On-Demand Time Synchronization with Predictable Accuracy

Ziguo Zhong
Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455
zhong@cs.umn.edu

Pengpeng Chen
Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455
pengpeng@cs.umn.edu

Tian He
Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455
tianhe@cs.umn.edu

***Abstract*—Time synchronization remains as a challenging task in wireless sensor networks that face severe resource constraints. Unlike previous work's aiming at pure clock accuracy, this paper proposes On-Demand Synchronization (ODS), a design to achieve efficient clock synchronization with customized performance. By carefully modeling the error uncertainty of skew detection and its propagation over time, ODS develops a novel uncertainty-driven mechanism to adaptively adjust each clock calibration interval individually rather than traditional periodic synchronization, for minimum communication overhead while satisfying the desired accuracy. Besides, ODS provides a nice feature of predictable accuracy, allowing nodes to acquire the useful information about real-time qualities of their synchronization. We implemented ODS on the MICAz mote platform, and evaluated it through test-bed experiments with 33 nodes as well as simulations obeying real world conditions. Results show that ODS is practical, flexible, and quickly adapts to varying accuracy requirements and different traffic load in the network for improved system efficiency.**

I. Introduction

Time synchronization is one of the most fundamental and widely employed middle-ware services in wireless sensor networks (WSN) [10]. It allows nodes in the network to have a common notion of time, either respect to a global reference or among themselves [9]. Accurate time synchronization is critical for saving communication energy [26][28], promoting localization accuracy [1][24], optimizing surveillance coverage [18][34], extending network lifetime [14][15] and improving system security [4]. However, due to extremely limited resources at each low-cost sensor node (e.g., poor clock quality, limited computation and communication capabilities, ultra tight energy budgets, etc), time synchronization remains as a challenging task in the WSN community.

Previous research in this field mostly focused on *how* to do time synchronization for better clock accuracy (e.g., AD [33], TSS [19], RBS [20], TPSN [21], FTSP [22], VHT [5], etc). In spite of microsecond (μs) level accuracy achieved [5][7][22], they are not designed to provide a precise and convenient trade-off between service performance and energy efficiency, which complicates their deployment for energy sensitive applications with different timing requirements. In practice, systems depend on diverse synchronization qualities. For example, the bridge surveillance project [18] demands tens-of-μs clock accuracy for effective data acquisition; to detect jamming attacks in the network [4], the deviation of a couple of milliseconds (ms) is tolerable; while [1] demonstrated reasonable localization results with $0.5 \sim 7\ ms$ synchronization accuracy. Essentially, high clock accuracy is at the cost of extra energy for communication [22] and measurements [31]. Therefore, we argue that a generic synchronization design in WSN should be application auto-adaptive for the best cost performance.

Realizing the limitations of prior work, this paper proposes *On-Demand Synchronization* (ODS), a design to determine *when* to do time synchronization for customized accuracy. ODS applies the design philosophy of substituting energy intensive wireless communication with local computation, so as to lower the overall cost of the synchronization service. By carefully modeling the error uncertainty of clock skew estimation and its propagation over time, ODS introduces an uncertainty-driven mechanism to adaptively adjust each clock calibration interval individually, instead of traditional periodic synchronization [3][6][20][21][22], for minimum communication overhead while satisfying the desired accuracy. In addition, ODS provides a nice feature of *predictable accuracy*, which enables nodes in the network to acquire the useful information about the qualities (probabilistic confidence) of their synchronization virtually at any time instance. In short, the intellectual contribution of this paper includes:

- To the best of our knowledge, ODS is the first work investigating on-demand synchronization for applications with different or time-varying accuracy requirements.
- We derived closed-form uncertainty analysis for skew and drift estimation, and proved that simply increasing synchronization frequency may not be beneficial at all.
- ODS develops a novel uncertainty-driven mechanism for adaptive clock calibration, which advances an important step towards system efficiency and flexibility.
- The design is evaluated with implementation, measurements and test-bed experiments. To reveal its performance at scale, we also provide a simulation study.

The rest of the paper is organized as follows. Section II explains clock uncertainty and its modeling. The major design of ODS is presented in Section III, and Section IV reports test-bed and simulation results. We overview related work in Section V. And finally, Section VI concludes the whole paper and proposes future research directions inspired by issues discovered in our evaluation experiments.

II. Understanding Clock Uncertainty

In this section, we carefully studied the characteristics of a real clock, including terminologies, empirical data and models that will be adopted by the design part in Section III.

A. Clock Drift and Skew

A clock is simply composed of a periodic signal source (e.g., a quartz oscillator or RC circuit) and a counter register to record the number of periods elapsed [8]. Due to a combination of factors in fabrication, signal sources at different clocks output frequencies with small offsets from the desired value [3][8]. As a result, clocks run at slightly different speeds and accumulate offsets respect to the ideal clock and among themselves, known as the *clock drift* phenomenon.

The slope of change in drift offset is defined as *clock skew* (or *drift rate* in some literatures [9]). To give an example, the skew of clock A respect to a reference clock B, denoted as S_A^B, can be calculated with

$$S_A^B = \frac{\tau_A - \tau_B}{\tau_B} \tag{1}$$

where τ_A is the interval measured by clock A for τ_B elapsed at clock B. $|\tau_A - \tau_B|$ is the drift offset, and if clock A runs faster than clock B, we have $\tau_A > \tau_B \Rightarrow S_A^B > 0$, otherwise, $S_A^B < 0$. Skew is evaluated in ppm (parts per million) and normally ranges from ± 5 ppm to ± 100 ppm [35]. For instance, $S_A^B = 20$ ppm indicates that clock A runs approximately 20 μs faster than clock B for every 1 second.

We carried out initial experiments on MICAz motes to examine the drift behaviors of their 32.768 KHz clocks wildly used for timekeeping [5]. Inspired by RBS [20], a sender node's periodic radio broadcasting was utilized as a global signal for 1-hop receivers that logged time gaps between consecutive packets. The experiment included 1000 transmissions and collected 25809 effective samples from 32 receivers. Fig.1 shows the clock skew (mean value with 3σ confidence range) of each receiver node respect to the sender, telling that nodes have diverse skews within the range of ± 20 ppm.

Another important aspect of a real clock is that it exhibits both long term and short term instabilities [11][39]. In other words, S_A^B in Eq.1 is essentially a function of time $S_A^B(t)$. In addition to natural component aging [35], working conditions such as temperature, power supply, air humility, etc [11][36], and jitter noise [3][11] affect clocks in low-cost embedded devices like Berkley motes. Our data shows that even when the environment was maintained relatively stable (e.g., temp. at 24∼27°C, VCC at 3.37±0.05 V), nodes demonstrated fluctuating skews over time. For example, the top two curves in Fig.3 depict filtered skew values for node 1 and 30 in previous experiment for a period of 15000 seconds, respectively.

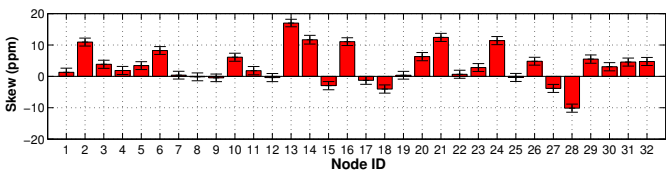


Fig. 1. Skew Measurements for 32 MICAz Nodes

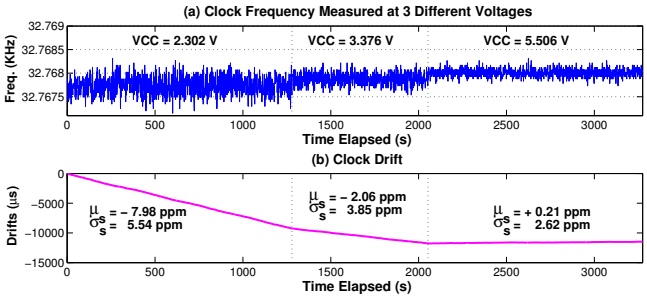


Fig. 2. Clock Frequency and Drift Measurements

More observable, Fig.2 shows the logged frequency of the 32.768 KHz oscillator on a MICAz under different VCC voltages with a Tecktronic DPO 4054 oscilloscope. From Fig.2(a), we can see that the clock frequency slightly increased as VCC went from 2.302 V to 5.506 V. Accordingly in Fig.2(b), the drift offset respect to the ideal clock (the oscilloscope in this case) stopped enlarging after 2000 s, when the skew turned to be positive (average skew $u_s = +0.21$ ppm).

A real clock features random and dynamic skew. However, it is far from formidable and we consider that continuous skew estimation is possible and beneficial for dual reasons:

- First, clock skew varies slowly and is bounded. Today's low-cost oscillators have already provided a low aging factor (e.g., ± 3 ppm/year [35]) and a small temperature coefficient (e.g., ± 0.035 ppm/°C^2 [35]), not to mention TCXO (temperature compensated oscillator). In Fig.2, despite measurement noise, we can observe that the skew difference between 2.302 V and 5.506 V is less than 10 ppm. Note that these two voltages are almost the lower and upper operation limits of the chip [32].
- Second, clock skew solely determines the drift offset that accumulates from instant skew over time, as exampled in Fig.2(b). In other words, if we can detect the clock skew, we are able to estimate and predict potential drift offset, which enables highly efficient time synchronization with predictable accuracy.

To prepare for effective clock offset estimation, skew and drift modeling are firstly discussed in the following.

B. Skew and Drift Modeling

Clock skew varies dynamically as a stochastic process [39]. Among multiple models observed by previous research [3][5][8][38], we employed the WGN (white Gaussian noise) random walk model [8][3] as a tough case example studied in this paper. The uncertainty introduced by this model is statistically higher than the wildly used constant-rate model [38]. Thus, it covers the scenario of networks deployed in outdoor harsh

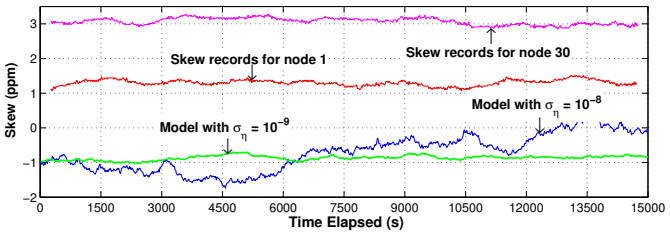


Fig. 3. Clock Skew over Time

environments. Note that the concept of ODS is independent of specific skew model and can be applied with various scenarios.

As a continues-time process, the skew of clock A respect to the reference clock B at time t_0+t, denoted as $S_A^B(t_0+t)$, is modeled to be evolved from that at time t_0 as follows

$$S_A^B(t_0+t) \;=\; S_A^B(t_0)+\int_0^t \eta(u)du \qquad (2)$$

where $\eta(u)\sim\mathcal{N}(0,\sigma_\eta^2)$ and $E[\eta(u)\eta(v)]=\sigma_\eta^2\cdot\delta(u-v)$

$\delta(x)$ is the Kronecker's delta: $\delta(x)=1$, if $x=0$; $\delta(x)=0$, otherwise. t_0 and t_0+t are reference time readings.

The above model takes both intrinsic clock frequency (base skew) and clock instability (random step noise) into consideration. In Eq.2, a bigger σ_η^2 for $\eta(u)$ indicates worse stability of the clock. Note that the value of σ_η^2 is associated with the step size du of the random walk. In fact, when using the ideal clock as reference, σ_η^2 equals $2\cdot AVRA(du)$, where $AVRA(du)$ is the Allan Variance (AVRA) [39], a standard metric for describing clock instability. In practice, the value of σ_η can be obtained from multiple sources: (i) frequency tolerance provided by oscillator specifications [35]; (ii) measurements reported in previous literatures [8][39], and (iii) pre-deployment system profiling [38]. To give an example, Fig.3 plots modeled skew processes as two bottom curves in the figure, where $\sigma_\eta=10^{-8}$ and 10^{-9} were generated for $du=1\ s$.

With clock skew $S_A^B(t)$, the accumulated drift offset t_{drift} between A and B from t_0 to t_1 ($t_1\geq t_0$) can be expressed as

$$t_{drift}=\int_{t_0}^{t_1} S_A^B(t)dt \qquad (3)$$

This drift model is quite general and it subsumes other skew models [7]. For example, for the constant-rate skew model [8] assuming only instant Gaussian noise with $S_A^B(t)$, the offset in Eq.3 degrades to the linear drift model in [10][38].

III. On-Demand Synchronization

In this section, we present the design of ODS that features application-oriented performance and predictable accuracy.

A. Overview

We consider node A as being synchronized to a reference node B with an accuracy of ϵ, if the error associated with A's estimation about the clock offset between itself and B is less than ϵ with a high probability p at any time instance. Note that this definition allows both clock alignment based synchronization (i.e., adjusting clocks to be equally [9][21][17][22]) and timescale transformation among node in the network [9][10]. Under this definition, ODS functions in an *on-demand* manner, by accepting real-time configuration (ϵ,p) from the application layer. For example, a combination of $\epsilon=1\ ms$, $p=99.7\%$ requires the offset estimation error to be less than $1\ ms$ with a confidence probability of at least 99.7%.

The synchronization service is abstracted as two phases: (i) the *detection phase*, during which nodes refresh their time information by exchanging time-stamped messages [21][28]; (ii) the *dormant phase*, as intervals between detection phases. It is easy to understand that nodes obtain the highest synchronization confidence right after each detection phase. However, this confidence of accuracy degrades gradually in the following dormant phase because of the accumulated uncertainty from dynamic clock skew and previous detection errors.

From the above observation, we conclude that the key for achieving time synchronization with desired accuracy while reducing redundant cost is to repeat the detection phase right on time, not too early and not too late. Given a requirement (ϵ,p), ODS adaptively regulates the duration of the following dormant phase after each detection, based on the error uncertainty of skew estimation, so that the required accuracy can be quickly satisfied with minimum traffic overhead.

In the following, we firstly use pairwise synchronization to convey ideas. Then, Section III-E extends the design to the multi-hop scenario. Unless noted otherwise, we denote the reference time at node B as t and the corresponding clock reading at node A as $t_A(t)$; $\hat{x}$ and $\tilde{x}$ are used to express the estimator and error residual for a variable x, respectively.

B. Clock Skew Detection and Estimation

Skew detection serves as the first step towards on-demand synchronization because clock drift is resulted from temporal accumulation of skew. While drift offset can be measured with bounded error uncertainty [2][7][20][22], estimating *instant clock skew* is far more challenging for its time-varying nature. In this section, we propose an instant skew estimator with explicit uncertainty range, by analyzing errors of skew detection with different sampling intervals.

Suppose that at time t_0 node B sent a message to node A about its clock reading, and later node A obtained another timestamp t_1 from node B, as illustrated in Fig.4. As a result, node A can acquire two pieces of information: (i) the time elapsed at the reference, denoted as $\Delta t=t_1-t_0$; and (ii) the drift offset between two nodes during this interval as

$$\hat{t}_{drift}=(t_A(t_1')-t_A(t_0'))-(t_1-t_0) \qquad (4)$$

where $t_A(t_0')$ and $t_A(t_1')$ are sampled clock readings at node A upon receiving B's messages $\boxed{t_0}$ and $\boxed{t_1}$, respectively.

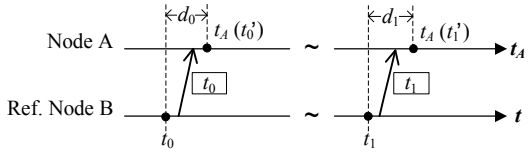


Fig. 4. Uncertainty in Drift Offset Detection

$\hat{t}_{drift}$ is written as an estimator because detections of $t_A(t_0')$ and $t_A(t_1')$ are subject to multiple non-deterministic delays and noise along the "critical path" of communication [12][22]. We consider the additive result of all sources of delays and noise as a random variable d following approximately normal distribution, based on the central limit theory [13] and empirical observations from previous research [22][5], as

$$d\sim\mathcal{N}(\mu_d,\sigma_d^2) \qquad (5)$$

where μ_d and σ_d^2 are determined by hardware/software performance, and can be profiled before network deployment. Then,

$\hat{t}_{drift}$ in Eq.4 can be rewritten as

$$\hat{t}_{drift} = t_{drift} + (d_1 - d_0) \tag{6}$$

where d_1 and d_0 are independent delays for $t_A(t_1')$ and $t_A(t_0')$, respectively. From Eq.5 and 6, we can conclude that $\hat{t}_{drift}$ is an unbiased estimator for t_{drift} with error variance $2\sigma_d^2$.

Based on the offset detection $\hat{t}_{drift}$ for interval Δt, node A can obtain a new estimation about its clock skew respect to the reference node B at t_1 (according to Eq.1) as

$$\hat{S}_A^B(t_1) = \frac{\hat{t}_{drift}}{\Delta t} \tag{7}$$

The error of $\hat{S}_A^B(t_1)$ actually comes from two aspects: (i) the detection error of $\hat{t}_{drift}$; and (ii) the dynamic fluctuation of clock skew during Δt, which is depicted as $S_A^B(t)$ in Fig.5.

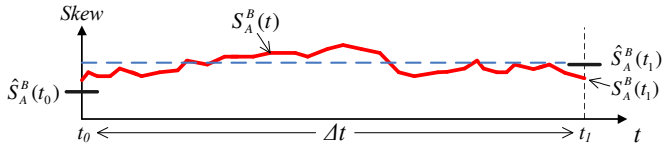


Fig. 5. Error of Skew Detection

The true value of the accumulated drift during the interval Δt equals the area under the solid curve $S_A^B(t)$ in Fig.5. Thus, $\hat{S}_A^B(t_1)$ obtained from Eq.7 essentially estimates the *average value* of $S_A^B(t)$ during Δt, marked as a dashed line. Therefore, $\hat{S}_A^B(t_1)$ could have dynamic offset errors from its true value $S_A^B(t_1)$ as depicted in the figure. We give the following theorem about this estimation.

THEOREM 1. $\hat{S}_A^B(t_1)$ in Eq.7 is statistically unbiased for the stochastic process $S_A^B(t)$, namely

$$E[\hat{S}_A^B(t_1)] = E[S_A^B(t_1)] \tag{8}$$

where $E[\cdots]$ is the expectation. And its error variance is

$$\sigma^2_{\hat{S}_A^B(t_1)} = \frac{2\sigma_d^2}{\Delta t^2} + \frac{\Delta t}{3} \cdot \sigma_\eta^2 \tag{9}$$

where σ_d^2 is the error variance of time detection in Eq.5; σ_η^2 is the step variance of the random walk skew in Eq.2; and Δt is the interval of the last dormant phase.

PROOF. Let $\bar{S}_A^B$ be the true average skew during Δt. $\hat{t}_{drift}$ is an unbiased estimator for t_{drift}, so we have

$$E[\hat{S}_A^B(t_1)] = \frac{E[\hat{t}_{drift}]}{\Delta t} = \frac{t_{drift}}{\Delta t} = \bar{S}_A^B \tag{10}$$

On the other hand, from Eq.2, 3 and 10, we can obtain that

$$\bar{S}_A^B \cdot \Delta t = S_A^B(t_0) \cdot \Delta t + \int_{t_0}^{t_1}\int_{t_0}^{t} \eta(u)dudt \tag{11}$$

Eq.11 and 2 combined give an expression for $S_A^B(t_1)$ as

$$S_A^B(t_1) = \bar{S}_A^B - \frac{1}{\Delta t}\int_{t_0}^{t_1}\int_{t_0}^{t} \eta(u)dudt + \int_{t_0}^{t_1}\eta(u)du \tag{12}$$

which tells $E[S_A^B(t_1)] = \bar{S}_A^B$. So with Eq.10, we have Eq.8.

From Eq.7 and 12, the error residual of $\hat{S}_A^B(t_1)$ is

$$\tilde{S}_A^B(t_1) = S_A^B(t_1) - \hat{S}_A^B(t_1) = \frac{t_{drift} - \hat{t}_{drift}}{\Delta t} + X \tag{13}$$

where $X = \int_{t_0}^{t_1}\eta(u)du - \frac{1}{\Delta t}\int_{t_0}^{t_1}\int_{t_0}^{t}\eta(u)dudt$

Then, its MSE (mean square error) is

$$E[\tilde{S}_A^B(t_1)^2] = \frac{2\sigma_d^2}{\Delta t^2} + E[X^2] \tag{14}$$

becasuse $E[(\tilde{t}_{drift})^2] = 2\sigma_d^2$, $E[X] = 0$, and $\hat{t}_{drift}$, X are independent. For $E[X^2]$, by transformation we can have

$$E[X^2] = E\left[\left(\int_{t_0}^{t_1}\eta(u)du\right)^2\right] + \frac{E\left[\left(\int_{t_0}^{t_1}\int_{t_0}^{t}\eta(u)dudt\right)^2\right]}{(\Delta t)^2} - \frac{2E\left[\int_{t_0}^{t_1}\eta(u)du \cdot \int_{t_0}^{t_1}\int_{t_0}^{t}\eta(u)dudt\right]}{\Delta t} \tag{15}$$

We proved in Appendix A.1 that $E[X^2] = \frac{\Delta t}{3}\cdot\sigma_\eta^2$. Thus,

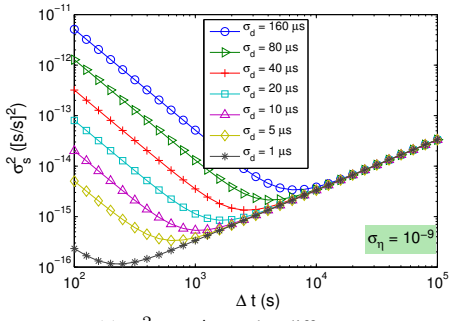
$$\sigma^2_{\tilde{S}_A^B(t_1)} = E[\tilde{S}_A^B(t_1)^2] = \frac{2\sigma_d^2}{\Delta t^2} + \frac{\Delta t}{3}\cdot\sigma_\eta^2$$

This finishes the proof. □

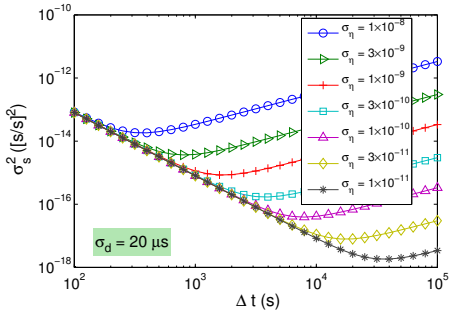
Theorem 1 provides the support and rationale behind Eq.7 for instant skew estimation. It also reveals that the performance of the estimator is determined by three factors:

- Quality of time detection, evaluated by σ_d;
- Stability of the clock, described by σ_η;
- Duration of previous dormant phase, i.e., interval Δt.

Fig.6(a) and 6(b) give example patterns of σ_S^2 against Δt for different values of σ_d and σ_η, respectively. For each curve, σ_S^2 firstly gets reduced along the Δt axis. This is because the first term $2\sigma_d^2/\Delta t^2$ in Eq.9 diminishes quickly with a larger Δt. After reaching a turning point, σ_S^2 enlarges with increasing Δt, since the second term $\Delta t\cdot\sigma_\eta^2/3$ dominates Eq.9 at this stage. From a system perspective, skew fluctuation is

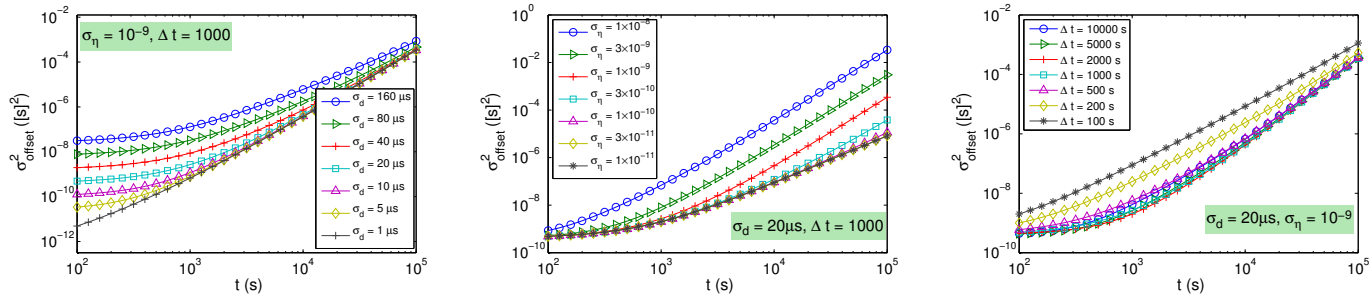


(a) σ_S^2 vs. Δt under different σ_d



(b) σ_S^2 vs. Δt under different σ_η

Fig. 6. The Error Uncertainty for Instant Skew Estimation



(a) $\sigma^2_{\hat{t}_{offset}}$ vs. t under different σ_d (b) $\sigma^2_{\hat{t}_{offset}}$ vs. t under different σ_η (c) $\sigma^2_{\hat{t}_{offset}}$ vs. t under different Δt

Fig. 7. Theoretical Error Uncertainty for Clock Offset Estimation

limited when Δt is small, so the offset detection error becomes relatively large respect to the tiny drift. This explains why σ_d is decisive in the left part of Fig.6(a). On the contrary, if Δt is considerable, the detection error is comparatively negligible, and the estimation performance is mostly affected by clock instability σ_η, as verified by the right part of Fig.6(b).

C. Predicting Clock Uncertainty

With estimated skew, nodes are allowed to predict their drift offsets during the dormant phase. We analyze the uncertainty of such prediction in this section, which paves the way for the key mechanism of uncertainty driven clock calibration.

Based on the detection at t_1, node A can predict its clock offset respect to the reference node B for time t_1+t, denoted as $\hat{t}_{offset}(t_1+t)$, with

$$\hat{t}_{offset}(t_1+t) = \hat{S}^B_A(t_1)\cdot t + \hat{t}_{offset}(t_1) \tag{16}$$

$$\text{where}\quad \hat{t}_{offset}(t_1) = t_A(t_1') - \mu_d - t_1 \tag{17}$$

Here, $\hat{t}_{offset}(t_1)$ is the estimated clock offset between two nodes at time t_1, and μ_d serves as an unbiased estimator for the delay d_1. Eq.16 has been widely used [38][6][3], however, what we are interested in is its error uncertainty.

THEOREM 2. The offset prediction using Eq.16 is unbiased with error variance satisfying

$$\sigma^2_{\hat{t}_{offset}(t_1+t)} = \sigma_d^2 + \frac{2\sigma_d^2}{\Delta t}\cdot t + \sigma^2_{\hat{S}^B_A(t_1)}\cdot t^2 + \frac{\sigma_\eta^2}{3}\cdot t^3 \tag{18}$$

where Δt is the length of previous dormant phase in Eq.7.

PROOF. The true value of $\hat{t}_{offset}(t_1+t)$ in Eq.16 is

$$t_{offset}(t_1+t) = \int_{t_1}^{t_1+t} S^B_A(u)du + (t_A(t_1') - d_1 - t_1) \tag{19}$$

From Eq.19, 16 and Eq.2, its residual can be expressed as

$$\tilde{t}_{offset}(t_1+t) = \tilde{S}^B_A(t_1)\cdot t + \int_{t_1}^{t_1+t}\int_{t_1}^{t_1+u}\eta(v)dvdu - \tilde{d}_1$$

where $\tilde{d}_1 = d_1 - \mu_d$ and $\tilde{S}^B_A(t_1) = \hat{S}^B_A(t_1) - S^B_A(t_1)$. $\tilde{d}_1$ and $\tilde{S}^B_A(t_1)$ are zero mean Gaussian, thus we have

$$E[\tilde{t}_{offset}(t_1+t)] = E\left[\int_{t_1}^{t_1+t}\int_{t_t}^{t_1+u}\eta(v)dvdu\right] = 0 \tag{20}$$

Namely, $\hat{t}_{offset}(t_1+t)$ is an unbiased estimator.

From $\tilde{t}_{offset}(t_1+t)$, we have its error variance as

$$\begin{aligned}\sigma^2_{\hat{t}_{offset}(t_1+t)} &= \sigma^2_{\hat{S}^B_A(t_1)}\cdot t^2 + E[Y^2] + \sigma_d^2 \\ &\quad -2t\cdot E[\tilde{S}^B_A(t_1)\cdot\tilde{d}_1] \\ \text{where}\quad Y &= \int_{t_1}^{t_1+t}\int_{t_1}^{t_1+u}\eta(v)dvdu\end{aligned} \tag{21}$$

It is proved in Appendix A.2 that $E[Y^2] = \frac{\sigma_\eta^2}{3}\cdot t^3$, and $E[\tilde{S}^B_A(t_1)\cdot\tilde{d}_1] = -\frac{\sigma_d^2}{\Delta t}$. As a result, we reach

$$\sigma^2_{\hat{t}_{offset}(t_1+t)} = \sigma_d^2 + \frac{2\sigma_d^2}{\Delta t}\cdot t + \sigma^2_{\hat{S}^B_A(t_1)}\cdot t^2 + \frac{\sigma_\eta^2}{3}\cdot t^3$$

This finishes the proof. □

Insights into Eq.18 tells that σ_d^2 captures errors in original offset detection; $\sigma^2_{\hat{S}^B_A(t_1)}\cdot t^2$ collects skew estimation error over time; $\sigma_\eta^2\cdot t^3/3$ evaluates the uncertainty from dynamic skew fluctuation; and $2\sigma_d^2\cdot t/\Delta t$ includes additional error due to the correlation between offset and skew detections.

Fig.7 shows impacts of σ_d, σ_η and Δt to the offset estimation, respectively. As shown in Fig.7(a), a smaller σ_d brings in less prediction uncertainty, especially when t is small. As t increases, this benefit becomes less significant because the cubic term of t dominates Eq.18 with clock instability σ_η. Fig.7(b) verifies this analysis and indicates that σ_η acts as the deterministic factor as t grows.

Different from σ_d and σ_η, Δt does not have a monotonic impact to the offset prediction. As shown in Fig.7(c), among various Δt values, the lowest curve is given by $\Delta t = 2000$ s. Ironically, a short Δt (e.g., 100 or 200 s) is more harmful than a long Δt (e.g., 5000 or 10000 s), indicating that *calibration at high frequency may not be helpful at all*. This interesting phenomenon is caused by the fact that the error of skew estimation could get *amplified* when Δt is tiny in Eq.9. The "V-shaped" curves in Fig.6 nicely confirms this observation.

Theorem 2 contributes dual benefits to the synchronization service. First, it enables a quantitative evaluation about the quality of synchronization. At any time, a node can not only estimate its clock offset, but also be ware of the error range of this estimation. Namely, a real-time predictable accuracy. In addition, based on this theorem, the synchronization can be conveniently optimized for the desired accuracy performance, as explained in the following.

D. Uncertainty Driven Clock Calibration

To guarantee the required accuracy (ϵ, p), the error in offset estimation should always be less than ϵ with probability p. Namely, $\sigma_{\hat{t}_{offset}}$ must satisfy the following inequality

$$n \cdot \sigma_{\hat{t}_{offset}} \leq \epsilon, \text{ where } n = \sqrt{2} \cdot erf^{-1}(p) \tag{22}$$

n obtained from the inverse Gaussian error function erf^{-1} is the confidence interval (error range) for probability p [27]. For example, if $p = 99.7\%$, $n = 3$ for the 3σ error range.

Clock uncertainty escalates quickly as time elapses in the dormant phase. To meet Eq.22, the dormant phase after the k^{th} detection has a maximum length $T(k)$ that is the solution of Eq.22 and Eq.18 combined as follows

$$f(T) = \left(\frac{\epsilon}{\sqrt{2} \cdot erf^{-1}(p)} \right)^2 \tag{23}$$

$$\text{where } f(T) = \sigma_d^2 + \frac{2\sigma_d^2}{\Delta t} \cdot T + \sigma_{\hat{S}(k)}^2 \cdot T^2 + \frac{\sigma_\eta^2}{3} \cdot T^3$$

In general, synchronization can be accomplished by adopting an uncertainty driven mechanism for clock calibration in Fig.8.

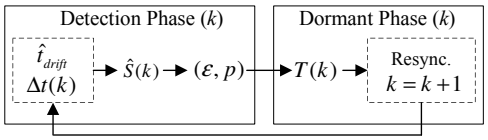


Fig. 8. Uncertainty Driven Clock Calibration

In the k^{th} detection phase, $\hat{t}_{drift}$ and $\Delta t(k)$ contribute a skew estimation $\hat{S}(k)$. Then, for the desired accuracy (ϵ, p), a duration limit $T(k)$ is set for the following dormant phase, and the expiration of $T(k)$ triggers a new round of calibration. Note that a new detection is not necessarily to happen only upon $T(k)$ time out. When the accuracy condition is changed, or data traffic occurs, on which timestamps can be piggybacked with little overhead [28], the dormant phase terminates immediately. In those cases, $\Delta t(k+1) \neq T(k)$.

To summarize, we list operations in detection phase k as Algorithm 1. Line 1 extracts reference time t_k from timestamped message M and samples its local counterpart t'_k. Line 2 clears the timer if it did not fire for $T(k-1)$. Line 3 calculates the drift offset based on Eq.4. Line 4 estimates the instant skew and its error variance from Eq.8 and 9. Then, line 5 gives $T(k)$ based on Eq.23. Finally, line 6 sets a new timer with $T(k)$ and line 7 returns parameters for real-time offset prediction and future calibration. To initialize, $\hat{S}(0)$ is set to be 0 ppm as an unbiased hypothesis, and $\sigma^2_{\hat{S}(0)}$ gets overestimated as S^2_{max} where S_{max} is the maximum skew from the oscillator datasheet [35].

Algorithm 1 ODS Operations at Detection Phase k

Input: (ϵ, p), t_{k-1}, t'_{k-1}, M
Output: $T(k), t_k, t'_k, \hat{S}(k), \sigma^2_{\hat{S}(k)}$
1: $[t_k, t'_k]$ = timeSampling(M);
2: timerClear($T(k-1)$);
3: $[\Delta t(k), \hat{t}_{drift}]$ = preCompute($t_k, t'_k, t_{k-1}, t'_{k-1}$);
4: $[\hat{S}(k), \sigma^2_{\hat{S}(k)}]$ = skewEstimate($\Delta t(k), \hat{t}_{drift}$);
5: $T(k)$ = errorPredict($\sigma^2_{S(k)}$, (ϵ, p));
6: timerSet($T(k)$);
7: return $[t_k, t'_k, \hat{S}(k), \sigma^2_{\hat{S}(k)}]$;

E. Discussion on Multi-Hop ODS

ODS can be conveniently extended for multi-hop synchronization without major theoretical modification. We explain in this section with an example shown in Fig. 9.

Suppose that at time t_R, the reference node disseminated a time-stamped message. An 1-hop node received this message at $t_R + d_1$, and applied Algorithm 1 for clock calibration. After delay τ_1, the 1-hop node took over the synchronization task by sending out newly estimated reference time $\hat{t}_{R(1)}$ together with its error variance $\sigma^2_{\hat{t}_{R(1)}}$ obtained from Theorem 2. Then, the 2-hop node can also apply Algorithm 1 for clock calibration, by simply merging the uncertainty of delay d_2 and that of $\hat{t}_{R(1)}$ as the overall error uncertainty of its reference time detection, i.e., $d'(2) = \sigma_d^2 + \sigma^2_{\hat{t}_{R(1)}}$ in Fig.9. This process repeats till hop n as shown in the figure.

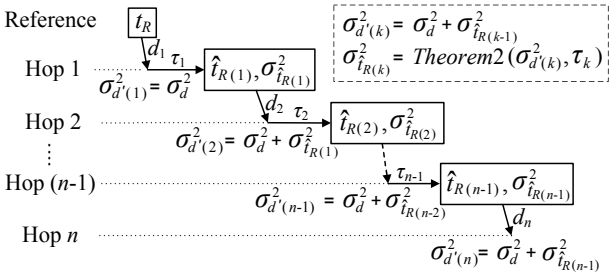


Fig. 9. Error Uncertainty of Multi-hop Reference Time Delivery

In general, we have the accumulated error uncertainty for reference time detection at hop k as

$$\sigma^2_{d'(k)} = \sigma_d^2 + \sigma^2_{\hat{t}_{R(k-1)}} \tag{24}$$

The uncertainty of the reference time sent out at hop k is

$$\sigma^2_{\hat{t}_{R(k)}} = Theorem2(\sigma^2_{d'(k)}, \tau_k) \tag{25}$$

The above two equations reveal that the performance of multi-hop synchronization is affected by (i) the number of hops for reference time delivery, and (ii) delays at each hop.

By forwarding the reference time with its error variance, each node can perform skew estimation and offset prediction as that in the 1-hop scenario by requesting updated reference time from its neighbors when necessary. The feature of predictable accuracy from ODS also allows nodes to effectively fuse time information obtained from different neighbors [30]. On the other hand, multi-hop ODS enables the node as global reference to be aware of the upper bound of clock errors in the network, according to the number of hops and worst-case delays to the most remote nodes. This information provides important guidance for efficient timestamp dissemination.

IV. Evaluation

We implemented ODS on the MICAz mote platform and evaluated with multiple experiments. To reveal its performance at scale, we also report a simulation study in this section.

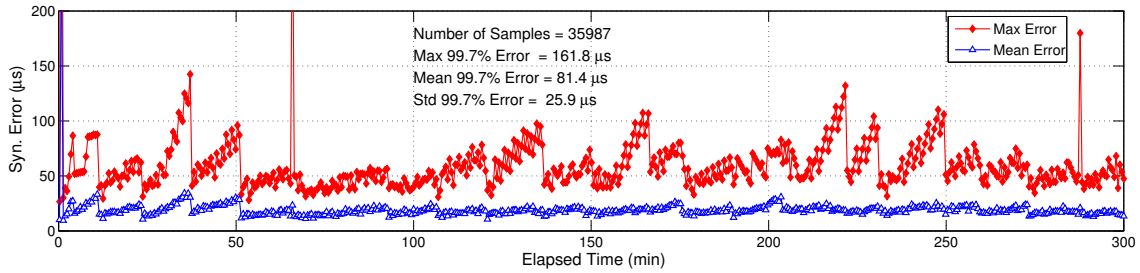


Fig. 10. The Maximum and Mean Synchronization Errors Among 32 MICAz Nodes ($\epsilon = 200\ \mu s$, $p = 99.7\%$, $\sigma_d = 15.26\ \mu s$, $\sigma_\eta = 3 \times 10^{-9}$)

A. Testbed Experiments

The ODS design was developed as a middle-ware module in the TinyOS framework as shown in Fig.11, where arrows indicate data and command flows among modules. To avoid intrinsic delays of TinyOS, low-level instructions were widely used. The kernel ODS task includes 39 lines of NesC code that was complied to 1710 byes of ROM and 68 bytes of RAM.

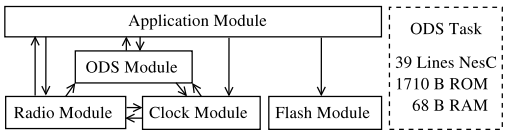


Fig. 11. Implementation of ODS with TinyOS

Considering that multi-hop synchronization heavily depends on assumptions of network topology, routing matrix, traffic load, etc, we focused on 1-hop ODS in the experiment to investigate its actual performance. Note that multi-hop design can be unified with the 1-hop ODS framework equivalently from our discussion in Section III-E.

Error Performance

In this experiment, 32 nodes were deployed on an in-door test-bed to synchronize with a BS, as shown in Fig.12. Nodes targeted an accuracy demand of ϵ = 200 μs, p = 99.7%. We measured $\sigma_d = 15.3\ \mu s$ and configured $\sigma_\eta = 3 \times 10^{-9}$. To get the true real-time synchronization errors, the BS also periodically broadcast a special type of packets that were used by normal nodes only as samples for calculating clock prediction errors logged into their flash memory.

Fig.10 illustrates the maximum and mean errors of 32 nodes during an interval of 5 hours (300 minutes). This figure tells that (i) the maximum errors were kept less than 200 μs all the time, except for initialization and two impulses of outliers; (ii) the max-error curve demonstrated an interesting "ZigZag" pattern along the x-axis, which is reasonable because clock errors escalate statistically over time during the dormant phase while get pushed back upon detections.

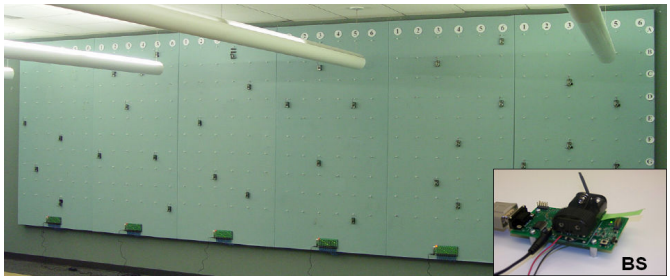


Fig. 12. ODS Evaluation with an Indoor Testbed

As marked in Fig.10, the maximum 99.7%-error range among nodes were 161.8 μs, which is smaller than the required 200 μs, indicating that $\sigma_\eta = 3 \times 10^{-9}$ was actually an overestimation. In addition, a std. of 25.9 μs reveals that nodes had diverse error performance. Both findings suggest that in-field real-time σ_η calibration could be very helpful in practical systems, which we put as part of our future work.

Behaviors under Varying Accuracy

ODS features quick adaption for varying accuracy. In this experiment, we emulated a scenario that at the first stage of about 150 minutes, nodes were required to have a high accuracy (200 μs) for localization purpose; after that, nodes maintain a lower accuracy (500 μs) for energy efficiency.

Fig.13 plots calibration intervals (i.e., the length of dormant phases) of a node in the test. We can see that (i) at the beginning of the first stage, ODS gradually reached an optimal calibration interval for the 200-μs accuracy; (ii) upon accuracy change, ODS swiftly shifted to lengthened calibration intervals, which is attributed to the well estimated clock skew as well as its error uncertainty at the first stage.

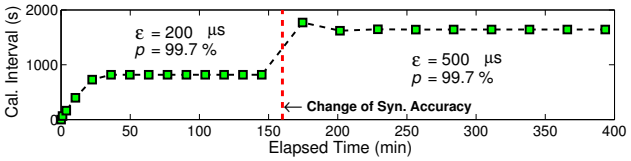


Fig. 13. ODS with Varying Accuracy Requirements

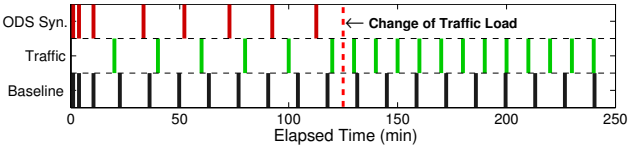


Fig. 14. ODS with Varying Traffic

Behaviors under Varying Traffic

ODS enables full utilization of data traffic in the network, so that dedicated beacon traffic for synchronization can be saved. In this experiment, we emulated two levels of traffic load under identical accuracy requirements, and conducted two tests: (i) ODS without utilizing the data traffic; and (ii) ODS with time-stamp-piggybacked data traffic.

In Fig.14, "Baseline" shows beacon traffic employed in case (i); "Traffic" gives the trace of data traffic; and "ODS Sync." depicts beacon packets in case (ii). We can see that "ODS Sync." used much less beacons than "Baseline". For example, with relatively high traffic load, "ODS Sync." even did not cost ultra packets for synchronization. We further analyzed the impact of traffic in the following with simulations.

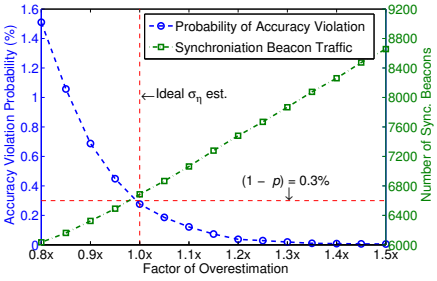


Fig. 15. Impact of σ_η Estimation

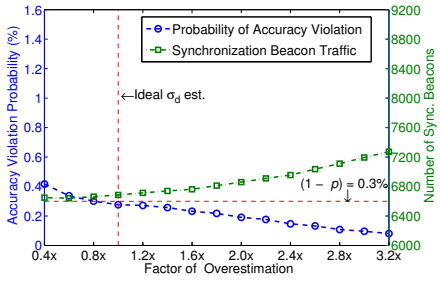


Fig. 16. Impact of σ_d Estimation

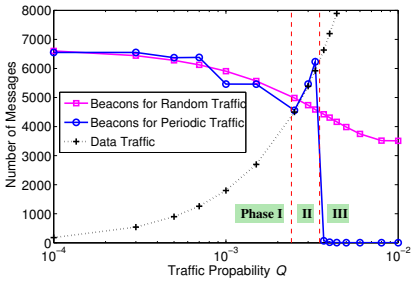


Fig. 17. Impact of Different Traffic

B. Simulation Study

ODS was also evaluated with extensive simulation. Due to space constraints, we only reported results for issues of biased parameter estimation and diverse traffic load in this section.

We developed an event-driven simulator that maintains n separate skew/time frames for systems with n nodes. To satisfy long simulated durations ($\geq 10^8$ s) and precise uncertainty accumulation ($\leq 10^{-24}$) simultaneously, the MAPM library [37] was applied for 64-digit calculation precision. Table I lists default simulation configurations, and all statistics reported are mean values averaged over 50 runs for high confidence.

TABLE I
DEFAULT SIMULATION CONFIGURATIONS

σ_η	σ_d	Accuracy (ϵ, p)	Skew Range	Node Pair	Duration
10^{-9}	15.3 μs	(500 μs , 99.7%)	($\pm$30) ppm	50	5000 hrs

Impact of σ_η Estimation

In this experiment, we investigated the impact of biased σ_η estimation. Fig.15 plots mean values of the probability of accuracy violation (left y-axis) and the number of synchronization beacons (right y-axis) under different factors of biased estimation, in steps of 0.05x. This figure tells that (i) in ideal case (1.0x), the required sub-0.3% violation probability is achieved; (ii) with increasing overestimation, the synchronization accuracy gets improved with reduced violation probability, but the overhead enlarges linearly; (iii) when underestimated, the accuracy performance degrades quickly.

Impact of σ_d Estimation

In this experiment, we investigated the impact of biased σ_d estimation, in steps of 0.2x. Curves in Fig.16 have similar trends as that in Fig.15, except that the impact of biased σ_d estimation is less significant than that of σ_η. By comparing results in Fig.15 and Fig.16, we can conclude that effective calibration for the environmental parameter σ_η plays an important role in practical systems using ODS.

Impact of Different Data Traffic

In this experiment, we investigated the impact of different levels of data traffic. Two types of data traffic were tested

- Random Traffic: data traffic is generated randomly with a probability of Q in every 10 seconds.
- Periodic Traffic: data traffic is generated periodically with a cycle of $10/Q$ seconds.

As shown in Fig.17, by applying ODS, dedicated beacons for clock calibration decrease smoothly with increasing Q values for the random traffic. For periodic traffic, the situation is more complex and deserves further explanation.

To better understand the impact of periodic traffic, we divide Fig.17 along the x-axis into three regions as **Phase I**, **II** and **III**. In phase I, beacons gets reduced slowly with increasing periodic traffic as that for the random traffic. However, more beacons are required with larger Q in phase II. This is because at this stage traffic period is slightly longer than the optimal clock calibration interval, and thus beacons always happen (but be wasted) before another traffic. That's why curves for data traffic and beacons overlap in phase II. Finally, when traffic period is shorter than the calibration interval, no beacon is needed any more. This explains the sharp slope between phase II and III, and a steady close-to-zero pattern in phase III, showing that periodic traffic is extremely helpful for reducing synchronization overhead at this stage with ODS.

V. RELATED WORK

Previous research on time synchronization in WSN mostly focused on enhancing clock accuracy. To combat the non-determinism along the critical path of wireless communication [12], RBS [20] proposed a *receiver-receiver* model to eliminate sender-side delays. While TPSN [21] applied a *sender-receiver* scheme to nullify propagation delay and used MAC layer time-stamping to mitigate other delays. FTSP [22] further advanced clock accuracy by marking multiple timestamps within one message to remove jitters in interrupt handling. Recently, PulseSync [7] reveals that swiftly distributing reference time is critical for multi-hop synchronization, since errors accumulate exponentially with increasing delays. Unlike their focusing on how to deliver reference time, ODS decides when to conduct clock calibration for the desired accuracy.

On the other hand, researchers have also tried to study the fundamentals of clock uncertainty [8][39] for better synchronization accuracy. Many designs assumed constant-rate clock models [9][38] and applied linear regression to estimate clock skew [6][20][22][29]. Considering the unpredictable yet rate-limited variation of clock drift, [7][19] employed bounded-drift models [9] for the worst-cast analysis. Applying the bounded-drift model is convenient but not energy efficient for its overestimated clock instability. On the contrary, [5][36][31] explored precise skew estimation with sensing aided calibration, which unfortunately depends on additional hardware [31] and specific oscillators [5][36]. ACES [3] suggested skew tracking with complex Kalman filtering and period-level sampling adaption [21][29]. ODS differs from aforementioned designs starting from the concept of on-demand accuracy that makes the system more flexible, agile and efficient.

VI. Conclusions and Future Work

In this paper, we present the first work for on-demand time synchronization in wireless sensor networks. By applying an uncertainty-driven mechanism, ODS adaptively adjusts each clock calibration interval for the desired accuracy with minimum communication overhead. Evaluation results demonstrate that ODS is practical, efficient and flexible for different and varying accuracy demands as well as diverse traffic load in the network. It is also suggested that effective in-field σ_η calibration and optimization for periodic traffic are important aspects for further investigation towards optimal system efficiency.

Acknowledgement

Supported by NSF grants CNS-0917097, CNS-0845994, CNS-0720465, InterDigital Inc., and McKnight Land-Grant.

References

[1] G.Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, et al. Sensor Network-based Countersniper System. SenSys '04.
[2] D.L. Mills. Internet Time Synchronization: The network Time Protocol. IEEE Trans. on Communications, 39(10), 1991.
[3] B.R. Hamilton, X.L. Ma, Q. Zhao, J. Xu. ACES: Adaptive Clock Estimation and Synchronization Using Kalman Filtering. MobiCom '08.
[4] M. Pajic and R. Mangharam. Anti-jamming for Embedded Wireless Networks. IPSN '09.
[5] T. Schmid, P. Dutta, M. B. Srivastava. High-Resolution, Low-Power Time Synchronization an Oxymoron No More. IPSN '10.
[6] H.-S. W. So, G. Nguyen, J. Walrand. Practical Synchronization Techniques for Multi-Channel MAC. MobiCom '06.
[7] C. Lenzen, P. Sommer and R. Wattenhofer. Optimal Clock Synchronization in Networks. SenSys '09.
[8] The Science of Timekeeping. Hewlet Packard. Application Note 1289.
[9] K. Römer, P. Blum and L. Meier. Time Synchronization and Calibration in Wireless Sensor Networks. In Handbook of Sensor Networks: Algorithms and Architectures, Wiley & Sons, Hoboken, NJ, 2005.
[10] J. Elson, and K. Römer. Wireless Sensor Networks: A New Regime for Time Synchronization. SigComm Comp. Com. Rev. 33(1), 2003.
[11] Cardinal Components Inc. Clock Oscillator Stability: Measuring Clock Oscillator Frequency Stability. Applications Brief No. A.N. 1006.
[12] H. Kopetz and W. Ochsenreiter. Clock Synchronzation in Distributed Real-Time Systems. IEEE Trans. on Computers, C36(8), 1987.
[13] Henk Tijms. Understanding Probability: Chance Rules in Everyday Life. Cambridge University Press, 2004.
[14] P. Dutta, D. Culler and S. Shenker. Procrastination Might Lead to A Longer and More Useful Lift. HotNets-VI '09.
[15] T. He, P. A. Vicaire, T. Yan, L.Q. Luo, L. Gu, G. Zhou, et al. Achieving Real-Time Target Tracking Using Wireless Sensor Networks. RTAS '06.
[16] J. Elson, and D. Estrin. Time Synchronization for Wireless Sensor Networks. IPDPS '01.
[17] H. Dai, and R. Han. TSync: A Lightweight Bidirectional Time Synchronization Service for Wireless Sensor Networks. ACM SigMobile Mob. Comp. Commun. Rev. 8(1), 2004.
[18] S. Kim, S. Pakzad, D. Culler, J. Demmel, et al. Health Monitoring of Civil Infrastructures using Wireless Sensor Networks. IPSN '07.
[19] K. Rmer. Time Synchronization in Ad hoc Networks. MobiHoc '01.
[20] J. Elson, L. Girod, and D. Estrin. Fine-grained Network Time Synchronization Using Reference Broadcasts. OSDI '02.
[21] S. Ganeriwal, R. Kumar, and M.B. Srivastava. Timing-sync Protocol for Sensor Networks. SenSys '03.
[22] M. Maróti, B. Kusy, G. Simon, and Á. Ldéczi. The Flooding Time Synchronization Protocol. SenSys '04.
[23] D. Revuz and M. Yor. Continuous Martingales and Brownian Motion, 2nd Edition, Springer-Verlag 1994.
[24] Z. Zhong and T. He. MSP: Multi-Sequence Positioning of Wireless Sensor Nodes. SenSys '07.
[25] H. Stark, J. W. Woods, Probability and Random Processes with Applications to Signal Processing, 3rd Edition. Prentice Hall, 2002.
[26] A. El-Hoiyi, J.-D. Decotignie, and J. Hernandez. Low Power MAC Protocols for Infrastructure Wireless Sensor Networks. EW '04.
[27] Feller, W. An Introduction to Probability Theory and Its Applications, V2, 3rd Edition, New York: Wiley, 1971.
[28] W. Ye, F. Silva and J. Heidemann. Ultra-low Duty Cycle MAC with Scheduled Channel Polling. SenSys '06.
[29] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, et al. Estimating Clock Uncertainty for Efficient Duty-cycling in Sensor Networks. SenSys '05.
[30] P. Sommer, R. Wattenhofe. Gradient Clock Synchronization in Wireless Sensor Networks. IPSN '09.
[31] A. Rowe, V. Gupta, R. Rajkumar. Low-power Clock Synchronization using Electromagnetic Energy Radiating from AC Power Lines. SenSys '09.
[32] ATMEL, 8-bit AVR® Microcontroller with 128K Bytes In-System Programmable Flash, ATmega128/128L. Rev. 2467S-AVR-07/09.
[33] Q. Li, D. Rus. Global Clock Synchronization in Sensor Networks. IEEE Trans. on Computers, 55(2), 2006.
[34] M. Ceriotti, L. Mottola, G. Picco, A. Murphy, S. Guna, M. Corra, M. Pozzi, D. Zonta and P. Zanon. Monitoring Heritage Buildings with Wireless Sensor Networks: The Torre Aquila Deployment. IPSN '09.
[35] Product List of Seiko Instruments Inc. (SII), 2009. Online availalbe at http://speed.sii.co.jp/pub/compo/quartz/productListEN.jsp
[36] T. Schmid, D. Torres, M. B. Srivastava. Demo Abstract: Low-power High-precision Timing Hardware for Sensor Networks. SenSys '09.
[37] M. C. Ring. MAPM, A Portable Arbitrary Precision Math Library in C. C/C++ Users Journal. Nov. 2001.
[38] D. Veitch, S. Babu, and A. Pàsztor. Robust Synchronization of Software Clocks Across the Internet. SigComm' 04.
[39] D.W. Allan. Should the Classical Variance Be Used as a Basic Measure in Standards Metrology? IEEE Trans. on I. M., 36, 1987.

Appendix

A.1 We drive Eq.15 in four steps:
(i) for the first term in Eq.15, we have

$$E\left[\left(\int_{t_0}^{t_1}\eta(u)du\right)^2\right]=\int_{t_0}^{t_1}\int_{t_0}^{t_1}E[\eta(u)\eta(v)]dudv=\sigma_\eta^2\cdot\Delta t$$

since $E[\eta(u)\eta(v)]=\delta(u-v)\cdot\sigma_\eta^2$, according to Eq.2.
(ii) for the second term in Eq.15, let $w(m)=\int_{t_0}^{t_0+m}\eta(u)du$, then

$$E\left[\left(\int_{t_0}^{t_1}\int_{t_0}^{t}\eta(u)dudt\right)^2\right]=\int_0^{\Delta t}\int_0^{\Delta t}E[w(m)w(n)]dmdn$$

$w(m)$ is a standard Wiener Process[25], and has a covariance of $E[w(m)w(n)]=min(m,n)\cdot\sigma_\eta^2$ [23]. Considering that

$$\int_0^{\Delta t}\int_0^{\Delta t}min(m,n)\cdot\sigma_\eta^2dmdn=\frac{\Delta t^3}{3}\cdot\sigma_\eta^2 \tag{26}$$

We have that the second term equals $\frac{\Delta t}{3}\cdot\sigma_\eta^2$.
(iii) for the third term in Eq.15, we apply similar substitutions as

$$E\left[\int_{t_0}^{t_1}\eta(u)du\cdot\int_{t_0}^{t_1}\int_{t_0}^{t}\eta(u)dudt\right]=\int_0^{\Delta t}E[w(\Delta t)w(m)]dm$$

where $E[w(\Delta t)w(m)]=m\cdot\sigma_\eta^2$ since $0\le m\le\Delta t$. As a result

$$\int_0^{\Delta t}E[w(\Delta t)w(m)]dm=\int_0^{\Delta t}(m\cdot\sigma_\eta^2)dm=\frac{\Delta t^2}{2}\cdot\sigma_\eta^2 \tag{27}$$

Therefore, the third term has a value of $-\sigma_\eta^2\cdot\Delta t$.
(iv) combining results from (i), (ii) and (iii), we have

$$E[X^2]=\sigma_\eta^2\cdot\Delta t+\frac{\Delta t}{3}\cdot\sigma_\eta^2-\sigma_\eta^2\cdot\Delta t=\frac{\Delta t}{3}\cdot\sigma_\eta^2$$

This finishes the proof. □

A.2 Following the same method in A.1 (ii), we can get

$$E[Y^2]=E\left[\left(\int_{t_1}^{t_1+t}\int_{t_1}^{t_1+u}\eta(v)dvdu\right)^2\right]=\frac{\sigma_\eta^2}{3}\cdot t^3 \tag{28}$$

For $E[\tilde{S}_A^B(t_1)\cdot\tilde{d}_1]$, Eq.6 and 13 give

$$\tilde{S}_A^B(t_1)=\frac{d_0-d_1}{\Delta t}+X \tag{29}$$

$\tilde{d}_1=d_1-\mu_d$, so we have

$$E[\tilde{S}_A^B(t_1)\cdot\tilde{d}_1]=E\left[\left(\frac{d_0-d_1}{\Delta t}+X\right)\cdot(d_1-\mu_d)\right] \tag{30}$$

where X, d_0 and d_1 are independent and $E[X]=0$. Thus, we have

$$E[\tilde{S}_A^B(t_1)\cdot\tilde{d}_1]=E\left[\frac{d_0-d_1}{\Delta t}\cdot(d_1-\mu_d)\right]=-\frac{\sigma_d^2}{\Delta t} \tag{31}$$

because $E[d_1d_0]=\mu_d^2$ and $E[d_1^2]=\mu_d^2+\sigma_d^2$. □