the next interval. Given an instant voltage value, the voltage reduction rate can be obtained from a theoretical voltage reduction curve. However, such a method leads to derivative operations and restricts EATS to be battery-type-dependent. Therefore, in EATS, node i calculates its voltage

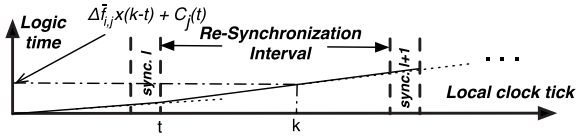


Fig. 8. Illustration of the logic time calculation

reduction rate $r_i(k) = l_i(k)/\Delta V_i(k)$ for the k-th past interval, where $\Delta V_i(k)$ is the supply voltage reduction in this interval, and further adopts linear regression to predict $\bar{r}_i$ using historical $r_i(k)$, where $k = 1, \ldots, M$ and M is empirically set to be 3 in EATS. The supply voltage reduction of node i in the next re-synchronization interval, ΔV_i, is finally estimated as $\Delta V_i = \bar{l}_i \cdot \bar{r}_i$.

For recharging operations, those operations are usually controlled by state-of-the-art protocols, like [14], [15], [16], with a predicable charging profile. Hence, if WSNs are equipped with solar cells or capacitances, each node can determine the voltage change within the next interval due to the recharging operations. Such a result can be additively or deductively combined with the result obtained from Eq. (5).

4.2.1 Module Specification

Supposing a synchronization operation is conducted at time t. The supply voltage value of parent node i is $V_i(t)$ and the estimated voltage change (due to the traffic load and recharging operations) within the next synchronization interval is ΔV_i, which can be either positive or negative. After receiving such information from each candidate parent (e.g., a neighbor with a smaller hop count to the sink), node j selects the synchronization parent as follows:

$$parent(j) = \arg\max_i (V_i(t) - \Delta V_i/2), \quad \forall i \in P_j \tag{6}$$

where P_j is the candidate parent set of node j. After selecting the synchronization parent, node j applies linear regression to estimate its clock skew and offset with respect to the synchronization parent, and updates its logic clock accordingly. More precisely, the synchronization parent selection consists of following three steps:

- Each node i' in P_j advertises its voltage $V_{i'}(t)$ and the estimated voltage change $\Delta V_i'$ in the synchronization messages.
- Node j selects its synchronization parent using Eq. (6) and the selected parent is denoted as node i.
- Node j applies the linear regression to estimate the clock skew $\Delta\overline{f}_{ij}$ and the offset $O_{ij}(t)$, where t indicates the time for node j to receive the last message from node i in this time synchronization process.

After obtaining $O_{ij}(t)$, the logic time of node j is updated to $C_j(t) = C_j(t) + O_{ij}(t)$ for eliminating the accumulated offset. In addition, the logic time within the next synchronization interval is maintained as

$$C_j(k) = \Delta\overline{f}_{ij} \times (k - t) + C_j(t), \tag{7}$$

where $k > t$, by using the estimated clock skew. Such a logic time maintenance can be illustrated by Fig. 8. In Fig. 8, node j eliminates offset at time t and the slope of the

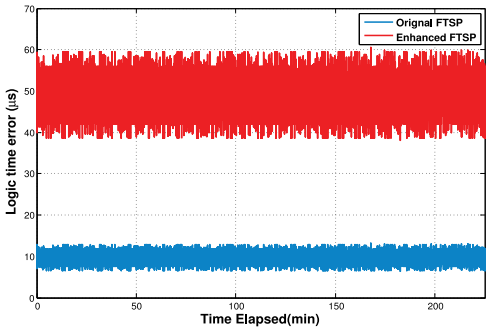


Fig. 9. Clock offset comparison.

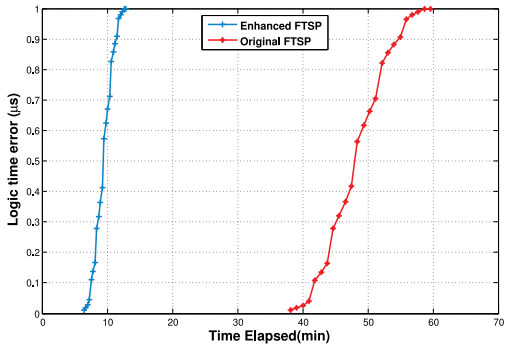


Fig. 10. CDF of clock offset.

straight line between *sync.* l and *sync.* $l+1$ indicates the estimated clock skew $\Delta\overline{f}_{ij}$ for the given re-synchronization interval in the figure. To maintain a consistent logic time with other nodes in the network, node j converts the local clock ticks k to $\Delta\overline{f}_{ij} \times (k-t) + C_j(t)$ as shown by Fig. 8.

4.2.2 Experimental Evaluation

We conduct an experiment to validate the effectiveness of the *synchronization parent selection* module using TelosB sensor nodes. The synchronization structure used in the experiment is plotted in Fig. 6. Node 0 is the base station. Node 3 is fixed as the synchronization parent of node 6 and the supply voltage of node 3 is 2.2 V. On the contrary, node 5 can select either node 2 (2.8 V) or node 3 as parent. In the experiment, the underlying synchronization protocol adopted is FTSP. We conduct the experiment using the original FTSP protocol and the enhanced version by the *synchronization parent selection* module. We focus on node 5 and report its performance in Figs. 9 and 10.

In Fig. 9, we calculate the offset of logic time between node 5 and the base station (node 0) every 10 seconds. In the experiment, the original FTSP uses the CTP tree and selects node 3 as the synchronization parent for node 5. Differently, the enhanced version of FTSP chooses node 1 as the parent. From the figure, we can see that node 5 with the original FTSP exhibits much worse time synchronization performance in terms of both the average logic time error and the variance. From the statistics, the average error of the original FTSP is close to 50 μs while the enhanced version is only about 10 μs. The *synchronization parent selection* module improves the synchronization performance more than 5 times in the experiment. In Fig. 10, we further plot the CDF of the result in Fig. 9. Fig. 10 indicates that the range of the offset with the original FTSP is between 38 and 60 μs. On the contrary, the enhanced FTSP introduces a much narrow range from 6 to 11 μs merely.

4.3 Clock Skew Compensation

As indicated by Eq. (4), for any node j, the accuracy of the estimated skew is also related to its own clock uncertainty. If such uncertainty can be (even partially) compensated, the skew estimation accuracy can be further improved and the synchronization interval can be prolonged to save energy.

To this end, we introduce a *clock skew table* for each node that contains clock skews under different supply voltages. The table can be formed using a direct current electrical source. Before the network deployment, sensor nodes can be powered by a DCES and clock skew tables are formed by varying the voltage output of DCES. The extra cost of the clock skew table is only slight due to following two reasons. First, one DCES can simultaneously power plenty of sensor nodes using parallel connection. Second, the output voltage of DCES does not need to be changed continuously. Within a small supply voltage step (e.g., 0.1 V), the skew is almost linearly changed. Therefore, the clock skew table can be constructed with only a series of discrete voltage values. The clock skew table is robust to the ambient environment. Since our protocol is compatible to other time synchronization protocols [10], [11], [12], if the surrounding environment changes (e.g., the temperature changes), existing solutions can be used to compensate such changes. Hence, only one copy of the clock skew table needs to be formed. In our current implementation, the voltage step length is 0.1 V.

4.3.1 Module Specification

After the construction of the clock skew table, the self clock skew compensation can be achieved as follows. Supposing V_1 and V_2 are two consecutive voltage values stored in the clock skew table. For any V_e, where $V_1 \le V_e \le V_2$, the total amount of skew to be compensated can be calculated by:

$$\Delta\hat{f}_j(V_e) = \Delta\hat{f}_j(V_1) + \frac{\Delta\hat{f}_j(V_2) - \Delta\hat{f}_j(V_1)}{V_2 - V_1}(V_e - V_1), \quad (8)$$

where $\Delta\hat{f}_i(V_1)$ and $\Delta\hat{f}_i(V_2)$ indicate the recorded skews under voltages V_1 and V_2 in the clock skew table, respectively. After obtaining $\Delta\hat{f}_i(V_e)$, instead of using $\Delta\overline{f}_{ij}$ directly in Eq. (7), a more accurate logic time can be gained by Eq. (9).

$$C_j(k) = (\Delta\overline{f}_{ij} + \Delta\hat{f}_j(V_e)) \times (k-t) + C_j(t). \quad (9)$$

One point worth noting is that the skew compensation module needs not to be triggered continuously. Instead, the module can be launched periodically for saving energy. In our current design, this module is enabled every 100 s.

4.3.2 Experimental Evaluation

To validate the effectiveness of the *skew compensation* module, we conduct another experiment. We synchronize a pair of nodes (A and B) to the base station. Node A runs the original FTSP and node B runs the FTSP enhanced by the *skew compensation* module. The supply voltages of those two

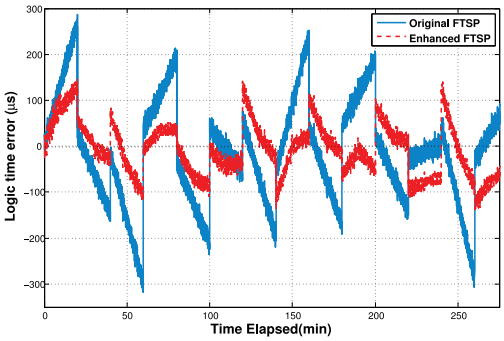


Fig. 11. Logic time error comparison.

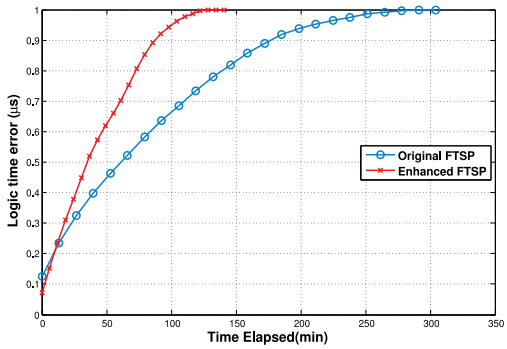


Fig. 12. CDF of logic time errors.

nodes are both 2.4 V. In the experiment, the synchronization interval is 20 min and node B self compensates its clock in the middle of each re-synchronization interval.

In Fig. 11, we plot logic time errors of nodes A and B as time elapses. From the figure, we see that within the first half of each re-synchronization interval, nodes A and B have similar synchronization performance. However, after the self skew compensation, node B experiences a much smaller skew compared with node A. Fig. 11 shows that with the skew compensation module, sensor nodes can achieve comparable synchronization performance with much prolonged re-synchronization intervals. As a result, the communication overhead and energy can be largely reduced. To further quantify the result obtained in Fig. 11, we draw the CDF of logic time errors in Fig. 12. Without the skew compensation module, the logic time error of node A is less than 180 μs for most of the time and the largest error is 310 μs. However, the module can reduce the error to 90 μs for the majority of the time and bounded by 150 μs.

Algorithm 1: The EATS protocol at sensor node j

1 Collect $V_{i'}(t)$ and $\Delta V_{i'}$ from node j' neighbors in P_j;
2 Calculate $parent(j) = arg\max_i(V_{i'}(t) - \frac{\Delta V_{i'}}{2}),\ \forall i \in P_j$;
3 Obtain the clock skew $\Delta\overline{f}_{ij}$ and the offset $O_{ij}(t)$;
4 Collect $\left(C_j(k), O_{ij}(k)\right)$ in table RegTbl;
5 $(\Delta\overline{f}_{ij}, \bar{O}_{ij})$ = Regression(RegTbl);
6 **while** *Every 100 seconds* **do**
7 Measure the local voltage V_e;
8 $\Delta\hat{f}_j(V_e) = \Delta\hat{f}_j(V_1) + \frac{\Delta\hat{f}_j(V_2) - \Delta\hat{f}_j(V_1)}{V_2 - V_1}(V_e - V_1)$, $V_e \in [V_1, V_2]$;
9 Record $C_j(k) = (\Delta\overline{f}_{ij} + \Delta\hat{f}_j(V_e)) \times (k-t) + C_j(t)$ for the event;

4.4 EATS protocol

With the synchronization parent selection module and the skew compensation module, the EATS protocol can be summarized by Algorithm 1. For any node j in the network, line 1 to line 3 invokes the function of the synchronization parent selection module. Node j collects necessary information in line 1 and determines the synchronization parent in line 2. After the synchronization parent has been selected for the next re-synchronization interval, the clock skew and offset are computed in line 3. In our current implementation, the skew compensation module is periodically triggered to save energy from line 6 to line 9. Whenever the skew compensation is triggered, node j measures its current supply voltage and calculates the skew to be compensated in lines 7 and 8, respectively. Then the logic clock of node j advances with the compensated clock skew as indicated by line 9.

5 Trace-Driven Evaluation

In the previous section, we have introduced EATS and tested the effectiveness of two modules in EATS with the TelosB platform. To further evaluate the network-wide performance of EATS, we conduct large-scale simulations in this section. To test a realistic network setting, simulations are conducted with a real network topology harvested from GreenOrbs [13]. GreenOrbs is a long-term and large-scale wireless sensor network deployed in the forest, which contains 433 nodes and has continuously worked over years. From the harvested trace over six months, we observe that the dynamics of wireless links result in fluctuation of the network topology. To mimic the link estimation for real data transmissions, we filter out lossy links with small RSSI values. In particular, links with the packet reception ratio lower than 30 percent or RSSI smaller than -80 dB are excluded by the filter. By doing so, we obtain a stable network topology for simulations. The topology includes $6,567$ links. Due to page limit, topology is not illustrated in the paper.

5.1 Simulation Settings

In the trace, sensor nodes are deployed in a 700×200 m rectangle field with the default transmission power. Parameters of sensor nodes are configured based on the Telosb mote specification [29]. The initial supply voltage of each node is uniformly distributed within $[2.9, 3.1$ V$]$. The underlying time synchronization protocols used are FTSP [9] and PulseSync [17]. The default re-synchronization interval is 20 minutes. To evaluate our proposed EATS protocol, we integrate EATS with the FTSP and PulseSync protocols, and compare with the original protocols. The period of the skew compensation of EATS is 100 seconds. In the network, the sink node is placed at $(-200.2, 115.7)$. To examine the impact of traffic load on the synchronization performance, the sink node uses the CTP protocol to collect data from all other nodes. To ensure the reachability of each node after selecting its synchronization parent, the selected parent should have a smaller hop count than the node itself. The default traffic rate is 5 kbps and the energy draining rate of

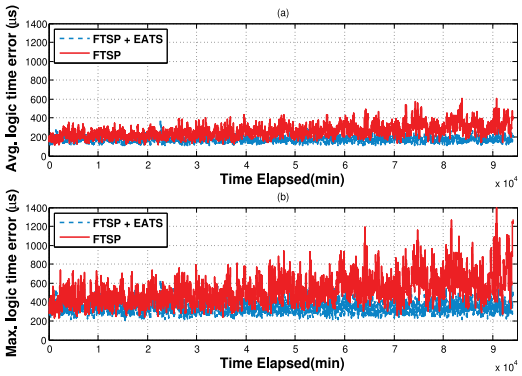


Fig. 13. Logic time errors with FTSP.

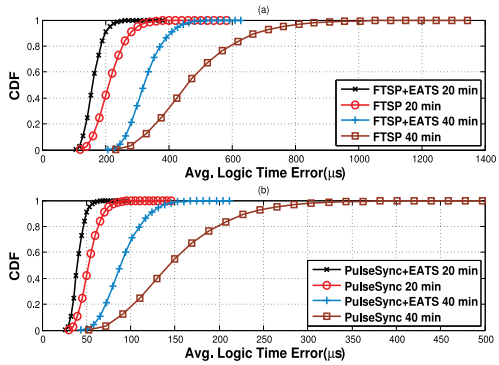


Fig. 15. CDF of avg. logic time errors with different synchronization intervals.

the battery is set to be 0.000001 V/1 kbps. To mimic the traffic dynamics in real applications, during the simulation we also manually trigger the traffic variance and investigate the impact of the traffic dynamics.

5.2 Simulation Results

5.2.1 Logic Time Error versus Time

We simulate a $94,000$-second time synchronization process across the entire network. In Fig. 13, we plot the average and maximum logic time errors of FTSP and FTSP+EATS as time elapses. Initially, the average logic time error of the original FTSP protocol is around 200 μs. As time elapses, the supply voltage of each node decreases. Clocks become unstable and the average logic time error thus increases. After $70,000$ seconds, the average logic time of FTSP approximates to be 300 μs for most of the time and climbs up to 600 μs occasionally. In addition, the variance of the average error is as high as 420 μs. During the entire process, the maximum logic time error is smaller than 800 μs for most time and bounded within 1400 μs. Compared with FTSP, EATS effectively improves the synchronization accuracy. The average logic time error of FTSP+EATS is around 180 μs and the maximum error is smaller than 660 μs. According to statistics, FTSP+EATS initially improves the synchronization performance by 1.4 times. As time goes by, FTSP+EATS outperforms FTSP at least 1.6 times.

In Fig. 14, we observe a similar performance trend on the average and maximum logic time errors of PulseSync and PulseSync+EATS. Due to the rapid flooding technique [17], the synchronization performance in Fig. 14 is generally better compared with Fig. 13. The average logic time error of PulseSync increases from 52 to 85 μs in the entire synchronization process. The maximum error is below 500 μs. On the contrary, PulseSync+EATS exhibits much smaller errors. The average error of PulseSync+EATS is below 50 μs for the majority of the time and the maximum error is bounded within 200 μs. According to statistics, PulseSync+EATS outperforms PulseSync at least 1.4 times.

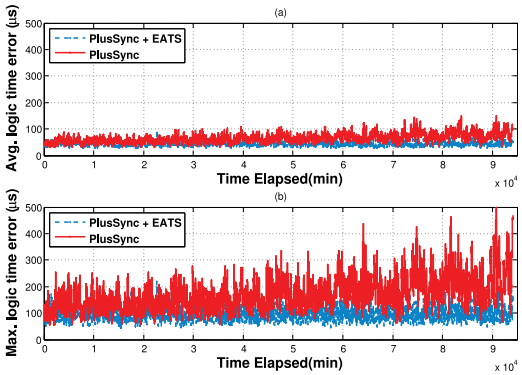


Fig. 14. Logic time errors with PulseSync.

5.2.2 Logic Time Errors with Different Intervals

In Fig. 15, we first plot the CDF of *average* logic time errors obtained in Figs. 13 and 14, in which the re-synchronization interval is 20 minutes. From the figure, we see that for 90 percent of the time, the average logic time error of FTSP+EATS (PulseSync+EATS) is smaller than 200 μs (50 μs) and the distribution is within a narrow interval. In contrast, the average logic time error of FTSP (PulseSync) is smaller than 300 μs (75 μs) for 90 percent of the time and suffers from a much longer tail in Fig. 15. In Fig. 15, we further evaluate the performance of those four schemes with a longer re-synchronization interval (i.e., 40 minutes). The average logic time error of FTSP+EATS (PulseSync+EATS) is less than 420 μs (125 μs) for 90 percent of the time and it is bounded within 625 μs (230 μs). Compared with FTSP+EATS (PulseSync+EATS), the average logic time of FTSP (PulseSync) increases 57 percent (69 percent) for most of the time and the tail is prolonged to 1380 μs (500 μs).

5.2.3 Logic Time Errors with Different Traffic Loads

We evaluate the synchronization performance with various traffic loads in Fig. 16. We vary the traffic load of each node from 5 to 80 kbps. To simulate the traffic dynamics, we refer to the link qualities recorded in the GreenOrbs trace and implement the CTP protocol to dynamically update the data collection tree. As PulseSync and PulseSync+EATS perform generally better than FTSP and FTSP+EATS, respectively. In the remaining experiments, we will mainly focus on the former two schemes. Fig. 16 shows that when the traffic load is light, PulseSync and PulseSync+EATS have comparable performance, e.g., PulseSync+EATS reduces the logic time error by 23 percent than PulseSync when the traffic load is 5 kbps. When

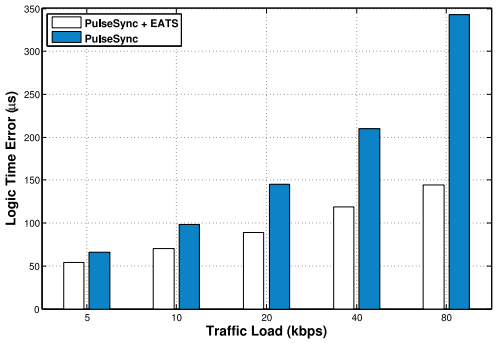


Fig. 16. Logic time error v.s. different traffic loads.

the traffic load increases, the energy heterogeneity becomes worse and more sensitive to the traffic dynamics. As a result, the logic time error of PulseSync increases rapidly. When the traffic load changes from 5 to 80 kbps, the error of PulseSync increases about 5.1 times. However, Fig. 16 depicts that the logic time error increasing of PulseSync+EATS is much smaller and the average error is controlled $< 150\ \mu s$ in all the cases within a 20-min re-synchronization interval.

5.2.4 Energy Consumption Comparison

When the sensor motes MCU is idle, the working power is 2.7 mW. After EATS starts calculating, the average power only slightly increases, e.g., 5.4 mW. Then, we stop EATS and open the radio, the working power suddenly jump to 50 mW. Therefore, compared with time synchronization protocols, which mainly rely on the radio communications, the extra overhead caused by EATS is quite small. On the other hand, EATS can improve the accuracy of nodes clocks, and the re-synchronization interval can be significantly prolonged. As a result, the overall energy consumption of sensor nodes can be largely reduced. We let PulseSync and PulseSync+EATS achieve similar synchronization accuracy by setting different re-synchronization intervals for them, and evaluate their energy consumption. According to Fig. 17, PulseSync+EATS reduces $60\ \mu$W energy consumption compared with PulseSync, which is significant for time synchronization protocols [25], [26], [27]. For the other two settings, PulseSync+EATS outperforms PulseSync as well. In particular, PulseSync+EATS reduces 50 and $21\ \mu$W when the average logic time error is controlled within 200 and $300\ \mu s$, respectively.

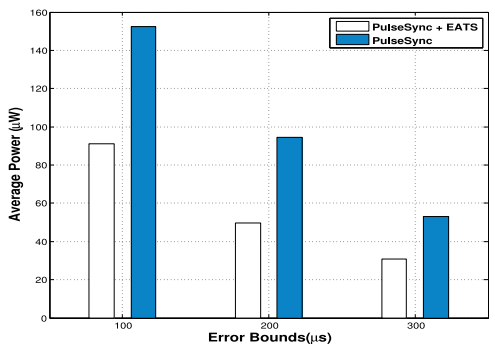


Fig. 17. Energy consumption comparison.

6 Conclusion and Future Work

In this paper, we study time synchronization with respect to the in-network energy heterogeneity in WSNs. We mathematically analyze the root cause of such clock uncertainty and propose the EATS protocol. Sensor nodes with EATS can intelligently select the best synchronization parents that minimize the negative impact of the energy heterogeneity. In addition, nodes can make use of local energy information and further calibrate the clocks. With EATS, sensor nodes can maintain more consistent logic time in the network and the energy consumption can be largely reduced. We implement EATS with TelosB motes and evaluate its effectiveness and efficiency through extensive experiments and simulations. Future works include evaluating the performance of EATS in large-scale WSNs, countermeasures to bursty traffics, and a uniform synchronization framework that integrates internal energy distribution and external environment factors into consideration.

Acknowledgments

This study was supported by Singapore MOE AcRF Tier 2 grant MOE2012-T2-1-070. This work was also supported by NSFC Grant No. 61303233, No. 61272456, No. 61272437, No. 61073189, Innovation Program of Shanghai Municipal Education Commission No. 13ZZ131 and Foundation Key Project of Shanghai Science and Technology Committee No. 12JC1404500, Project of Shanghai Science and Technology Committee 12510500700.

References

[1] S.-J. Tang, X. Mao, and X.-Y. Li, "Efficient and Fast Distributed Top-K Query Protocol in Wireless Sensor Networks," *Proc. IEEE 19th Int'l Conf. Network Protocols (ICNP)*, 2011.

[2] W. Liu, H. Jiang, X. Bai, G. Tan, G. Wang, W. Liu, and K. Cai, "Skeleton Extraction from Incomplete Boundaries in Sensor Networks Based on Distance Transform," *Proc. IEEE 32nd Int'l Conf. Distributed Computing Systems (ICDCS)*, 2012.

[3] G. Ghidini and S. Das, "Cool: An Energy-Efficient Markov Chain-Based Randomized Duty Cycling Scheme for Wireless Sensor Networks," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2011.

[4] J. Chen, W. Xu, S. He, Y. Sun, P. Thulasiraman, and X. Shen, "Utility-Based Asynchronous Flow Control Algorithm for Wireless Sensor Networks," *IEEE J. Selected Areas in Comm.*, vol. 28, no. 7, pp. 1116-1126, Sept. 2010.

[5] B. Kerkez, R. Rice, S. Glaser, R. Bales, M. Meadows, and P. Saksa, "Wireless Sensor Networks for Distributed Snow Depth Monitoring in the Sierra Nevada," *Proc. Western Snow Conf.*, 2011.

[6] Z. Zhong, P. Chen, and T. He, "On-Demand Time Synchronization with Predictable Accuracy," *Proc. IEEE INFOCOM*, 2011.

[7] J. Elson, L. Girod, and D. Estrin, "Fine Grained Network Time Synchronization Using Reference Broadcasts," *Proc. USENIX Fifth Symp. Operating Systems Design and Implementation (OSDI)*, 2002.

[8] S. Ganeriwal, R. Kumar, and M. Srivastava, "Timing Sync Protocol for Sensor Networks," *Proc. ACM First Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, 2003.

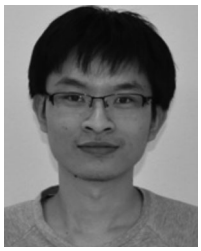
[9] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, 2004.

[10] T. Schmid, Z. Charbiwala, Z. Anagnostopoulou, M. Srivastava, and P. Dutta, "A Case Against Routing-Integrated Time Synchronization," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, 2010.

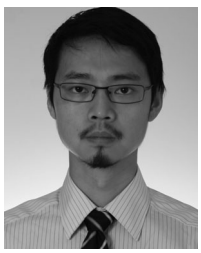
[11] Z. Yang, L. Cai, Y. Liu, and J. Pan, "Environment-Aware Clock Skew Estimation and Synchronization for Wireless Sensor Networks," *Proc. IEEE INFOCOM*, 2012.

[12] R. Sugihara and R. Gupta, "Clock Synchronization with Deterministic Accuracy Guarantee," *Proc. Eighth European Conf. Wireless Sensor Networks (EWSN)*, 2011.
[13] Y. Liu et al., "Does Wireless Sensor Network Scale? A Measurement Study on Greenorbs," *Proc. IEEE INFOCOM*, 2011.
[14] Y. Gu, T. Zhu, and T. He, "ESC: Energy Synchronized Communication in Sustainable Sensor Networks," *Proc. IEEE 17th Int'l Conf. Network Protocols (ICNP)*, 2009.
[15] T. Zhu, Y. Gu, T. He, and Z. Zhang, "Eshare: A Capacitor-Driven Energy Storage and Sharing Network for Long-Term Operation," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, pp. 239-252, 2010.
[16] S.-J. Tang, X. Li, X. Shen, J. Zhang, G. Dai, and S. Das, "Cool: On Coverage with Solar-Powered Sensors," *Proc. IEEE Int'l Conf. Distributed Computing Systems (ICDCS)*, 2011.
[17] C. Lenzen, P. Sommer, and R. Wattenhofer, "Optimal Clock Synchronization in Networks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, 2009.
[18] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, "Efficient Network Flooding and Time Synchronization with Glossy," *Proc. ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2011.
[19] J. Koo, R. Panta, S. Bagchi, and L. Montestruque, "A Tale of two Synchronizing Clocks," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, 2009.
[20] S. Ganeriwal, I. Tsigkogiannis, H. Shim, V. Tsiatsis, M. Srivastava, and D. Ganesan, "Estimating Clock Uncertainty for Efficient Duty-Cycling in Sensor Networks," *IEEE/ACM Trans. Networking*, vol. 17, no. 3, pp. 843-856, June 2009.
[21] L. Schenato and F. Fiorentin, "Average Timesynch: A Consensus-Based Protocol for Clock Synchronization in Wireless Sensor Networks," *Automatica*, vol. 47, pp. 1878-1886, 2011.
[22] J. He, P. Cheng, L. Shi, and J. Chen, "Time Synchronization in WSNs: A Maximum Value Based Consensus Approach," *Proc. IEEE 50th Conf. Decision and Control and European Control Conf. (CDC-ECC)*, 2011.
[23] B. Hamilton, X. Ma, Q. Zhao, and J. Xu, "Aces: Adaptive Clock Estimation and Synchronization Using Kalman Filtering," *Proc. ACM Mobicom*, 2008.
[24] T. Schmid, P. Dutta, and M. Srivastava, "High Resolution, Low Power Time Synchronization an Oxymoron no More," *Proc. ACM/IEEE Int'l Conf. Information Processing in Sensor Networks (IPSN)*, 2010.
[25] L. Li, G. Xing, L. Sun, W. Huangfu, R. Zhou, and H. Zhu, "Exploiting FM Radio Data System for Adaptive Clock Calibration in Sensor Networks," *Proc. ACM Ninth Int'l Conf. Mobile Systems, Applications, and Services (Mobisys)*, 2011.
[26] T. Hao, R. Zhou, G. Xing, and M. Mutka, "WizSync: Exploiting Wi-Fi Infrastructure for Clock Synchronization in Wireless Sensor Networks," *Proc. IEEE Real-Time Systems Symp. (RTSS)*, 2011.
[27] Z. Li, W. Chen, C. Li, M. Li, X.-Y. Li, and Y. Liu, "Flight: Clock Calibration Using Fluorescent Lighting," *Proc. ACM MobiCom*, 2012.
[28] A. Rowe, V. Gupta, and R. Rajkumar, "Low-Power Clock Synchronization Using Electromagnetic Energy Radiating from AC Power Lines," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (Sensys)*, 2009.
[29] "TelosB," http://www2.ece.ohio-state.edu/~bibyk/ee582/telosMote.pdf, 2014.
[30] B. Kerkez, "Adaptive Time Synchronization and Frequency Channel Hopping for Wireless Sensor Networks," master's thesis, EECS Department, Univ. of California, 2012.
[31] P. Spevak and P. Forstner, *MSP430 32-kHz Crystal Oscillators*, Application Report, 2009.
[32] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," *Proc. ACM Int'l Conf. Embedded Networked Sensor Systems (SenSys)*, 2009.

Zhenjiang Li (M'12) received the BE degree from the Department of Computer Science and Technology, Xi'an Jiaotong University, China, in 2007, the Mphil and PhD degrees from the Department of Electronic and Computer Engineering and Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2009 and 2012, respectively. His research interests include networked distributed systems and mobile computing. He is a member of the IEEE and the ACM.



Wenwei Chen received the BE degree from the Automation Department, University of Science and Technology, in 2011. He is currently working toward the PhD degree from Nanyang Technological University. His current research interest includes wireless sensor networks.



Mo Li (M'06) received the BS degree from the Department of Computer Science and Technology, Tsinghua University, China, in 2004, and the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2009. He is currently an assistant professor in the School of Computer Engineering, Nanyang Technological University, Singapore. His research interest includes wireless sensor networking, pervasive computing, mobile and wireless computing. He is a member of the IEEE and the ACM.

Jingsheng Lei received the BS degree in mathematics from Shanxi Normal University in 1987, and the MS and PhD degrees in computer science from Xinjiang University, in 2000 and 2003, respectively. He is a professor and the dean of the College of Computer Science and Technology, Shanghai University of Electronic Power. He has wide research interests, mainly including machine learning, data mining, pattern recognition and cloud computing. In these areas he has published more than 100 papers in international journals or conferences. He is as an editor-in-chief of the *Journal of Computational Information Systems*. He is a member of the Artificial Intelligence and Pattern Recognition Technical Committee of the China Computer Federation (CCF), a member of the Machine Learning Technnical Committee of the Chinese Association of Artificial Intelligence (CAAI), and a member of the Academic Committee of ACM Shanghai chapter.