# *PSR*: Practical Synchronous Rendezvous in Low-duty-cycle Wireless Networks

Hao Huang*, Jihoon Yun*, Ziguo Zhong*, Songmin Kim†, and Tian He†
*Computer Science & Engineering, University of Nebraska-Lincoln, Lincoln, NE
†Computer Science, University of Minnesota-TwinCities, Minneapolis, MN
Email: {{hhuang,jyun, zzhong}@cse.unl.edu, {ksong,tianhe}@cs.umn.edu}

***Abstract*—Low-duty-cycle radio operations have been proposed for wireless networks facing severe energy constraints. Despite energy savings, duty-cycling the radio creates transient-available wireless links, making communication rendezvous a challenging task under the practical issue of clock drift. To overcome limitations of prior work, this paper presents *PSR*, a practical design for synchronous rendezvous in low-duty-cycle wireless networks. The key idea behind *PSR* is to extract timing information naturally embedded in the pattern of radio duty-cycling, so that normal traffic in the network can be utilized as a "free" input for drift detection, which helps reduce (or even eliminate) the overhead of traditional time-stamp exchange with dedicated packets or bits. To prevent an overuse of such free information, leading to energy waste, an energy-driven adaptive mechanism is developed for clock calibration to balance between energy efficiency and rendezvous accuracy. *PSR* is evaluated with both test-bed experiments and extensive simulations, by augmenting and comparing with four different MAC protocols. Results show that *PSR* is practical and effective under different levels of traffic load, and can be fused with those MAC protocols to improve their energy efficiency without major change of the original designs.**

## I. Introduction

In wireless networks with severe energy constraints, e.g., wireless sensor networks [1], low-duty-cycle radio operations have been proposed as one of the major techniques to elongate the network lifetime [5], since the radio can be a main source of energy consumption [13]. Basically, the RF module of a node stays active only for a small percentage of time during each duty-cycle period (e.g., 1%), while keeps in low-energy sleep/off mode for the rest of the time [5][28]. Low-duty-cycle radio activity has been favorable in applications such as environment monitoring (e.g., Redwood [3], GreenOrbs [9]), animal observation (e.g., Great Duck Island [10]), civil structure surveillance (e.g., Mine [11]), etc. In all those applications, low-duty-cycle networking provides a nice trade-off between service quality and energy cost; however, it also brings about *transient-available radio links* that are essentially at odd with highly efficient communication. This is because in low-duty-cycle networks, two nodes located within each other's radio range can communicate only when both of them are active simultaneously for transmitting (TX) and receiving (RX) [12]. A problem called communication rendezvous [17].

Many smart ideas have been proposed for the rendezvous task in low-duty-cycle wireless networks. They usually function at the MAC layer and can be categorized into two general classes: (i) asynchronous [5][6][28][17][23][39] and (ii) synchronous [8][22][25][26][27][29]. In asynchronous designs, the sender tries to capture the unknown active time of the receiver, by sacrificing energy [5][6][28], channel efficiency [5][6], or per-hop delay [23][17], which can work well under low traffic load. Synchronous solutions, on the contrary, show improved channel efficiency by controlling and tracking active schedules. Designs in this category usually have to depend on the underlying support of time synchronization [8][22][27] to eliminate negative impacts of clock drift. However, synchronization [4][18][19][20] itself could be costly and difficult in low-duty-cycle wireless networks [15] due to impaired radio channels being lack of broadcasting capability and ultra-tight energy budgets that deny periodic time-stamp exchanges.

To overcome limitations of prior work, this paper presents *PSR*, a practical design for synchronous rendezvous in low-duty-cycle wireless networks. The novelty of *PSR* originates from our observation that the pattern of radio duty-cycling can be used as a time-domain reference for clock drift detection. By extracting such timing information naturally embedded in low-duty-cycle wireless networks, normal traffic can be employed to achieve *0-bit synchronization*, and thus reduces or even eliminates the overhead of traditional time-stamp exchange. To prevent an overuse of such free information, which leads to energy waste, an adaptive mechanism is proposed to balance between energy efficiency and rendezvous accuracy. In short, the intellectual contributions of this paper may include:

- Practical synchronous rendezvous (*PSR*) is proposed for low-duty-cycle wireless networks. As a generic element at the MAC layer, *PSR* can be conveniently fused with many state-of-the-art MAC protocols to improve energy efficiency without major change of the original designs.
- To the best of our knowledge, *PSR* is the first work that enables and implements *clock drift detection with normal traffic in the network*, by extracting timing information naturally embedded in the low-duty-cycle radio pattern.
- An energy-driven adaptive mechanism is developed to make best use of the timing information extracted from normal traffic, allowing *PSR* to work with low, medium and high traffic loads in an on-demand manner.
- *PSR* is implemented and tested with four different MAC protocols from two categorizes using 24 MicaZ motes coupled as 110 different node pairs. To reveal its performance at scale, we also provide an extensive simulation study obeying real-world constraints

In the following, we start with the preliminary information in Section II. Section III gives an overview of key ideas. Then, *PSR* is detailed in Section IV and Section V reports test-bed and simulation evalution results. Finally, Section VII concludes the whole paper.

## II. Preliminary

This section provides basic information on low-duty-cycle radio, communication rendezvous, and clock drift modeling.

### A. Low-duty-cycle Radio Operation

Fig.1 illustrates a typical pattern of low-duty-cycle radio operation [5][6], where node $B$ turns off its RF module most of the time during $T_B$ (node $B$'s duty-cycle period) to conserve energy. Such behavior breaks the traditional always-on radio and creates *transient-available radio links* for accessing this node. As a result, "capturing" the active time slots, denoted as black-filled bars in Fig.1, becomes the precondition for other nodes to communicate with node $B$.

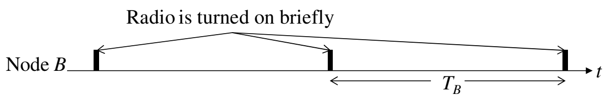


Fig. 1. Pattern of Low-duty-cycle Radio

### B. Communication Rendezvous

In low-duty-cycle wireless networks, communication usually consist two steps: link establishment and data exchange. To capture active time slots for link establishment, two types of methods have been proposed: asynchronous and synchronous rendezvous. Fig.2 show examples for both types of methods.

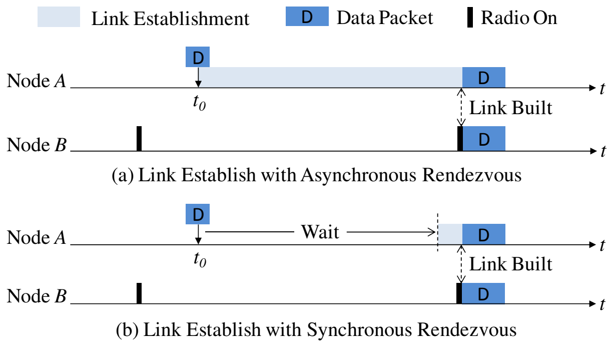

Link Establishment
D Data Packet
Radio On
Node A
D
t_0
Link Built
Node B
t
(a) Link Establish with Asynchronous Rendezvous
Wait
(b) Link Establish with Synchronous Rendezvous

Fig. 2. Link Establish in Low-duty-cycle Networks

In the above Fig.2, node $A$ and $B$ are 1-hop neighboring nodes in a low-duty-cycle wireless network. Consider that node $A$ has a data packet for node $B$ at time $t_0$ as marked in the figure. With asynchronous rendezvous as shown in Fig.2(a), node $A$ does not have any knowledge about node $B$'s active schedule and thus has to turn on its radio ever since $t_0$ and remains active (for TX [5][6] or RX [28][39]) till capturing node $B$'s active signal to establish the radio link. While with synchronous rendezvous as depicted in Fig.2(b), node $A$ has some knowledge about $B$'s active schedule and therefore does not need to turn on its radio until approaching node $B$'s next active time [8][25], resulting in a significantly reduced duration for link establishment, and thus improved energy and channel efficiency.

### C. Clock Drift Modeling

The benefits of synchronous rendezvous essentially come at the cost of synchronization efforts [8][27] for dealing with the imperfectness of clocks that could otherwise break the coordinated communication between node pairs.

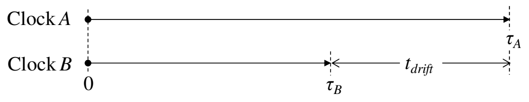


Fig. 3. Clock Drift Offset

In practice, nodes have clocks running at different speeds, resulting in accumulated offsets among them. For example, $\tau_A$ is the interval measured by clock $A$ for $\tau_B$ elapsed at clock $B$ as shown in Fig.3. In this case, $t_{drift} = \tau_A - \tau_B$ is the *clock drift* between $A$ and $B$ during this common interval. The speed difference between clocks is defined as *clock skew*, and the average skew of clock $A$ respective to clock $B$ in this example, denoted as $\bar{S}_A^B$, can be calculated with

$$\bar{S}_A^B = \frac{\tau_A - \tau_B}{\tau_B} \tag{1}$$

A real clock features random and dynamic skews varying as a stochastic process [2][14][29][36][38]. Among multiple skew models [14], we applied the WGN (white Gaussian noise) random walk model [2][14], expressed by Eq.2, as a tough-case example studied in this paper.

$$S_A^B(t_0 + t) \;=\; S_A^B(t_0) + \int_{t_0}^{t_0+t} \eta(u)du \tag{2}$$

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

In Eq.2, $S_A^B(t_0)$ counts for the original speed difference between two clocks at $t_0$; the integration part $\int \eta(u)du$ accumulates real-time environmental impacts during $t$. In practice, the value of $\sigma_\eta$ can be obtained from empirical literatures [33][14][35] or system profiling before network deployments [16][34].

With clock skew $S_A^B(t)$, the drift offset $t_{drift}$ in Fig.3 can be formulated as follows

$$t_{drift} = \bar{S}_A^B \cdot \tau_B = \int_0^{\tau_B} S_A^B(t)dt \tag{3}$$

Note that as the most generic form, the drift model in Eq.3 can work with any clock skew model.

## III. Overview

*PSR* is developed as a supporting component at the MAC layer for efficient synchronous rendezvous in low-duty-cycle wireless networks. The major challenge is to reduce the energy cost for the synchronization service. To address this issue, two unique techniques are proposed: *0-bit clock drift detection* and *energy-driven adaptive skew calibration*, which are briefed in the following before touching details of *PSR* in Section IV.

### A. 0-bit Clock Drift Detection

By leveraging the low-duty-cycle radio pattern as a time-domain reference, *PSR* enables normal traffic in the network for clock drift detection without time-stamp exchanges. We give a simplified example in Fig.4 to illustrate the idea.

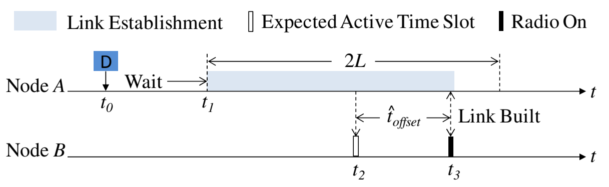


Fig. 4. 0-bit Clock Drift Detection

For the sake of clarity, Fig.4 shows the link establishment with a zoom-in view while omits the following data exchange. Assume that at time $t_0$, node $A$ has a message for node $B$. Based on the previous knowledge of $B$'s schedule, $A$ predicts the next active time of $B$ at $t_2$, and tries to capture it with a window of length $2L$ centered at $t_2$. However, node $B$ wakes up at $t_3$, after which the link is built as shown in Fig.4. In this case, the interval between $t_2$ and $t_3$ is just the drift offset between two nodes, which can be estimated at node $A$ by comparing its original expectation $t_2$ with its detection at $t_3$.

The rationale behind this scheme is that the low-duty-cycle radio pattern of a node carries implicit information about its real-time clock readings and can be used as a time reference equivalent to time-stamps. So, clock drift can be obtained as a by-product during link establishment, and *0-bit drift detection* could be achieved in the presence of normal traffic.

### B. Energy-driven Adaptive Skew Calibration

With 0-bit drift detection, a follow-up question is whether all "free" detections shall be used *equally* for synchronization. The answer is "no" by careful analysis. In *PSR*, time synchronization is treated as two separate operations with diverse energy costs (see Section V). *Drift calibration* can be conducted after each detection for its low-energy profile; while *skew calibration* is launched less frequently for dual reasons: (i) it requires significant calculation efforts, and (ii) updating skew at a high frequency may do harm to synchronization [36]. *PSR* applies an energy-driven mechanism for adaptive skew calibration, the basic principle of which is shown in Fig.5.

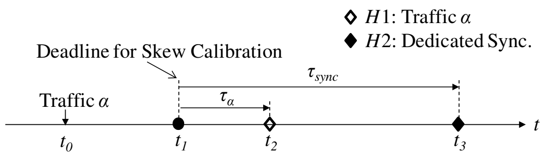


Fig. 5. Traffic for Clock Skew Calibration

Fig.5 illustrates two exclusive options for skew calibration: (i) using the "free" drift detection from traffic $\alpha$ at $t_0$ (choice $H1$), or (ii) applying additional radio operations for dedicated synchronization at $t_1$ that is the deadline for skew calibration (choice $H2$). For $H1$, it can extend the deadline at $t_1$ by $\tau_\alpha$ to $t_2$ as shown in Fig.5, at the cost of $E_{cal}$ that is the energy overhead for skew calculation. For $H2$, it can extend the deadline by $\tau_{sync}$ to $t_3$, however, at the cost of $(E_{com} + E_{cal})$ where $E_{com}$ is the energy cost for additional radio operations.

*PSR* selects between $H1$ and $H2$ based on their cost performance in term of energy efficiency as follows

$$\frac{\tau_\alpha}{E_{cal}} \underset{H2}{\overset{H1}{\gtrless}} \frac{\tau_{sync}}{(E_{com} + E_{cal})} \tag{4}$$

Note that Eq.4 works in an *adaptive* manner because $\tau_\alpha$ is determined by the arrival time of traffic $\alpha$. Generally, the closer between $t_0$ and $t_1$, the larger $\tau_\alpha$ will be, and the more likely to select $H1$. In an extreme case, if traffic $\alpha$ arrives at $t_1$, i.e., $t_0 = t_1$, we would have $\tau_\alpha = \tau_{sync}$, resulting in $H1$ by Eq.4 and by intuition. It will be revealed later that there exists *pivotal points* regarding the arrival time of traffic, based on which a quick decision can be made for skew calibration.

## IV. The *PSR* Design

This section presents the *PSR* design starting with the prediction of active schedules. To obtain the clock skew as a key parameter, we explain the basics on drift detection (0-bit) and skew estimation, followed by an analysis of the estimation uncertainty and its impact on the active schedule prediction. Then, energy-driven clock calibration is introduced. At last, practical issues are discussed, including link initialization, rendezvous failure recovery, and duty-cycle schedule variation. Unless noted otherwise, $\hat{x}$ and $\tilde{x}$ are used to express an estimator (or a detection) and its corresponding error residual, respectively, for the variable with true value $x$.

### A. Active Schedule Prediction

Active schedule prediction serves as the first step towards low-duty-cycle synchronous rendezvous. To fulfill this task, a node requires several pieces of information depicted in Fig.6 as an example where node $A$ and $B$ have diverse active schedules. An important difference between Fig.6 and previous figures is that in Fig.6 two nodes have different timelines: $t_A$ and $t_B$, respectively. This is because *PSR* itself does not demand aligned clocks among nodes in the network.

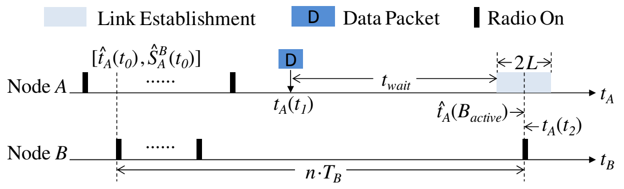


Fig. 6. Predicting the Coming Active Time Slot

Suppose that node $A$ has a data packet for node $B$ at $t_A(t_1)$ (time $t_1$ by node $A$'s clock). To predict $B$'s next active time, which eventually comes at $t_A(t_2)$, node $A$ makes use of four pieces of information marked in Fig.6: (i) its current clock reading $t_A(t_1)$, (ii) the last captured active time of node $B$ $\hat{t}_A(t_0)$ (a detection by node $A$), (iii) its latest skew estimation respect to node $B$, i.e., $\hat{S}_A^B(t_0)$, and (iv) node $B$'s duty-cycle period $T_B$. Then, $A$ obtains its prediction, denoted as $\hat{t}_A(B_{active})$ in Fig.6, with the following Eq.5 and Eq.6.

$$\hat{t}_A(B_{active}) = \hat{t}_A(t_0) + n \cdot T_B \cdot (1 + \hat{S}_A^B(t_0)) \tag{5}$$

$$\text{where } n = \left\lfloor \frac{t_A(t_1) - \hat{t}_A(t_0)}{T_B \cdot (1 + \hat{S}_A^B(t_0))} \right\rfloor + 1 \tag{6}$$

In the above, $\lfloor \cdots \rfloor$ stands for the floor operation, and the term $(1 + \hat{S}_A^B(t_0))$ is used to convert intervals between $t_A$ and $t_B$ based on Eq.1 for the indispensable task of drift compensation. For example, suppose that $n \cdot T_B = 3000$ s (seconds) and the average skew between two nodes is 20 ppm (parts per million) during this interval, a normal value for clocks with embedded devices [14][16][33], then a drift of 60 ms (milliseconds) could get accumulated. Without drift adjustment and if $L < 60$ ms in Fig.6, node $A$ would miss node $B$'s active time slot and fail the tasks of link establishment (rendezvous).

With drift compensation, in theory we shall have an accurate prediction such that $\hat{t}_A(B_{active}) = t_A(t_2)$ as shown in Fig.6. In practice, however, node $A$'s prediction with Eq.5 and 6 suffers from two additional error sources: (i) the detection error of $\hat{t}_A(t_0)$, and (ii) the estimation error of $\hat{S}_A^B(t_0)$ as the average skew. Fortunately, both errors can be modeled, profiled and adjusted, allowing unbiased estimation of active schedules with predictable uncertainty (Section IV-B and IV-C).

Given the active schedule prediction for node $B$, $A$ can launch its link establishment phase (the "capturing window") after waiting for $t_{wait}$ since $t_A(t_1)$ as follows

$$t_{wait} = \hat{t}_A(B_{active}) - t_A(t_1) - L \tag{7}$$

where $L$ as a constant is the radius of node $A$'s capturing window in Fig.6 and can be configured in different applications.

## *B. Drift Detection and Skew Estimation*

Clock skew estimation plays a critical role for drift compensation in schedule prediction (Eq.5 and 6). However, unlike the drift offset which can be measured with bounded error uncertainty [7][20], *instant skew estimation* is challenging for its dynamic nature [14][35]. This section presents an instant skew estimator working with the *0-bit drift detection scheme* briefed in the overview section. For clarity, we use the same example in Fig.6 with slight changes as Fig.7 to convey ideas.

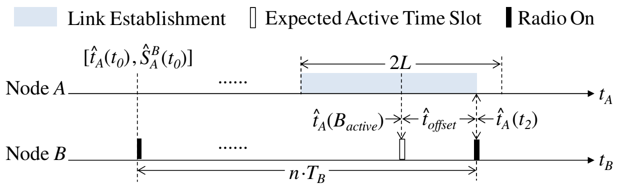


Fig. 7. 0-bit Drift Detection and Skew Estimation

In Fig.7, the error offset between node $A$'s *prior* prediction $\hat{t}_a(B_{active})$ and the corresponding *posterior* detection $\hat{t}_A(t_2)$ for node $B$'s active time can be expressed as

$$\hat{t}_{offset} = \hat{t}_A(t_2) - \hat{t}_A(B_{active}) \tag{8}$$

where time detection $\hat{t}_A(t_2)$ (and $\hat{t}_A(t_0)$) obtained with radio operations is subject to multiple non-deterministic delays and noise along the "critical path" of radio communication [20]. We consider the additive results of all delays and noise as a random variable following the normal distribution based on the central limit theorem [37] and empirical results [7][20]. With delay adjustments, time detections in Fig.7 can be written as

$$\hat{t}_A(t_0) = t_A(t_0) + \phi(t_0),\ \hat{t}_A(t_2) = t_A(t_2) + \phi(t_2) \tag{9}$$

where $\phi(t_0)$ and $\phi(t_2)$ are independent detection noise satisfying $\phi \sim \mathcal{N}(0, \sigma_\phi^2)$. On the other hand, we have

$$t_A(t_2) - t_A(t_0) = n \cdot T_B \cdot \left(1 + \bar{S}_A^B\right) \tag{10}$$

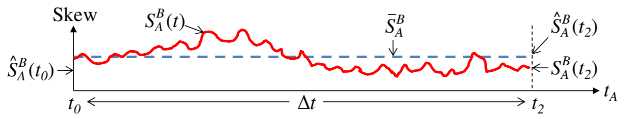


Fig. 8. Example Clock Skew as A Stochastic Process

where $\bar{S}_A^B$ is the true average clock skew of node $A$ respect to node $B$ during the interval from $t_A(t_0)$ to $t_A(t_2)$. By integrating Eq.5, 8, 9 and 10, we can rewrite $\hat{t}_{offset}$ as

$$\hat{t}_{offset} = n \cdot T_B \cdot (\bar{S}_A^B - \hat{S}_A^B(t_0)) + \phi(t_2) - \phi(t_0) \tag{11}$$

which tells that this offset comes from two physical components: (i) skew estimation error $(\bar{S}_A^B - \hat{S}_A^B(t_0))$ and (ii) active time detection errors $\phi(t_2)$ and $\phi(t_0)$.

Given $\hat{t}_{offset}$ from Eq.8 and based on Eq.1, node $A$ can update its skew estimation at $t_A(t_2)$ as

$$\hat{S}_A^B(t_2) = \hat{S}_A^B(t_0) + \frac{\hat{t}_{offset}}{\Delta t},\ \text{where } \Delta t = n \cdot T_B \tag{12}$$

The true skew of node $A$ respect to node $B$ varies along the time line as exampled in Fig.8. $\hat{S}_A^B(t_2)$ obtained with Eq.12 is actually an estimation of the average skew $\bar{S}_A^B$ during $\Delta t$, marked as the thick dashed line in Fig.8. $\hat{S}_A^B(t_2)$ may have random offsets from $S_A^B(t_2)$ that is the true instant clock skew at $t_A(t_2)$. However, we can comfortably apply $\hat{S}_A^B(t_2)$ for future schedule prediction based on the following theorem.

THEOREM 1. $\hat{S}_A^B(t_2)$ from Eq.12 is an unbiased estimator for $\bar{S}_A^B$ during $\Delta t$ and for the instant skew $S_A^B(t_2)$, namely,

$$E[\hat{S}_A^B(t_2)] = \bar{S}_A^B = E[S_A^B(t_2)] \tag{13}$$

with an error variance (respect to its true value $S_A^B(t_2)$) of

$$\sigma^2_{\hat{S}_A^B(t_2)} = \frac{2\sigma_\phi^2}{\Delta t^2} + \frac{\sigma_\eta^2}{3} \cdot \Delta t \tag{14}$$

where $\sigma_\phi^2$ is the error variance of active slot detection and $\sigma_\eta^2$ comes from the skew model in Eq.2.

PROOF 1. With Eq.11, Eq.12 and $\phi \sim \mathcal{N}(0, \sigma_\phi^2)$, we have

$$E[\hat{S}_A^B(t_2)] = E[\bar{S}_A^B] + \frac{E[\phi(t_2) - \phi(t_0)]}{\Delta t} = \bar{S}_A^B \tag{15}$$

Based on Eq.2 and Eq.3, we can get

$$\bar{S}_A^B = S_A^B(t_0) + \frac{1}{\Delta t}\int_{t_0}^{t_2}\int_{t_0}^{t}\eta(u)dudt \tag{16}$$

Combine Eq.16 with $S_A^B(t_2) = S_A^B(t_0) + \int_{t_0}^{t_2}\eta(u)du$ (Eq.2),

$$S_A^B(t_2) = \bar{S}_A^B - \frac{1}{\Delta t}\int_{t_0}^{t_2}\int_{t_0}^{t}\eta(u)dudt + \int_{t_0}^{t_2}\eta(u)du \tag{17}$$

which tells $E[S_A^B(t_2)] = \bar{S}_A^B = E[\hat{S}_A^B(t_2)]$ with Eq.15.

From Eq.11, 12 and 17, $\tilde{S}_A^B(t_2) = S_A^B(t_2) - \hat{S}_A^B(t_2)$ equals

$$\tilde{S}_A^B(t_2) = \int_{t_0}^{t_2}\eta(u)du - \frac{1}{\Delta t}\int_{t_0}^{t_2}\int_{t_0}^{t}\eta(u)dudt - \left(\phi(t_2) - \phi(t_0)\right)/\Delta t \tag{18}$$

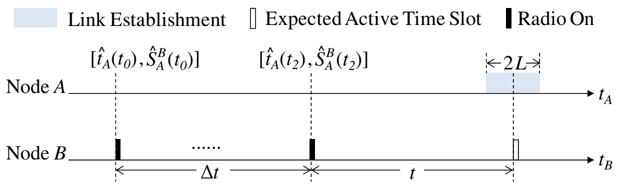
For the above expression, we prove in Appendix A.1 that

$$\sigma^2_{\hat{S}_A^B(t_2)} = E[(\tilde{S}_A^B(t_2))^2] = \frac{2\sigma_\phi^2}{\Delta t^2} + \frac{\sigma_\eta^2}{3} \cdot \Delta t \tag{19}$$

which converges to Eq.14 and finishes the proof. □

## C. Bounding the Error of Schedule Prediction

To guarantee the capture of active time slots of the desired receiver, the error associated with schedule prediction must be smaller than the radius of the capture window $L$ with a high probability. We apply detections at $t_A(t_2)$ in previous example as a starting point to explain the error uncertainty of future schedule prediction as illustrated in Fig.9



Link Establishment
Expected Active Time Slot
Radio On
$[\hat{t}_A(t_0), \hat{S}_A^B(t_0)]$
$[\hat{t}_A(t_2), \hat{S}_A^B(t_2)]$
$2L$
Node $A$
Node $B$
$t_A$
$t_B$
......
$\Delta t$
$t$

Fig. 9. Future Active Schedule Prediction

For the example in Fig.9, based on Eq.5 the expected active time of node $B$ after $t$ since $t_A(t_2)$ can be expressed as

$$\hat{t}_A(B_{active}) = \hat{t}_A^B(t_2) + t \cdot (1 + \hat{S}_A^B(t_2)) \quad (20)$$

where $t$ is an integer number of $T_B$ and $\hat{S}_A^B(t_2)$ comes from the skew updating at $t_A^B(t_2)$ with Eq.12. The error variance of $\hat{t}_A(B_{active})$ in Eq.20 can be described as follows.

THEOREM 2. The prediction with Eq.20 is unbiased with an error variance of

$$\sigma^2_{\hat{t}_A(B_{active})} = \sigma^2_\phi + \frac{2\sigma^2_\phi}{\Delta t} \cdot t + \sigma^2_{\hat{S}_A^B(t_2)} \cdot t^2 + \frac{\sigma^2_\eta}{3} \cdot t^3 \quad (21)$$

where $\Delta t$ is the interval for the latest skew detection in Fig.9.

PROOF 2. Combining Eq.9, Eq.10 and Eq.20, the error residual $\tilde{t}_A(B_{active}) = t_A(B_{active}) - \hat{t}_A(B_{active})$ can be

$$\tilde{t}_A(B_{active}) = t \cdot (\bar{S}_A^B - \hat{S}_A^B(t_2)) - \phi(t_2) \quad (22)$$

where $\bar{S}_A^B$ in this case is the true average skew between node $A$ and $B$ during $t$. Similar to Eq.16, $\bar{S}_A^B$ can be expressed as

$$\bar{S}_A^B = S_A^B(t_2) + \frac{1}{t}\int_{t_2}^{t_2+t}\int_{t_2}^{t_2+v} \eta(u)dudv \quad (23)$$

then, Eq.22 turns into

$$\tilde{t}_A(B_{active}) = t \cdot \tilde{S}_A^B(t_2) + \int_{t_2}^{t_2+t}\int_{t_2}^{t_2+v} \eta(u)dudv - \phi(t_2)$$

where $\hat{S}_A^B(t_2)$, $\phi(t_2)$, and $\eta(u)$ are all zero-mean Gaussian, therefore $E[\tilde{t}_A(B_{active})] = 0$, i.e., $\hat{t}_A(B_{active})$ is unbiased.

For the above $\tilde{t}_A(B_{active})$, we prove in Appendix A.2 that

$$E[\tilde{t}_A(B_{active})^2] = \sigma^2_\phi + \frac{2\sigma^2_\phi}{\Delta t} \cdot t + \sigma^2_{\hat{S}_A^B(t_2)} \cdot t^2 + \frac{\sigma^2_\eta}{3} \cdot t^3$$

which converges to Eq.21 and finishes the proof. □

Theorem 2 enables quantitative evaluation of the error uncertainty associated with future schedule predication, based on which *PSR* adaptively sets a deadline $t_{sync}$ (an interval $\tau_{sync}$ into the future) for the next skew calibration, immediately after the current skew calibration by solving

$$3\sigma_{\hat{t}_A(B_{active})} = L \quad (24)$$

where $\sigma_{\hat{t}_A(B_{active})}$ is a function of $\tau_{sync}$ in Eq.21 in this case. Eq.24 assures that in theory a "capture window" of radius $L$ can catch any active time slots before the deadline $t_{sync}$ at the probability of at least 99.7% (i.e., the $3\sigma$ confidence range).

## D. Energy-Driven Adaptive Clock Calibration

With 0-bit drift detection, normal traffic in the network can be utilized for clock calibration, including *drift calibration* and *skew calibration*. For drift calibration, a node can simply update its record for the receiver's active schedule with the new detection, e.g., $\hat{t}_A(t_2)$ in Fig.9, which incurs little cost. While for skew calibration, besides skew updating with Eq.12, the deadline $T_{sync}$ for future skew calibration also needs to be refreshed by solving the equation listed as Eq.24, which could demand considerable computation efforts. Realizing the different energy costs, *PSR* develops diverse mechanisms for offset and skew calibration, respectively.

*1) Immediate Drift Calibration: A node updates its record of the receiver's active schedule immediately after obtaining the "free" offset information from each traffic in the network.*

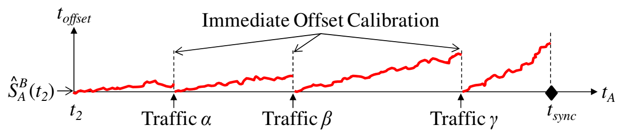


Fig. 10. Immediate Offset Calibration

Fig.10 illustrates an example of immediate offset calibration, where the deadline $t_{sync}$ is set after skew estimation at $t_2$. The operation of immediate offset calibration can reset the error of schedule prediction, denoted as the $y$-axis $t_{offset}$ in Fig.10, upon traffic $\alpha$, $\beta$ and $\gamma$. This is because a new and accurate active time detection can be obtained with each traffic. As a result, $t_{sync}$ set at $t_2$ becomes a conservative deadline.

Note that the slope of offset accumulation enlarges after each traffic as depicted in Fig.10. This is actually not because of the traffic but the nature of dynamic clock skew variation. Without skew calibration, the confidence of $\hat{S}_A^B(t_2)$ employed as the expected average skew for schedule prediction decreases quickly as time elapses, the uncertainty of which is essentially evaluated by the term $(\sigma^2_\eta \cdot t^3/3)$ in Eq.21. Therefore, it is clear that such immediate drift calibration alone is not sufficient for sustaining the synchronous rendezvous, especially for networks without extra-high traffic load.

*2) Energy-driven Skew Calibration: A node conducts skew calibration based on the "free" drift detection form normal traffic only when it is more energy efficient than not doing so.*

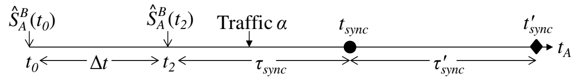


Fig. 11. Not Using Traffic for Skew Calibration

Fig.11 depicts the situation of not using the "free" drift detection from normal traffic for skew calibration, in which skew recalibration has to be conducted with additional radio operations at $t_{sync}$, extending the deadline by $\tau'_{sync}$ to $t'_{sync}$. Note that $\tau'_{sync}$ is obtained by solving the same equation in Eq.24, and in this case $\Delta t$ in Eq.21 and Eq.14 becomes $\tau_{sync}$. Such option in Fig.11 is named as $H2$ that contributes a synchronous interval $\tau_{sync'}$ at the cost of $(E_{cal} + E_{com})$ where $E_{cal}$ and $E_{com}$ are the energy overhead for the radio operation and computation (equation solving), respectively.

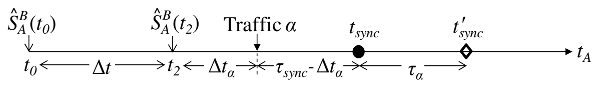


Fig. 12. Using Traffic for Skew Calibration

Fig.12 shows the opposite situation of using normal traffic for skew calibration, in which skew recalibration can also be conducted at $t_{sync}$ but with the "free" drift detection recorded from traffic $\alpha$, extending the deadline by $\tau_\alpha$ to $t'_{sync}$. In theory, skew calibration with traffic $\alpha$ contributes a synchronous interval of $(\tau_{sync} - \Delta t_\alpha) + \tau_\alpha$ in Fig.12, by solving Eq.24, Eq.21 and Eq.14. However, $(\tau_{sync} - \Delta t_\alpha)$ can not be counted as "new", because it is an duration before the deadline $t_{sync}$. Therefore, the option of using traffic for skew calibration, denoted as $H1$, contributes $\tau_\alpha$ at the cost of $E_{cal}$.

Based on the above analysis, we can choose $H1$ or $H2$ by comparing their energy efficiency as follows

$$\frac{\tau_\alpha}{E_{cal}} \underset{H2}{\overset{H1}{\gtrless}} \frac{\tau'_{sync}}{(E_{com} + E_{cal})} \tag{25}$$

Eq.25 is simple and conceptually elegant, however it can hardly be used directly in practice, because neither $\tau_\alpha$ nor $\tau_{sync}$ is available before conducting the corresponding skew calibration. *PSR* overcomes this challenging dilemma by identifying *pivotal points* computed off-line based on Eq.25 and potential network settings, so that a node can select $H1$ or $H2$ simply depending on whether the traffic occurs after or before the pivotal point. For example, $\Delta t_\alpha$ in Fig.12 determines the use of traffic $\alpha$ or not for skew calibration.

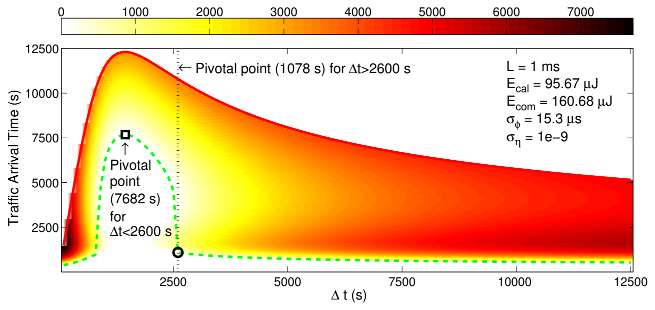


Fig. 13. Example Pivotal Point Profiling

Fig.13 shows results of off-line pivotal point profiling for an example set of parameters used in our test-bed experiments ($L$, $E_{cal}$, $E_{com}$, $\sigma_\phi$, $\sigma_\eta$). The $x$-axis shows $\Delta t$ in Fig.11. The envelop of the pattern depicted as the red-solid curve gives the deadline $t_{sync}$ for different $\Delta t$, which is also the possible range of traffic arrival time, since $\Delta t_\alpha$ in Fig.11 can not be larger than $t_{sync}$. The green-dashed curve gives the lower bound of $\Delta t_\alpha$ in Fig.11 for selecting $H1$, above which using the traffic for skew calibration is always more energy efficient. The color density (or darkness) in Fig.13 indicates additional deadline extensions comparing with the threshold value given by the lower bound. For this example, two pivotal points are selected: if $\Delta t < 2600$, the pivotal point is set as 7682 s (the black square), otherwise 1078 s (the black circle), as marked in Fig.13. Note that both pivotal points are set in a very conservative manner for the sake of simplicity, and such pivotal points can always be set with different systems.

**Algorithm 1**: PSR Adaptive Clock Calibration

```
   input : [$\hat{t}_{offset}, \Delta t_\alpha$], $\Delta t$, $\hat{S}(k-1)$, $t_{sync}(k-1)$
   output: $\hat{t}_A(B_{active})$, $\hat{S}(k)$, $t_{sync}(k)$
 1 $\hat{t}_A(B_{active})$ ← driftCal($\hat{t}_{offset}$);
 2 if $\Delta t_\alpha$ exceeds the pivotal point for $\Delta t$ then
 3     sample ← detectionRec([$\hat{t}_{offset}, \Delta t_\alpha$]);
 4 end
 5 if reach deadline $t_{sync}(k-1)$ then
 6     if sample = ∅ then
 7         sample ← radioAct($\hat{t}_A(B_{active})$, $\hat{S}(k-1)$);
 8     end
 9     [$\hat{S}(k)$, $t_{sync}(k)$] ← skewEst(sample, $\hat{S}(k-1)$);
10 end
11 return [$\hat{t}_A(B_{active})$, $\hat{S}(k)$, $t_{sync}(k)$];
```

To summarize, we list major operations for clock calibration as Algorithm 1 triggered upon traffic or at the calibration deadline. Line 1 conducts immediate drift calibration with each traffic. Line 2 to 4 performs the pivotal point test to determine whether the current traffic can be used for skew calibration later. Line 5 to 10 describes tasks at the calibration deadline, including skew and deadline updating with the latest traffic sample or additional radio actions. Skew calibration with the latest traffic may not be optimal, but good enough as a heuristic solution featuring little cost (see Fig.13). Finally, line 11 returns results for future active schedule prediction.

### E. Discussion on Practical Issues

In practical systems, rendezvous failure happens because of various reasons, including (i) schedule prediction error that occurs with expected marginal probability, (ii) poor radio link quality, (iii) uninformed duty-cycle schedule variation, (iv) node malfunction, etc. For failures caused by (i), (ii) and (iii), *PSR* degrades to the asynchronous rendezvous method for reestablishing the line as a special case of system initialization.

*PSR* launches normal system initialization under two circumstances: (i) a new node joins the network; and (ii) lost neighbor is declared. In both situations, a long link establishment phase is used by either the new node or the sender node experiencing neighbor lost, the length of which equals the maximum possible period as that in B-MAC [5], X-MAC [6] or RI-MAC [28] so as to guarantee active schedule capturing.

## V. Evaluation

We implemented the *PSR* design as an non-intrusive supporting layer with four low-duty-cycle MAC protocols (i.e., X-MAC [6], RI-MAC [28], Wise-MAC [25], and CSMA-MPS [26]) as illustrated in Fig.15, and evaluated their performance with test-bed experiments (using 24 MicaZ nodes coupled as 110 different node pairs) as well as an extensive simulation study obeying real world conditions.

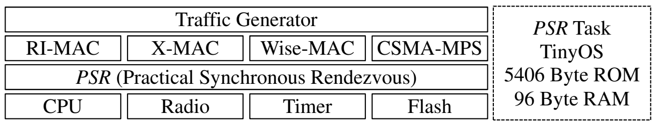


Fig. 15. Implementation of *PSR* with TinyOS

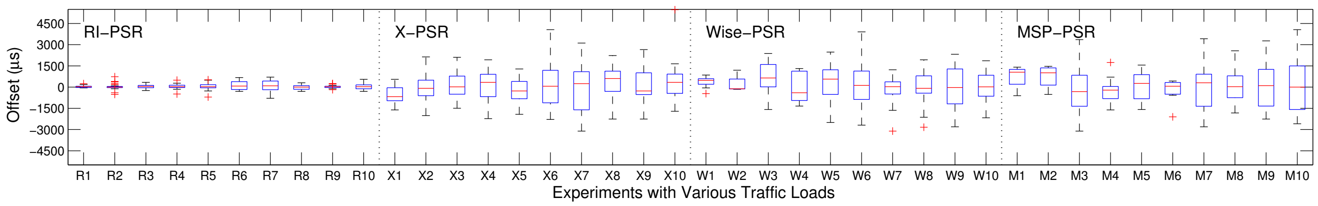


Fig. 14. Rendezvous Offset Measurements of RI-PSR ($L = 2ms$,$\sigma_\phi = 15.03\mu s$), X-PSR, Wise-PSR, and MPS-PSR ($L = 7.5ms$, $\sigma_\phi = 1ms$)

**Table II.** RI-MAC vs. RI-PSR: Average Energy Cost Per Rendezvous

| **Traffic Density** $Q$ (in *min*) | **15′** | **30′** | **45′** | **60′** | **75′** | **90′** | **105′** | **120′** | **135′** | **150′** |
|---|---|---|---|---|---|---|---|---|---|---|
| RI-MAC (in $mJ$) | 40.447 | 34.743 | 37.285 | 46.946 | 34.348 | 36.713 | 47.752 | 38.377 | 30.106 | 43.187 |
| RI-PSR (in $mJ$) | 0.190 | 0.243 | 0.279 | 0.349 | 0.394 | 0.428 | 0.477 | 0.502 | 0.552 | 0.655 |
| Efficiency Improvement (in $X$) | 213.447 | 143.780 | 134.691 | 135.442 | 88.110 | 86.756 | 101.171 | 77.366 | 55.512 | 66.903 |

**Table III.** X-MAC vs. X-PSR: Average Energy Cost Per Rendezvous

| **Traffic Density** $Q$ (in *min*) | **15′** | **30′** | **45′** | **60′** | **75′** | **90′** | **105′** | **120′** | **135′** | **150′** |
|---|---|---|---|---|---|---|---|---|---|---|
| X-MAC (in $mJ$) | 34.990 | 37.136 | 49.281 | 33.300 | 42.629 | 39.661 | 37.272 | 40.839 | 40.414 | 41.484 |
| X-PSR (in $mJ$) | 0.776 | 0.867 | 0.959 | 0.990 | 1.025 | 1.200 | 1.267 | 1.318 | 1.385 | 1.354 |
| Efficiency Improvement (in $X$) | 46.076 | 43.826 | 52.389 | 34.636 | 42.565 | 34.050 | 30.412 | 31.983 | 30.178 | 31.631 |

**Table IV.** Wise-MAC vs. Wise-PSR: Average Energy Cost Per Rendezvous

| **Traffic Density** $Q$ (in *min*) | **15′** | **30′** | **45′** | **60′** | **75′** | **90′** | **105′** | **120′** | **135′** | **150′** |
|---|---|---|---|---|---|---|---|---|---|---|
| Wise-MAC (in $mJ$) | 10.176 | 21.436 | 33.121 | 35.251 | 45.573 | 50.621 | 52.760 | 64.819 | 67.470 | 56.056 |
| Wise-PSR (in $mJ$) | 1.961 | 2.100 | 2.115 | 2.582 | 2.593 | 2.999 | 3.193 | 4.155 | 3.651 | 3.398 |
| Efficiency Improvement (in $X$) | 6.188 | 11.206 | 16.659 | 14.653 | 18.577 | 17.880 | 17.525 | 16.599 | 19.478 | 17.494 |

**Table V.** CSMA-MPS vs. MPS-PSR: Average Energy Cost Per Rendezvous

| **Traffic Density** $Q$ **(in** ***min*****)** | **15′** | **30′** | **45′** | **60′** | **75′** | **90′** | **105′** | **120′** | **135′** | **150′** |
|---|---|---|---|---|---|---|---|---|---|---|
| CSMA-MPS (in $mJ$) | 7.509 | 13.499 | 13.368 | 18.026 | 16.742 | 22.657 | 30.040 | 24.741 | 22.111 | 27.549 |
| MPS-PSR (in $mJ$) | 0.768 | 0.954 | 0.892 | 1.000 | 1.200 | 1.189 | 1.481 | 1.318 | 1.345 | 1.357 |
| Efficiency Improvement (in $X$) | 10.780 | 15.143 | 15.993 | 19.021 | 14.952 | 20.061 | 21.284 | 19.770 | 17.443 | 21.296 |

## *A. Test-bed Evaluation*

We implemented *PSR* with four MAC protocols: X-MAC, RI-MAC, Wise-MAC, and CSMA-MPS, considering that X-MAC and RI-MAC are typical asynchronous rendezvous solutions while Wise-MAC and CSMA-MPS are typical synchronous rendezvous methods. Their *PSR*-augmented versions are denoted as X-PSR, RI-PSR, Wise-PSR, and MPS-PSR, respectively. For each protocol, 20 (for RI-MAC) to 30 (for X-MAC, Wise-MAC, and CSMA-MSP) different MicaZ node pairs are tested and each experiment lasted for at least 24 hours so as to cover one cycle of daily environment variation. Table I lists basic system configurations, where the expected energy parameters $E_{cal}$ and $E_{com}$ are obtained from system measurements and component data sheets [30][32], details of which are provided with each experiment in the following.

**Table I.** Implementation Configurations

| **Protocol** | $\sigma_\phi$, $\sigma_\eta$ , $L$, $E_{cal}$, $E_{com}$ | **Sample Size** |
|---|---|---|
| RI-PSR | $15.3\mu s$, $10^{-9}$, $1ms$, $95.76\mu J$, $160.68\mu J$ | 292 |
| X-PSR | $1ms$, $10^{-9}$, $7.5ms$, $95.76\mu J$, $743.28\mu J$ | 454 |
| Wise-PSR | $1ms$, $10^{-9}$, $7.5ms$, $95.76\mu J$, $1896.93\mu J$ | 399 |
| MPS-PSR | $1ms$, $10^{-9}$, $7.5ms$, $95.76\mu J$, $743.28\mu J$ | 398 |

### RI-MAC vs. RI-PSR

RI-MAC [28] is a receiver initiated asynchronous MAC design. When a node needs to send packets, it keeps listening for incoming beacons from the desired receiver to build the link. The expected duration for link establishment in RI-MAC is about half of the receiver's duty-cycle period. In its *PSR*-augmented version RI-PSR, the maximum RX time of the sender for each rendezvous is only about $3ms$ in our implementation, which is the summation of capture window width ($2L = 2ms$) and the duration for beacon receiving ($1ms$). The corresponding $E_{com}$ is calculated as $160.68\mu J$.

Table II compares average energy costs per rendezvous between RI-MAC and RI-PSR. The traffic is generated following a uniform random distribution with the density of 1 packet per $Q$ minutes (min) as a variable listed in the first row. Throughout our experiments, RI-PSR performs at least 50 times more energy efficient than RI-MAC as shown in the bottom row of the table, where $X$ stands for "times" obtained by dividing the energy cost of RI-MAC by that of RI-PSR. Under high traffic density, for example 1 packet per 15 min, RI-PSR is about $213X$ more energy efficient. While as the traffic load decreases, the synchronization cost increase for RI-PSR because less traffic can be used for "free" drift detection, leading to reduced efficiency improvements.

Fig.14 shows boxplots of offsets at recalibration deadlines for all methods. The left-most part for RI-PSR tells that its offsets are smaller than the error bound $L = 1ms$, which is because (i) a tough-case skew model is used for triggering the recalibration, resulting in reduced error offsets; and (ii) normal traffic is utilized by *PSR* for immediate offset calibration, contributing to smaller synchronization errors.

### X-MAC vs. X-PSR

X-MAC [6] is a typical sender initiated asynchronous MAC

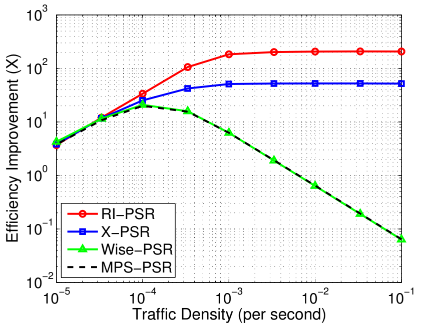


Fig. 16. Impact of Traffic Load

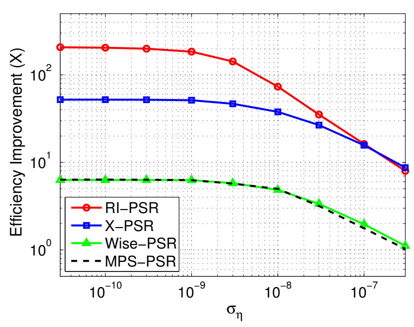


Fig. 17. Impact of $\sigma_\eta$

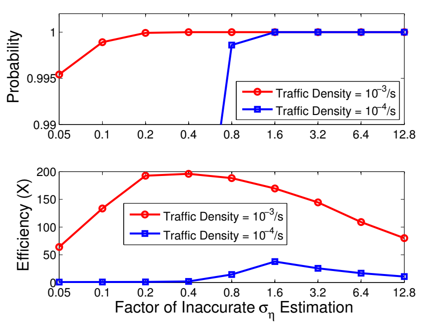


Fig. 18. Inaccurate $\sigma_\eta$ Estimation (RI-PSR)

protocol that proposes the mechanism of early ACK (termination). We set $\sigma_\phi$ for X-PSR as 1 $ms$ that is determined by the minimum TX/RX period within X-MAC's preamble [31]. With the support of *PSR*, X-PSR exhibits high efficiency improvements as shown in Table III. X-PSR requires a longer link establishment phase ($L = 7.5ms$ in Table I) than that of RI-PSR due to its larger detection error $\sigma_\phi = 1ms$ comparing with $\sigma_\phi = 15.3\mu s$ for RI-PSR. Consequently, the rendezvous offsets of X-PSR spread in a wider range in Fig.14.

**Wise-MAC vs. Wise-PSR**

Wise-MAC [25] is a typical synchronous MAC protocol feathering a simple design with adaptive preamble stretching. The detection uncertainty $\sigma_\phi$ for Wise-PSR is still 1 $ms$, because with the MicaZ platform, we have to apply a packet radio [32] to emulate the continues preamble in the original Wise-MAC . And in this case, the accuracy of drift detection is determined by the minimum packet transmission time as that in X-MAC. Table IV shows the performance comparison between Wise-MAC and Wise-PSR. Both Wise-MAC and Wise-PSR favor high traffic load; however, as the traffic density decreases, Wise-MAC's energy cost quickly increases, because its simple drift compensating mechanism (with $4\theta$ [25]) does not scale well with long periods. Wise-PSR, supported with skew estimation and uncertainty model, provides high performance gains with lower traffic densities towards the right side of Table IV.

**CSMA-MPS vs. MPS-PSR**

CSMA-MPS [26] is essentially a combined version of Wise-MAC and X-MAC with better energy performance than both of them, which can be observed by comparing Table III, IV and V. With early ACK, MPS-PSR requires less energy than Wise-PSR in Table V as expected. Like Wise-PSR, the efficiency improvement of MSP-PSR increases as the traffic load decreases, because of the power of skew estimation and on-demand recalibration brought about by *PSR*.

## *B. Simulation Evaluation*

A discrete event-driven simulator has been developed to explore *PSR*'s performance under (i) a broad range of traffic load, (ii) various environmental conditions, and (iii) different environmental factor estimation errors. For each data point, we simulated 30 node pairs running for a duration of 1000 hours with and without the support of *PSR*. Unless noted otherwise, default parameters in Table.I are used in simulation.

**Impact of Traffic Density**

Fig.16 shows efficiency improvements of PSR-augmented protocols under a broad range of traffic densities varying from $10^{-5}/s$ to $10^{-1}/s$ as the $x$ axis. For RI-PSR and X-PSR, their efficiency improvements are comparatively low under low traffic load. However, as traffic density increases, so do their improvements as shown in the figure. When the traffic density reaches $10^{-3}/s$, RI-PSR achieves 200X or larger, and two curves becomes flat afterwards as expected. For Wise-PSR and MPS-PSR (two curves almost overlap in Fig.16), their maximum efficiency improvements appear with medium traffic load. Under extremely low traffic load, improvements of Wise-PSR and MSP-PSR are relatively low because in this case the synchronization cost can not be amortized among traffic; while under high traffic load, Wise-MAC and CSMA-MPS improves regarding energy efficient due to their simple mechanism of schedule updating and adaptive preamble stretching.

**Impact of Environmental Factor $\sigma_\eta$**

Fig.17 gives improvements of PSR-augmented protocol in different environments, represented by varying $\sigma_\eta$ values as the $x$ axis. For all four protocols, Fig.17 shows similar trends for their performance gains: efficiency improvements decline with increasing $\sigma_\eta$. This is expected because more energy is required for synchronization with tougher environments in which clock skew varies dynamically with a larger $\sigma_\eta$.

**Impact of $\sigma_\eta$ Estimation Error**

In practice, estimation errors for the environment factor $\sigma_\eta$ is unavoidable. We investigated the impact of underestimation (by factors between 0.05∼0.8) and overestimation (by factors between 1.6∼12.8) of $\sigma_\eta$ to the rendezvous probability and energy improvement as shown in Fig.18. Underestimation of $\sigma_\eta$ results in reduced rendezvous probability, especially with low traffic load (e.g., $10^{-4}/s$ denoted as square-marked curves in the figure) and worse efficiency due to rendezvous failures. Overestimation of $\sigma_\eta$ does not affect the rendezvous probability, however at the cost of extra energy for redundant synchronization, which also results in worse efficiency.

## VI. Conclusion

This paper presents *PSR*, a practical design for synchronous communication rendezvous in low-duty-cycle wireless networks. By leveraging the duty-cycle pattern of radio operations, *PSR* enables "free" clock drift detection with normal traffic in the network, which works together with an energy-driven adaptive scheme for skew calibration. Test-bed and simulation evaluations demonstrate that *PSR* is practical, versatile, and can be conveniently embedded in state-of-the-art low-duty-cycle MAC protocols to greatly improve energy efficiency.

## REFERENCES

[1] D. Culler, D. Estrin, and M. Srivastava. Guest Editors' Introduction: Overview of Sensor Networks. Computer, 37(8), Aug. 2004.

[2] B. R. Hamilton, X. Ma, Q. Zhao, J. Xu. ACES: Adaptive Clock Estimation and Synchronization Using Kalman Filtering. Mobicom'08.

[3] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, P. Buonadonna, D. Gay, et al. A Macroscope in the Redwoods. SenSys'05.

[4] H.-S. W. So, G. Nguyen, J. Walrand. Practical Synchronization Techniques for Multi-Channel MAC. MobiCom'06.

[5] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. SenSys'04.

[6] M. Buettner, G. V. Yee, E. Anderson, et al. X-MAC: A Short Preamble MAC Protocol for Duty-cycled Wireless Sensor Networks. SenSys'06.

[7] C. Lenzen, P. Sommer, and R. Wattenhofer. Optimal Clock Synchronization in Networks. SenSys'09.

[8] W. Ye, J. Heidemann, and D. Estrin. An Energy-efficient MAC Protocol for Wireless Sensor Networks. Infocom'02.

[9] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai. Canopy Closure Estimates with GreenOrbs: Sustainable Sensing in the Forest. SenSys'09.

[10] J. Polastre. Design and Implementation of Wireless Sensor Networks for Habitat Monitoring. Master Thesis, U.C. Berkeley, 2003.

[11] M. Li and Y.H. Liu. Underground Structure Monitoring with Wireless Sensor Networks. IPSN'07.

[12] Y. Cao, Z. Zhong, Y. Gu, and T. He. Safeguarding Schedule Updates in Wireless Sensor Networks. Infocom'11.

[13] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. IEEE Micro, 22(6):1224, Nov. 2002.

[14] The Science of Timekeeping. Hewlet Packard. Application Note 1289.

[15] J. Elson, and K. Römer. Wireless Sensor Networks: A New Regime for Time Synchronization. Sigcomm Comp. Com. Rev. 33(1), 2003.

[16] Cardinal Components Inc. Clock Oscillator Stability: Measuring Clock Oscillator Frequency Stability. Applications Brief No. A.N. 1006.

[17] P. Dutta, and D. Culler. Practical Asynchronous Neighbor Discovery and Rendezvous for Mobile Sensing Applications. Sensys'08.

[18] J. Elson, L. Girod, and D. Estrin. Fine-grained Network Time Synchronization Using Reference Broadcasts. OSDI'02.

[19] S. Ganeriwal, R. Kumar, and M.B. Srivastava. Timing-sync Protocol for Sensor Networks. SenSys'03.

[20] M. Maróti, B. Kusy, G. Simon, and Á. Ldéczi. The Flooding Time Synchronization Protocol. SenSys'04.

[21] D. Revuz and M. Yor. Continuous Martingales and Brownian Motion, 2nd Edition, Springer-Verlag 1994.

[22] T. V. Dam, and K. Langendoen. An Adaptive Energy-efficient MAC protocol for Wireless Sensor Network. Sensys'03.

[23] R. Zheng, J. Hou, and L. Sha. Asynchronous Wakeup for Ad hoc Networks. MobiHoc'03.

[24] H. Stark, and J. W. Woods. Probability and Random Processes with Applications to Signal Processing, 3rd Edition. Prentice Hall, 2002.

[25] A. El-Hoiyi, J.-D. Decotignie, and J. Hernandez. Low Power MAC Protocols for Infrastructure Wireless Sensor Networks. EW'04.

[26] S. Mahlknecht. CSMA-MPS: A Minimum Preamble Sampling MAC protocol for Low Power Wireless Sensor Networks. WFCS'04.

[27] W. Ye, F. Silva and J. Heidemann. Ultra-low Duty Cycle MAC with Scheduled Channel Polling. SenSys'06.

[28] Y. Sun, O. Gurewitz, and D. B. Johnson. RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. SenSys'08.

[29] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, et al. Estimating Clock Uncertainty for Efficient Duty-cycling in Sensor Networks. SenSys'05.

[30] ATMEL, 8-bit AVR® Microcontroller with 128K Bytes In-System Programmable Flash, ATmega128/128L. Rev. 2467S-AVR-07/09.

[31] S. Moon, T. Kim, and H. Cha. Enabling Low Power Listening on IEEE 802.15.4-based Sensor Nodes. WCNC'07.

[32] Texas Instruments. CC2420, 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. Datasheet Rev.B. SWRS041B.

[33] Product List of Seiko Instruments Inc. (SII), 2009. Online availalbe at http://speed.sii.co.jp/pub/compo/quartz/productListEN.jsp

[34] D. Veitch, S. Babu, and A. Pàsztor. Robust Synchronization of Software Clocks Across the Internet. Sigcomm'04.

[35] D. W. Allan. Should the Classical Variance Be Used as a Basic Measure in Standards Metrology? IEEE Trans. on I. M., 36, 1987.

[36] Z. Zhong, P. Chen and T. He. On-Demand Time Synchronization with Predictable Accuracy. Infocom'11.

[37] H. Tijms. Understanding Probability: Chance Rules in Evertyday Life. Cambridge University Press, 2004.

[38] Z. Yang, L. Cai, Y. Liu, and J. Pan, Environment-Aware Clock Skew Estimation and Synchronization for Wireless Sensor Networks. Infocom'12.

[39] P. Dutta, S. Dawson-Haggerty, Y. Chen, C-J Liang, and A. Terzis. Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. SenSys'10.

## APPENDIX

**A.1** Proof for Eq.19. Given the expression of $\tilde{S}_A^B(t_2)$ as Eq.18, where $\phi(t_2)$, $\phi(t_0)$ and $\eta(u)$ are independent, we can have

$$E[(\tilde{S}_A^B(t_2))^2] = E[X^2] + \frac{2\sigma_\phi^2}{\Delta t^2} \tag{26}$$

$$\text{where } X = \int_{t_0}^{t_2} \eta(u)du - \frac{1}{\Delta t}\int_{t_0}^{t_2}\int_{t_0}^{t} \eta(u)dudt \tag{27}$$

$E[X^2]$ includes three terms addressed one by one in the following.
(i) For the left term of $X$ in Eq.27, we have

$$E\left[\left(\int_{t_0}^{t_2} \eta(u)du\right)^2\right] = \int_{t_0}^{t_2}\int_{t_0}^{t_2} E[\eta(u)\eta(v)]dudv = \sigma_\eta^2 \cdot \Delta t$$

which is because $E[\eta(u)\eta(v)] = \delta(u-v)\cdot\sigma_\eta^2$ from Eq.2.
(ii) For the right term of $X$, let $w(m) = \int_{t_0}^{t_0+m}\eta(u)du$, then

$$E\left[\left(\int_{t_0}^{t_2}\int_{t_0}^{t} \eta(u)dudt\right)^2\right] = \int_0^{\Delta t}\int_0^{\Delta t} E[w(m)w(n)]dmdn$$

$w(m)$ is a standard Wiener Process[24], and has a covariance of $E[w(m)w(n)] = min(m,n)\cdot\sigma_\eta^2$ [21]. Thus, we have

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

And the overall expectation is $\frac{1}{\Delta t^2}\cdot\frac{\Delta t^3}{3}\cdot\sigma_\eta^2 = \sigma_\eta^2\cdot\frac{\Delta t}{3}$
(iii) For the cross-product term in $X^2$, we use similar substitution:

$$E\left[\int_{t_0}^{t_2}\eta(u)du \cdot \int_{t_0}^{t_2}\int_{t_0}^{t}\eta(u)dudt\right] = \int_0^{\Delta t} E[w(\Delta t)w(m)]dm$$

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

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

So this term has an expectation of $-\frac{2}{\Delta t}\cdot(\frac{\Delta t^2}{2}\cdot\sigma_\eta^2) = -\sigma_\eta^2\cdot\Delta t$.
Combining results from (i), (ii) and (iii) with Eq.26, we have

$$E[(\tilde{S}_A^B(t_2))^2] = \frac{2\sigma_\phi^2}{\Delta t^2} + \frac{\Delta t}{3}\cdot\sigma_\eta^2 \qquad \square$$

**A.2** Given the expression of $\tilde{t}_A(B_{active})$ in PROOF 2., we have

$$E[\tilde{t}_A(B_{active})^2] = t^2\cdot\sigma^2_{\tilde{S}_A^B(t_2)} + \sigma_\phi^2 - 2t\cdot E[\tilde{S}_A^B(t_2)\phi(t_2)] + E\left[\left(\int_{t_2}^{t_2+t}\int_{t_2}^{t_2+v}\eta(u)dudv\right)^2\right] \tag{28}$$

because $\tilde{S}_A^B(t_2)$ and $\phi(t_2)$ are correlated while both of them are independent from $\eta(u)$ during $t$.

Following the same method in A.1 (ii), the last term in Eq.28 is

$$E\left[\left(\int_{t_2}^{t_2+t}\int_{t_2}^{t_2+v}\eta(u)dudv\right)^2\right] = \frac{\sigma_\eta^2}{3}\cdot t^3 \tag{29}$$

For $E[\tilde{S}_A^B(t_2)\phi(t_2)]$, apply $\tilde{S}_A^B(t_2)$ in Eq.18, we have

$$E[\tilde{S}_A^B(t_2)\phi(t_2)] = E\left[-\frac{\phi^2(t_2)-\phi(t_0)\phi(t_2)}{\Delta t}\right] = -\frac{\sigma_\phi^2}{\Delta t} \tag{30}$$

Combining above results, finally we have

$$E[\tilde{t}_A(B_{active})^2] = \sigma_\phi^2 + \frac{2\sigma_\phi^2}{\Delta t}\cdot t + \sigma^2_{\tilde{S}_A^B(t_2)}\cdot t^2 + \frac{\sigma_\eta^2}{3}\cdot t^3 \tag{31}$$

which converges to Eq.21 in THEOREM 2. $\square$