A

Achieving Energy Synchronized Communication in Energy-Harvesting Wireless Sensor Networks

Yu Gu, Singapore University of Technology and Design
Liang He, Singapore University of Technology and Design
Ting Zhu, State University of New York
Tian He, University of Minnesota

With advances in energy-harvesting techniques, it is now feasible to build sustainable sensor networks to support long-term applications. Unlike battery-powered sensor networks, the objective of sustainable sensor networks is to effectively utilize a continuous stream of ambient energy. Instead of pushing the limits of energy conservation, we aim to design energy-synchronized schemes that keep energy supplies and demands in balance. Specifically, this work presents Energy Synchronized Communication (ESC) as a transparent middleware between the network layer and MAC layer that controls the amount and timing of RF activity at receiving nodes. In this work, we first derive a delay model for cross-traffic at individual nodes, which reveals an interesting *stair effect*. This effect allows us to design a localized energy synchronization control with $\mathcal{O}(d^3)$ time complexity that *shuffles* or *adjusts* the working schedule of a node to optimize cross-traffic delays in the presence of changing duty cycle budgets, where d is the node degree in the network. Under different rates of energy fluctuations, shuffle-based and adjustment-based methods have different influences on *logical connectivity* and *cross-traffic delay*, due to the inconsistent views of working schedules among neighboring nodes before schedule updates. We study the tradeoff between them and propose methods to update working schedules efficiently. To evaluate our work, ESC is implemented on MicaZ nodes with two state-of-the-art routing protocols. Both testbed experiment and large scale simulation results show significant performance improvements over randomized synchronization controls.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: Wireless Sensor Networks, Low-Duty-Cycle Networks, Energy Synchronized Communication, Logical Connectivity

ACM Reference Format:
Y., Gu, L., He, T., Zhu, T., He. 2012. Achieving Energy Synchronized Communication in Energy-Harvesting Wireless Sensor Networks. ACM Trans. Embedd. Comput. Syst. V, N, Article A (January YYYY), 25 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

With the increasing need for cyber-physical interaction, Wireless Sensor Networks (WSN) have emerged as a key technology for many long-term applications, such as military surveillance, field monitoring and assisted living. Due to the stringent constraints on cost and form factors, traditional battery-powered sensor networks must balance the trade-off between sustainability and system performance. Due to the lim-

A conference paper [Gu et al. 2009b] containing some preliminary results of this paper has appeared in IEEE ICNP 2009. Author's addresses: Y. Gu and L. He are with Singapore University of Technology and Design, Singapore. T. Zhu is with the State University of New York, Binghamton, USA. T. He is with University of Minnesota, Twin Cities, USA.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

ited on-board energy supply of sensor nodes, the design objective of these systems is normally to *conserve as much energy as possible* while meeting minimal performance requirements [Gui and Mohapatra 2004; Hoang and Motani 2007; Zamalloa et al. 2008; Liu et al. 2011; Eu et al. 2010; Langendoen and Meier 2010].

Because sensor networks usually interact with the physical environment, they are especially well-suited to exploit ambient energy resources. For example, there are already many existing technologies to extract energy from the environment such as solar, thermal, optical, and kinetic energies [Meninger et al. 1999; Rahimi et al. 2003; Wright et al. 2000; Fafoutis and Dragoni 2011]. In addition, several recent works have built prototypes [Park and Chou 2006; Kansal et al. 2004; Zhu et al. 2009] to demonstrate the feasibility of deploying *sustainable sensor networks* with energy-harvesting nodes. However, influenced by the traditional belief in energy management, the designers of those systems still think that *maximum* energy-harvesting and *minimum* energy consumption are always beneficial.

In contrast to the previously ingrained belief, we take a new position in this work. We argue that energy management should *synchronize* the supply with demand. The equilibrium point is achieved when the energy supplied and demanded are in balance. It is not always beneficial to conserve energy with degrading system performance when a network can harvest excess energy because energy storage devices (e.g., batteries or capacitors) are always limited in capacity and usually leakage-prone. Therefore, energy saving with reduced performance during energy-rich periods is actually wasteful and counter-productive. In other words, in sustainable sensor networks, we would like to *consume as much energy as necessary* for improved system performance while maintaining their sustainability for those best effort sensor applications. Although a similar idea of adjusting energy consumption according to energy supply has been implied in [Kansal et al. 2007; Vigorito et al. 2007], we advance the state-of-the-art by optimizing the system performance under available energy. More specifically, we explicitly investigate the relationship between the consumed energy and performance, and adaptively synchronize node schedules based on specified duty cycle budgets.

In this work, we are particularly interested in studying the impact of energy synchronization on communication performance. We propose *Energy Synchronized Communication* (ESC), a novel solution that *dynamically synchronizes node activity patterns with available energy budgets, so as to minimize communication delay at individual nodes*. Specifically, by exploiting an interesting stair-effect of delay during energy synchronization, ESC is capable of minimizing communication delay at individual nodes in linear time and acts as a generic middleware between the MAC layer and network layer. The major intellectual contributions of this work are as follows:

— An Energy Synchronized Communication (ESC) protocol is designed as a generic, transparent middleware service for supporting existing MAC and network protocols.
— We formulate a generic model for cross-traffic delay. This model reveals an interesting *stair effect of delays* in low duty cycle networks. We show that, counterintuitively, the communication delay is not affected when a packet is received as long as the packet arrives within a certain time interval. This stair effect allows us to design a localized $\mathcal{O}(d^3)$ algorithm to minimize communication delay while synchronizing duty-cycles of individual nodes with available ambient energy, where d is the node degree in the network.
— *Logical Link Quality (LLQ)* is put forward to reflect the true reliability of a link in low duty cycle networks. We study the impact of *shuffle-based* and *adjustment-based* synchronization on LLQ and propose how to maintain LLQ with a small overhead.

The rest of the paper is organized as follows: Section 2 advocates the necessity of ESC. Section 3 articulates the network model and related assumptions. Section 4 for-

malizes a delay model for cross traffic. Section 5 and Section 6 describe our energy-synchronized control algorithm and the impact of updates on logical link quality. Section 7 discusses practical design issues. Section 8 and Section 9 present test-bed and simulation evaluation results, respectively. Section 10 briefly discusses related work. Section 11 concludes the paper.

2. MOTIVATION

The motivation of this work originated from our empirical experience of deploying best effort energy-harvesting, sustainable sensor networks. In such networks, the energy supply is usually highly dynamic and the whole network normally has to operate at a low duty cycle to ensure sustainability. With those unique characteristics, effective data communication in such networks therefore poses a new challenge beyond traditional static, battery-powered sensor networks. To generalize major application requirements and illustrate the benefit of energy synchronized communication, we study two representative scenarios, as follows:

Scenario 1: Network Control and Status Report For many energy-harvesting, long-term operation sensor networks, a self monitoring infrastructure that indicates failures and abnormalities is a key component. For example, a volcano monitoring system for Mount St. Helens includes a network control and status report component [Song et al. 2009]. In order to collect system status while not affecting the normal execution of other system functions, such self-monitoring services should ensure the sustainable operation of the whole system while trying to deliver the system status as soon as possible. Consequently, the capability of ESC to adaptively change radio activity patterns while minimizing cross-traffic communication delays greatly benefits such middleware services.

Scenario 2: Delay Tolerant Networking For many infrastructure monitoring applications, depending on the availability of energy harvested from ambient environment, the system may decide to delay or slow down the transmission of data messages so as to ensure the stable data sampling rate [Steck and Rosing 2009]. When there is excess energy available, the application would need to speed up the data transmission process so as to deliver the delayed messages as soon as possible. In such scenarios, ESC would also help conserve the energy when the power supply is insufficient while reducing the End-to-End (E2E) communication delay.

In the following paragraphs, we explain the impact of dynamic energy supply and low duty cycle networking on the data communication process in detail.

First, in energy-harvesting networks, the harvested energy is usually very unpredictable and could vary significantly over time [Kansal et al. 2004; Zhu et al. 2009]. To study the empirical energy-harvesting rate over time, we have deployed 11 solar-powered sensor nodes in an open field. We collect the energy-harvesting rates for the 11 deployed sensor nodes for a period of two days and Figure 1 plots the harvested energy over time for two nodes, which are spatially close to each other. Clearly in Figure 1, in the time dimension, the harvested energy varies significantly both within a day and across days. In the space dimension, even when these two nodes are physically co-located, their energy-harvesting rates also vary significantly. Furthermore, although energy-harvesting sensor nodes are usually equipped with rechargeable batteries or super capacitors to alleviate the impact of energy variations, they are limited in energy storage capacity due to the small form factor requirement and waste energy unnecessarily due to the problem of energy leakage [Zhu et al. 2009]. With such energy dynamics in the network, we can no longer schedule sensor nodes *a priori* as most existing battery-powered solutions do. Consequently, it is a demanding task to effectively synchronize energy supply with sensor node activities so as to optimize the communication process.

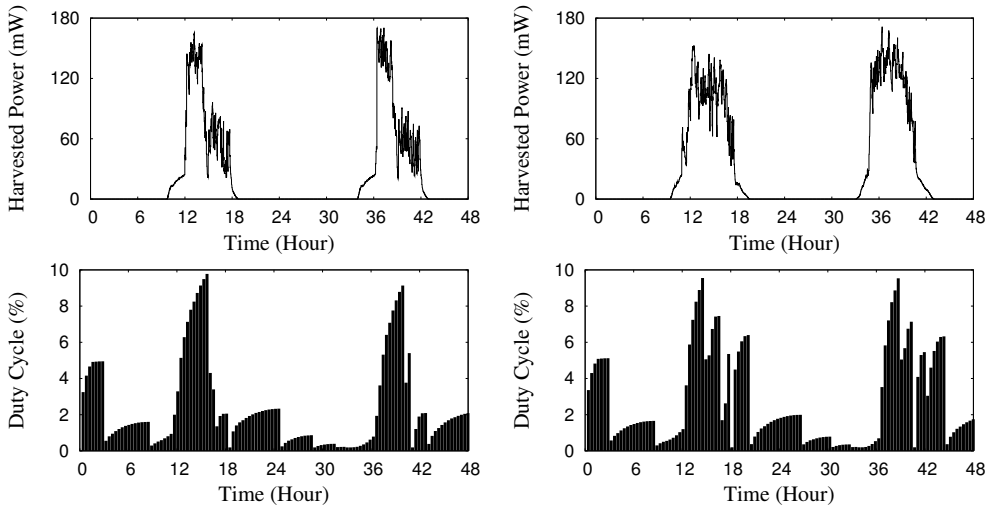


Fig. 1. Harvested Power vs. Duty Cycle

Second, in the long run, ambient energy (e.g., solar, wind) in some cases is not intensive enough to sustain the continuous 100% duty cycle operation of sensor nodes all the time [Kansal et al. 2004; Zhu et al. 2009; X. Jiang, J. Polastre and D. Culler 2005]. For example, from our empirical measurement results, even on a sunny day, the total energy harvested at a node can only allow itself to operate at 100% duty cycle for 6.37 hours. Figure 1 shows the affordable duty cycles of two sensor nodes, given the energy harvested. The dynamic change of the duty cycle is due to the combined effect of the ultra-capacitor's remaining energy, leakage power of the ultra-capacitor, instantaneous energy consumption rate, and expected lifetime in our controller design (detailed in Section 6 of [Zhu et al. 2009]). Our design objective is to consume as much energy as necessary while maintaining sustainability. Therefore, the duty cycle increases from hour 18 to hour 24 when the remaining energy and the leakage power of the ultra-capacitor are high. The duty cycle dramatically decreases at hour 24 when the controller detects that (i) the expected lifetime is significantly less than the target lifetime at instantaneous energy consumption rate and (ii) the leakage power of the ultra-capacitor is relatively low. From hour 24 to hour 27, the controller slightly increases the duty cycle to consume as much energy as necessary to maintain the viability of the sensor node.

From Figure 1, we can see that in order to ensure the continuous operation, the duty cycles of an energy-harvesting node can range only from 0.2% to 9.78%. Essentially, during the operation of an energy-harvesting network, sensor nodes have to activate briefly and stay in a dormant state for a long period of time. In order to forward a packet in such always-dormant networks, a sender would experience *sleep latency*: the time spent waiting for the receiver to wake up [Gu and He 2007]. Moreover, because communication links between low-power sensor devices are usually unreliable [Zhao and Govindan 2003], it brings further challenges in managing communication in sustainable sensor networks.

To the best of our knowledge, no prior work has studied the impact of *energy synchronization* on *low duty cycle networks with unreliable links*. We contribute this research direction with (i) theory, (ii) architecture and (iii) design.

3. SYSTEM MODELS AND ASSUMPTIONS

Before presenting ESC in detail, we introduce the models and assumptions used in this work. To simplify our description, we introduce our ESC design assuming (i) neighboring nodes are synchronized in the unit of a time instance. Local Synchronization can be achieved by using a MAC-layer time stamping technique [Maroti et al. 2004], which achieves an accuracy of 2.24μs with the cost of exchanging a few bytes of packets among neighboring nodes every 15 minutes. This level of accuracy is sufficient for most applications; (ii) There is at most one packet transmission within such a time instance. Later on in Section 7, we discuss how we can relax those assumptions in practice; (iii) Energy harvested by individual devices normally cannot support 100% duty cycle operation and the devices have to operate in the low duty cycle mode to ensure sustainability. This is evidenced by our field experiments and other deployment experiences [Kansal et al. 2004; Zhu et al. 2009; X. Jiang, J. Polastre and D. Culler 2005]; (iv) Data traffic/congestion is low as nodes normally activate briefly and stay in the dormant state for the majority of time. This assumption holds true for existing low duty cycle sensor networks [Gu and He 2007; Lu et al. 2005; Su et al. 2008].

3.1. Duty Cycle Controller

To ensure the sustainable operation of energy-harvesting sensor network applications, we need to make sure sensor nodes can survive even during dark periods, the time when there is not enough ambient energy available. To decide the appropriate duty cycle of a node, we need to consider factors such as the remaining energy stored in the energy storage devices, instantaneous energy-harvesting rate and the instantaneous energy consumption rate. In this paper, we build on top of our previous work which introduces a duty cycle controller by measuring the remaining energy at the energy storage devices, monitoring the duration of time that sensor is in the active and sleep mode, modeling average power consumption, predicting remaining energy at a specific time in the future and finally deciding the most suitable duty cycle [Zhu et al. 2009]. For our ESC design, we take inputs from such duty cycle controllers which only decides how much energy a node can afford to spend, then decides when a node should be active so as to minimize communication delay in the network.

3.2. Working Schedule

The working schedule of a sensor node denotes the active-dormant behaviors of the sensor node over its lifetime. It consists of a set of *active instances*, during which a node can receive packets. Each active instance j at node i can be represented by a tuple (t_j^i, d_j^i), where t_j^i denotes the starting time of the active instance and d_j^i denotes the corresponding duration of the active instance j. Because many sensor node working schedules are periodic [Wu et al. 2007; Su et al. 2008; He et al. 2009], it is sufficient to represent an infinite sequence of active instances, using repeated occurrences with a round time T. Let Γ_i be the working schedule of node i and the number of active instances within a period be M, we can have $\Gamma_i = \{(t_1^i, d_1^i), (t_2^i, d_2^i), ..., (t_M^i, d_M^i)\}$. According to its working schedule, a node continuously transits its state between active and dormant. Therefore, the duty cycle of node i is $\frac{\sum_{j=1}^{M} d_j^i}{T}$. For example, Figure 2 shows a periodic working schedule $\Gamma_i = \{(1,1), (5,2), (8,1)\}$ with a time period 10. The duty cycle of node i here is $\frac{4}{10} = 40\%$.

To simplify our description, in the rest of the paper we assume all active instances have the same durations (τ). When a node is said to be active at time t, it has an active instance that starts at time t with a duration of τ. We note that this definition of working schedule can actually accommodate active instances with varying durations. Essentially, if we let τ be the finest granularity of time durations, we can represent

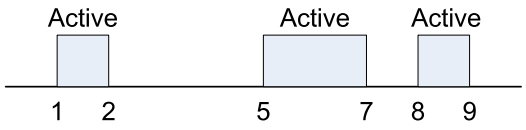


Fig. 2. A Working Schedule

any node schedule with the fixed τ. The working schedule of a node i therefore can be simplified as $\Gamma_i = \{t_1^i, t_2^i, ..., t_M^i\}$.

3.3. Network Model

We assume a network with N sensor nodes. At a given time t, a node is in either an active or a dormant state. When a node is in the active state, it can receive packets transmitted from neighboring nodes. When a node is in the dormant state, it turns off all function modules except a timer (for the purpose of waking itself up). In other words, a node can *wake up to transmit a packet at any time, but can receive packets only when it is in its active state*. Because a node can receive packets only during its active states, the *packet ready time* at a node (i.e., the time a node receives a new packet and is ready to forward to the next hop node), *is the same as active instances in its working schedule*. For each individual node, as it normally needs to wake up beyond its original working schedule for data transmissions, its actual duty cycle is normally larger than the originally specified duty cycle. To compensate for such duty cycle inflations, we rely on the duty cycle controller layer introduced earlier, which closely monitors the actual energy consumption by the sensor node [Kansal et al. 2007; Zhu et al. 2009; Vigorito et al. 2007]. Specifically for this work, we utilize the duty cycle controller layer introduced in [Zhu et al. 2009] to ensure the sustainable operation of individual nodes.

3.4. Sleep Latency in Low Duty Cycle Network

In connected networks, a one-hop packet delivery latency usually includes processing delay, transmission delay, and propagation delay, which are normally in the order of milliseconds. In low duty cycle sensor networks, however, a sender may need to wait for its receiver to wake up before it can send a packet. We define *sleep latency* as the time duration from the moment a packet is ready at the sender to the moment the destined one-hop receiver is active. Sleep latency is usually in the order of seconds, which is much longer than other delivery latencies. Therefore, in this paper we only consider sleep latency for measuring E2E delay. In a network with perfect links, the E2E delay is the sum of sleep latencies along the path of data delivery.

To further illustrate the concept of sleep latency, let us assume node a has a packet ready to be sent at time 1, and the receiving node b wakes up at time 3, the sleep latency for the first attempted transmission from node a to node b therefore is $3 - 1 = 2$.

3.5. Localized Schedule Adjustment

We focus on the localized scheduling adjustment in this paper. Specifically, the scheduling adjustment of a given node will have a quite limited impact on the schedule of other nodes. First of all, the schedule adjustment of a given node is only determined by its own available energy, and thus the decision on whether the adjustment should be carried out is very localized. Second, the schedule adjustment of a given node only has a very small impact on the operation of its predecessors and successors. To illustrate this, let us assume node b has augmented one active instance to its schedule, which means its predecessor p has more chances to transmit packets to b, and thus may consume energy faster. However, these communication opportunities are utilized only if p has packets to send to b, while the number of packets that need to be transmitted is

usually quite limited in low duty cycle sensor networks. As a result, the augmenting of active instances at node b only has a small impact on the energy consumption, and thus the active instances schedule, of its predecessor nodes. A similar conclusion can be obtained for the successor nodes of b. The same reasoning holds when b decreases its active instances. Based on these two points, we can see that the schedule adjustment of nodes is a very localized operation.

4. MODELING OF CROSS-TRAFFIC DELAY

4.1. Cross-Traffic Pattern

ESC is designed to be a flexible middleware between the network layer and MAC layer, so that it can be used to support various existing routing protocols. Therefore, it is important to have a delay model that can capture the behavior of cross-traffic (many-to-many), a generalized case for one-to-one, many-to-one, and one-to-many traffic. The relationship between the cross-traffic delay and E2E delay is further investigated in Section 9.4.

For a node b in the network, depending on the specific routing protocol adopted, there may be a set of nodes that forwards packets to node b, and we call those nodes *predecessors* of node b. Similarly, for different final destinations or multi-path routing protocols, node b would forward its packets to a certain set of nodes, and we call those nodes *successors* of node b. For example, in Figure 3, the predecessors of node b include node p_1, p_2 and p_3, while the successors of node b are nodes s_1 and s_2. Here we note that a node can be both a predecessor and a successor of node b, if this node exchanges data bidirectionally with node b. For an individual node, by keeping track of source and destination node identification numbers of incoming and outgoing data packets, respectively, it can easily acquire and update its predecessor and successor list locally.

Note that although the modeling and design introduced in this and the following sections are based on fixed predecessors and successors of a given node, nodes are able to perform adaptive ESC operation even when the routing paths change. This is because the ESC operation only requires localized input parameters such as the predecessors/successors of nodes and the percentage of the traffic through a node to its given successor.

To model the cross-traffic delay at node b, we consider the expected delays for packets from all predecessor nodes through node b, then to corresponding successor nodes.

4.2. Delay Modeling

Assume at a predecessor node p_1, a packet destined to node b is ready at time t, where $t \in [0, T]$. Because the radio link between a pair of nodes is usually not perfect, node p_1 may need to initiate multiple transmissions before the packet has successfully arrived at node b. In order to obtain a corresponding sleep latency for each attempted transmission at node p_1 after packet ready time t, we introduce a cycle representation of a node working schedule shown in Figure 4.

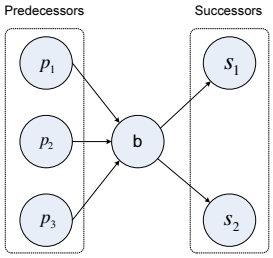


Fig. 3. Predecessors and Successors

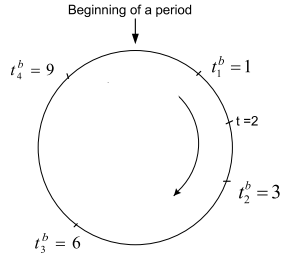


Fig. 4. A Cyclic Working Schedule

In Figure 4, the cycle is equally divided by T ticks. Beginning at the 12 o'clock position, the time increases from 0 to T clockwise. Consequently, we can easily label the sequence of active instances at node b on the cycle. To measure the sleep latency (denoted as $L_t^{\Gamma_b}(k)$) introduced by working schedule of node b, for a given k^{th} attempted transmission at time t, we can simply start from time t, follow the clockwise direction and measure the total distance traversed by visiting k labels after time t on the cycle. For example, as depicted in Figure 4 where T is 10 units of time, the packet is ready at time $t = 2$ and node b wakes up during time 1, 3, 6 and 9. For the first attempted transmission ($L_t^{\Gamma_b}(1)$), the corresponding sleep latency is 1 because the total time traversed for visiting one label (t_2^b) from time 2 is 1. Similarly, for the fourth attempted transmission ($L_t^{\Gamma_b}(4)$), the sleep latency is total time traversed from t to t_2^b, then to t_3^b, t_4^b and the fourth label t_1^b, which gives $1 + 3 + 3 + 2 = 9$ in total.

Link Reliability: Let the bi-directional link quality p_{ab} denote the success ratio of a round-trip transmission (DATA and ACK) between node a and node b. The probability that packet transmission succeeds at the k^{th} attempt is the probability that previous $k-1$ attempts failed times the probability that the current attempt succeeds, which is simply the link quality p_{ab}. Therefore the probability that the packet reaches node b from node a at its k^{th} attempt can be expressed as $(1-p_{ab})^{k-1}p_{ab}$.

Assuming the maximum number of packet transmission attempts within the network is R_{max}, for the packets arriving at node b from a, the probability that they arrive at the k^{th} attempt is under the condition that the packet is delivered within R_{max} transmission attempts. The probability that a packet is delivered within R_{max} transmission attempts can be expressed as $1-(1-p_{ab})^{R_{max}}$, and the corresponding conditional probability can be written as:

$$P_{ab}(k) = \frac{(1-p_{ab})^{k-1}p_{ab}}{1-(1-p_{ab})^{R_{max}}} \tag{1}$$

Conditional Expected Delay Over a Single link: For a packet ready time t at node a, the expected transmission delay to reach node b is the sum of the product of probability that the packet reaches node b at its k^{th} attempt and corresponding sleep latency. Consequently, it can be formulated as:

$$D_{ab}(t) = \sum_{k=1}^{R_{max}} P_{ab}(k) L_t^{\Gamma_b}(k) \tag{2}$$

Conditional Expected Delay from One Predecessor to One Successors: For a packet ready time t at a predecessor node p_i, assuming the packet arrives at node b at the k^{th} attempted transmission, its corresponding delay is simply $L_t^{\Gamma_b}(k)$. Upon receiving the packet at time $t + L_t^{\Gamma_b}(k)$, node b would forward the received packet to a destined successor node s_j with a delay of $D_{bs_j}(t + L_t^{\Gamma_b}(k))$. Then the expected delay from predecessor p_i to successor s_j with packet ready time t at predecessor p_i is the product of probability that the packet arrives node b at its k^{th} attempted transmission and corresponding cross-traffic delay, which can be expressed as:

$$D_{p_i s_j}^{\Gamma_b}(t) = \sum_{k=1}^{R_{max}} P_{p_i b}(k)(L_t^{\Gamma_b}(k) + D_{bs_j}(t + L_t^{\Gamma_b}(k))) \tag{3}$$

Delay from Multiple Predecessors to Multiple Successors: To model the expected cross-traffic delay at node b from all packet ready times at all predecessor nodes to all successor nodes, node b needs to know the portion of traffic that reaches node b from each packet ready time at all predecessor nodes to all successor nodes. To obtain those statistics, each predecessor of node b can piggyback packet the ready time for each sent packet. Then at node b, with known sender and packet ready time for each

packet, it can easily keep track of what percentage of packets is from a given packet ready time t at a predecessor node p_i, to a specific successor node s_j. For each packet ready time $t_k^{p_i}$ at a predecessor node p_i, we can denote the percentage of its traffic through node b to a successor node s_j as $W_k^{p_i s_j}$. Let the number of packet ready times at the predecessor node p_i be N_{p_i} and the number of predecessor nodes and successor nodes at node b be N^p and N^s, respectively, we can express the expected delay of cross-traffic at node b with working schedule Γ_b as:

$$D_{\Gamma_b} = \sum_{i=1}^{N^p} \sum_{j=1}^{N^s} \sum_{k=1}^{N_{p_i}} W_k^{p_i s_j} D_{p_i s_j}^{\Gamma_b}(t_k^{p_i}) \tag{4}$$

The computational complexity of calculating the cross-traffic delay is determined by N^p, N^s, N_{p_i}, and R_{max}. It is clear that N^{p_i} and R_{max} can not be excessively large in low-duty-cycles sensor networks, and are upper bounded with a specific network setting. Furthermore, it is clear that both N^p and N^s is bounded by the node degree d. Thus as a summary, the computational complexity of calculating D_{Γ_b} is $\mathcal{O}(d^2)$. Note that d usually is also constrained, as will be explained below.

Next we discuss the spatial complexity in calculating D_{Γ_b}. From Equation 4, we can see that for node b to calculate D_{Γ_b}, it has to know the schedules of all its predecessors and successors. If it takes 1 B of memory to store one active instance, a total amount of $1 \times M_b = M_b$ B memory is needed to store the schedule of one node, where M_b is the number of active instances of b, which is usually a small number in low duty cycle sensor networks. Thus, the total amount of memory for b to calculate D_{Γ_b} is $1 \times M_b(N^p + N^s) = M_b(N^p + N^s)$ when there is no overlapping between the sets of predecessors and successors of b, and smaller than this when the two sets overlap. It is clear that asymptotically, $N^p + N^s \leq 2d$, where d is the average number of neighbors in the network, and is desirable to be small to reduce access contention [Gelal et al. 2005]. On the other hand, the memory available for a typical MicaZ mote is 128 kB, in addition to a measurement flash of 512 kB. This available memory guarantees the feasibility in the calculation of D_{Γ_b}.

5. ENERGY SYNCHRONIZATION CONTROL

With the cross-traffic delay model available, we now introduce energy synchronization control for minimizing communication delay. In this section, we first assume the working schedules of predecessors and successors of a node are known and up-to-date. Later in Section 6, we discuss the impact of obsolete schedules and methods to keep the schedules up-to-date.

5.1. Main Idea

As shown in Section 2, energy harvested from surrounding environments varies significantly over time at a sensor node. In order to make full use of the available energy supply, we need to enhance system performance when there is abundant scavenged energy available; conversely, we need to decrease duty cycles with a minimum performance degradation when there is a shortage in the power supply. Previous works [Kansal et al. 2007; Vigorito et al. 2007] have demonstrated methods for deciding appropriate duty cycle of a sensor node with an in-situ energy supply. In this section, we further focus on the impact of duty cycle changes on communication delay. More specifically, when we increase the duty-cycle of a node with additional available energy, we aim to minimize cross-traffic delay at the node. Similarly, when a node needs to decrease its duty cycle due to the insufficient energy supply, we would like to achieve a minimum increase in the cross-traffic delay of the node. As a result, harvested energy at an indi-

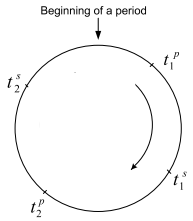


Fig. 5. Period Partition Example

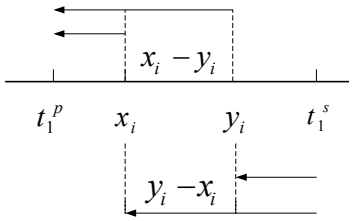


Fig. 6. Delay Difference Example

vidual node is synchronized by adjusting its duty cycle, with the objective to minimize the communication delay in the network.

5.2. Decrementing Single Active Instance

For a given node b, assume its working schedule is $\Gamma_b = \{t_1^b, t_2^b, ..., t_{M_b}^b\}$ and the energy supply can only afford $M_b - 1$ active instances to guarantee the sustainability of the node. Therefore, we need to remove one active instance from node b's working schedule such that the increase of cross-traffic delay at node b is minimized. Because sustainable sensor nodes work at extremely low duty cycles, the number of active instances in its working schedule is small and would always be below a constant value $M_b \leq \frac{T}{\tau}$. Consequently, we can simply attempt to remove each of the existing active instances in sequence to find the optimal active instance for decrement, each with computational complexity $\mathcal{O}(d^2)$ as being explained in Section 4.2, and select the one that yields the minimal cross-traffic delay at node b. Therefore, the complexity of the optimal single active instance decrement, is just $\mathcal{O}(d^2)$.

5.3. Augmenting Single Active Instance

Intuitively, to augment one active instance at a node, we can perform an exhaustive search for all time instances within a round time T and choose the time instance that yields a minimum cross-traffic delay at a node. Although this simple algorithm is acceptable, a much more efficient algorithm can be designed based on the stair effect as presented in this section.

For a given node b, we can divide its time period into multiple intervals according to active instances of its predecessors and successors. Formally, we define an interval as a time duration between two consecutive time instances from working schedules of all predecessors and successors of node b on a cyclic working schedule. The summation of all intervals on a cyclic working schedule is equal to T. For example, as shown in Figure 5, assume the predecessor and successor of node b are node p and node s, respectively. Let the active instances at node p and node s be $\{t_1^p, t_2^p\}$ and $\{t_1^s, t_2^s\}$, respectively. According to their locations on the cyclic working schedule, we can easily obtain four intervals for this specific scenario, namely: (t_1^p, t_1^s), (t_1^s, t_2^p), (t_2^p, t_2^s) and (t_2^s, t_1^p).

Intuitively, one would expect the timing of an augmented active instance within an interval to yield different cross-traffic delays. However this is not the case. We observe that *given schedules of predecessors and successors of node b, cross-traffic delay at node b depends only on the* **counts***, instead of* **timing***, of active instances within each interval.*

To validate this observation, it is sufficient to prove that for two arbitrary layouts of active instances of the same size within an interval, the cross-traffic delays at node b are the same if packets are received within this interval.

LEMMA 5.1. *For any packet ready time t at a predecessor node of node b, let $X = \{x_1, x_2, \cdots, x_m\}$ and $Y = \{y_1, y_2, \cdots, y_n\}$ be two sets of active instances of node b within the same interval starting at t ($t < x_1 < x_2 < \cdots < x_m < t + t'$ and $t < y_1 < y_2 <$*

$\cdots < y_n < t + t'$),*where* $t + t'$ *is the end of the current interval determined by the active instances of the predecessor and successor (as shown in Fig. 5). Let* D_X *and* D_Y *be corresponding cross-traffic delays. Then* $D_X = D_Y$ *if* $m = n$.

PROOF. For any packet ready time t at a predecessor node p, because $x_1, x_2, ..., x_m$ and $y_1, y_2, ..., y_m$ are in the same interval, as shown in Figure 6, the difference of sleep latencies for the i^{th} attempted transmission for X and Y after time t is just the difference of their respective active instances. Consequently, if $m = n$, then $L_t^X(i) - L_t^Y(i) = x_i - y_i$, for any $1 \leq i \leq m$.

Similarly, because x_i and y_i are within the same interval, the j^{th} attempted transmission from either x_i or y_i reaches the same active instance at a successor node s. As clearly shown in Figure 6, if $m = n$, then $L_{x_i}^{\Gamma_s}(j) - L_{y_i}^{\Gamma_s}(j) = y_i - x_i$, where $1 \leq i \leq m$ and $1 \leq j \leq R_{max}$.

By applying Equation 3 to X and Y, we have,

$$\begin{aligned} D_{ps}^X(t) - D_{ps}^Y(t) &= \sum_{i=1}^{R_{max}} P_{pb}(i) \left(L_t^X(i) - L_t^Y(i) + D_{bs}(L_t^X(i)) - D_{bs}(L_t^Y(i)) \right) \\ &= \sum_{i=1}^{R_{max}} P_{pb}(i) \left(x_i - y_i + D_{bs}(x_i) - D_{bs}(y_i) \right) \end{aligned} \tag{5}$$

Since,

$$D_{bs}(x_i) - D_{bs}(y_i) = \sum_{j=1}^{R_{max}} P_{bs}(j)(L_{x_i}^{\Gamma_s}(j) - L_{y_i}^{\Gamma_s}(j)) = \sum_{j=1}^{R_{max}} P_{bs}(j)(y_i - x_i)$$

As $\sum_{j=1}^{R_{max}} P_{bs}(j) = 1$ we have

$$D_{bs}(x_i) - D_{bs}(y_i) = y_i - x_i. \tag{6}$$

Consequently,

$$D_{ps}^X(t) - D_{ps}^Y(t) = \sum_{i=1}^{R_{max}} P_{pb}(i)(x_i - y_i + y_i - x_i) = 0. \tag{7}$$

As a linear combination of $D_{ps}^X(t) - D_{ps}^Y(t)$ for all packet ready times at all predecessor nodes, we have $D_X - D_Y = 0$. □

Note that the requirement on $m = n$ in lemma 5.1 always holds when augmenting an active instance to the schedule of a node. Specifically, given a specific node b, $m = n = M_b$, where M_b is the number of active slots in b's current schedule.

According to this observation, when augmenting one active instance within a partitioned time interval, the cross-delay at node b is not affected by the timing of the augmented active instance. In other words, single active instance augmentation yields the same cross-traffic delay at node b within each of partitioned time intervals. Consequently, for a single active instance augmentation, we have a *stair effect of cross-traffic delays at node* b within each partitioned time interval. Based on this counterintuitive observation, we can have the following theorem on finding the optimal single active instance augmentation for minimizing cross-traffic delay at node b.

THEOREM 5.2. *Assuming there are* c *intervals partitioned by the active instances at predecessors and successors of node* b*, and* x_i *is a random active instance within interval* i $(i = 1, 2, ..., c)$. *Let* $D_{\Gamma_b \bigcup \{j\}}$ *represent the cross-traffic delay at node* b *after augmenting*

an active instance j to its original working schedule, and $k = \arg\min_i D_{\Gamma_b \bigcup \{x_i\}}$, then any time instance within interval k is the optimal single active instance augmentation and the complexity of this process is $\mathcal{O}(d^3)$, where d is the average number of neighbours in the network (network density).

PROOF. Denote M as the average number of active slots of nodes. We need $\mathcal{O}(\log(dM))$ time to identify all the c intervals, which can be done by sorting all the active slots of b's neighbors. Within any time interval i, and for any random augmented active instance x_i, we essentially increase the number of active instances within the interval by one. By Lemma 5.1, to find the optimal augmented active instance at a node, we simply need to check the cross-traffic delay with a random active instance augmentation within each of the time intervals, and find the interval k that yields the minimum cross-traffic delay. Because sustainable sensor networks operate at extremely low duty cycle, the packet ready times from predecessor nodes and active instances from successor nodes are quite limited. Thus the complexity of finding the optimal active instance is just $\mathcal{O}(d^2(\log dM + c))$. Considering the fact that $c \leq \frac{dT}{\tau}$, the complexity can be represented by $\mathcal{O}(d^3)$. □

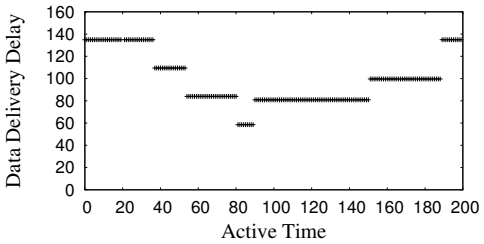


Fig. 7. Stair Effect of Cross-Traffic Delay

To further illustrate Theorem 5.2, we give an example in Figure 7. Assuming one period time contains 200 time instances, Figure 7 shows the expected cross-traffic delay at a node for different augmented active instances. For example, the delay corresponding to active instance 1 represents the delay at node b after augmenting an active instance at time 1. The node shown in Figure 7 has a predecessor with active instances $(36, 53, 80)$ and a successor with active instances $(90, 151, 189)$. According to our above analysis, we can divide one time period into the following intervals: $(36, 53)$, $(53, 80)$, $(80, 90)$, $(90, 151)$, $(151, 189)$, $(189, 36)$. From Figure 7, it is clear that we have a stair effect of delays at the node among the above time intervals, which is consistent with our analysis. The optimal augmented active instance, therefore, is any value within interval $(80, 90)$.

5.4. Bursty Augmentation and Decrementation

In the previous two subsections, we introduced optimal solutions for augmenting and decreasing *a single active instance* for the scenario that energy variation is slow. However, the change in the harvested energy could be bursty and therefore a node may need to increase or decrease multiple active instances simultaneously. In this section, we present a minimal cost solution for augmenting and decreasing multiple active instances.

For multiple active instances augmentation and decrementation, the exhaustive search approach is no longer acceptable because the computational complexity grows exponentially with the number of augmented or decreased instances. Therefore, a low-cost solution with guaranteed optimality in performance is desirable. Fortunately, we

observe that the multiple active instances augmentation and decrementation can be *optimally* solved with a greedy approach.

THEOREM 5.3. *The optimal solution for augmenting/decreasing u active instances at a node can be obtained by applying the single active instance augmentation/decrementation u times.*

Because the computational complexity of single active instance augmentation or decrement is $\mathcal{O}(d^3)$ and $\mathcal{O}(d^2)$ respectively, the corresponding complexity for augmenting or decrementing u active instances is $\mathcal{O}(ud^3)$ and $\mathcal{O}(ud^2)$.

6. MAINTAINING LOGICAL CONNECTIVITY

In the previous section, energy synchronization control is designed to adjust the working schedules at receiving nodes, assuming the schedule of predecessors and successors are known and up-to-date. In this section, we investigate the impact of obsolete schedules and how to maintain connectivity while updating schedules.

6.1. Impact of Schedule Updates

To understand how obsolete schedules can affect the connectivity between nodes, let us consider an example where the working schedule of node b has been changed but has not yet been updated to its predecessor node p:

— **Unnecessary loss:** If node b decreases its duty cycle due to insufficient energy supply, then one or more original active instances at node b may be removed. However, before node p is updated, it would continue to deliver packets according to the old (obsolete) working schedule of node b, which could suffer significant greater packet loss than necessary.
— **Suboptimal delay:** If node b increases its duty cycle and a predecessor node is unaware of the augmented active instances at node b, it would continue to deliver the packets at node b's original active instances. In addition, for both predecessors and successors, the new schedule of node b is essential for them to perform effective energy synchronization. Therefore, it is crucial for node b to promptly disseminate its new working schedule to its predecessor and successor nodes.

6.2. Reactive vs. Proactive Updates

In order to inform predecessor nodes or successor nodes of its new working schedule, node b can either proactively notify them immediately after its duty cycle synchronization or reactively send its new schedule to nodes with which it has ongoing communication. Specifically, for proactive notification, node b generates one packet that contains its new schedule for each of its predecessor and successor nodes, and tries to deliver all those packets according to the active instances of predecessor and successor nodes. In contrast, for a reactive update, node b only sends its new schedule to a predecessor or a successor if the predecessor sends a packet to node b or node b sends a packet to the successor nodes.

Obviously, the advantage of proactive notification is that predecessors and successors can receive the new working schedule of node b promptly. However, proactively sending schedule update packets to all predecessors and successors increases the traffic load within the network and consequently increases the chances of collisions and interferences within the network. More importantly, if there is no traffic flow between a predecessor/successor and node b during two or more node working schedule updates, sending those schedule updates would waste energy. In contrast, if adopting a reactive schedule update, node b only sends the new schedule to predecessors or successors with which it has communication, therefore minimizing the chance of sending unnec-

essary schedule updates due to the well-known temporal and spatial locality for traffic flow in communication networks. As a result, in this work, we recommend *the reactive schedule update for the dissemination of new working schedules of node* b.

6.3. Shuffle-Based vs. Adjustment-Based Energy Synchronization

Generally, two approaches can be adopted when node b increases or decreases its duty cycles. Node b can either generate a completely new working schedule with a new given energy budget (termed a *shuffle*), or node b can increase or decrease its duty cycle on top of its previous working schedule (termed an *adjustment*).

To facilitate the comparison of shuffle-based and adjustment-based energy synchronization, we first present two types of connectivities in duty cycled sensor networks, namely, the physical connectivity and logical connectivity. Two nodes are *physically connected* if they are within each other's communication range and *logically connected* only if they can communicate. Unlike traditional networks, a low duty cycle network could be physically connected, but logically partitioned if nodes do not know each other's working schedules.

Let logical connectivity ℓ_{ab} be the packet delivery ratio between two nodes a and b, after R_{max} retransmissions. Let Γ_b be the set of first R_{max} active instances in the original schedule and Γ'_b be the new schedule of node b. The logical connectivity ℓ_{ab} when b switches from Γ_b to Γ'_b, therefore, is:

$$\ell_{ab} = 1 - (1 - p_{ab})^K \; where \; K = |\Gamma_b \cap \Gamma'_b| \tag{8}$$

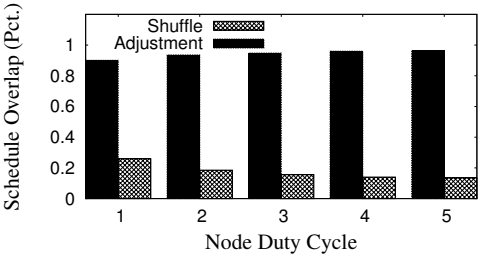


Fig. 8. Comparison of Logical Connectivity

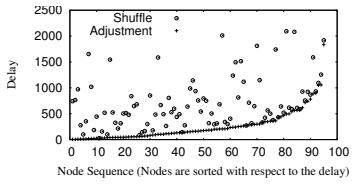


Fig. 9. Comparison of Delay Before Schedule Update

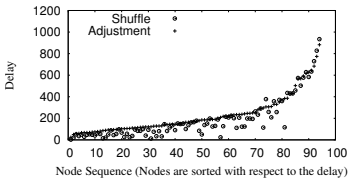


Fig. 10. Comparison of Delay After Schedule Update

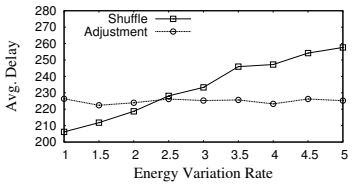


Fig. 11. Delay vs. Energy Variation Rate

6.3.1. Comparison. In this section, we investigate the performance of shuffle and adjustment in terms of logical connectivity and cross-traffic delay. The simulation setup is the same as described in Section 9. Figure 8 shows the percentage of the old working schedule that is preserved after schedule synchronization using shuffle and adjustment. Clearly, adjustment preserves a much larger portion of the old working schedule than shuffle does for all node duty cycles. As the node duty cycle increases, the percentage of the working schedule preserved in an adjustment increases while that for a shuffle decreases. This is because with the increasing duty cycle at a node, the effect of adjusting the node working schedule is constantly reduced, while a shuffling of the

working schedule could result in increasing deviations between the old and new working schedules. For example, the percentage of the working schedule preserved in an adjustment increases from 90.1% to 96.4% when the duty cycle increases from 1% to 5%. In contrast, shuffling decreases the percentage from 26% to 13.5%. Because logical connectivity is decided by the intersection of old and new working schedules at a node, Figure 8 confirms that adjustment maintains a significantly better logical connectivity than that of the shuffle.

Before the new working schedule of a node reaches its predecessors, the traffic delay from the predecessor nodes is dominated by the logical connectivity. Figure 9 shows the traffic delay for shuffle and adjustment before the new energy-synchronized working schedule reaches the predecessor nodes in a randomly generated network. Figure 9 clearly indicates that because adjustment maintains a much better logical connectivity than shuffle, a pair-wise comparison of delays at each node shows that adjustment produces much smaller delays than shuffle. For example, the average delays for an adjustment and a shuffle are 252.79 and 711.2 units of time, respectively.

After the new working schedule of a node reaches its predecessors, a shuffle would achieve the optimal expected delay because it creates a completely new working schedule that minimizes cross-traffic delay at the node. Figure 10 shows cross-traffic delays after a new working schedule has reached predecessor nodes. As expected, shuffle yields smaller delays than adjustment at all nodes. For example, the average delay for adjustment and shuffle is 239.26 and 213.07 units of time, respectively.

Because adjustment produces smaller delays before the new working schedule reaches predecessor nodes and shuffle creates smaller delays after the new working schedule reaches predecessor nodes, the overall delay at a node is influenced by the energy variation at the node. To further investigate the influence of energy diversity on the delay, we introduce the energy variation rate as a new parameter in the simulation, which is defined as *the average number of times that a node increases or decreases its duty cycle over a period of* $100,000$ *units of time*. Figure 11 shows average cross-traffic delays for both adjustment and shuffle under different energy variation rates. When the energy variation rate is low, the delay after schedule dissemination dominates the overall delay, and therefore, shuffle has a smaller overall delay than that of adjustment. For example, when the energy variation rate is 1, the overall delay for adjustment and shuffle is 226.34 and 206.17 units of time, respectively. As the energy variation rate becomes larger, the delay before schedule dissemination weights more in overall cross-traffic delay. Consequently, adjustment that has a smaller delay before schedule dissemination also has a smaller overall delay than shuffle. As shown in Figure 11, from energy variation rate 2.5 to 5, adjustment has a smaller delay than that of shuffle. At energy variation rate 5, the delay for adjustment and shuffle is 225.22 and 257.65 units of time, respectively. In addition, the amplitude of duty cycle changes also has the similar impact, as we can convert such changes of amplitude to the frequency of changes. This study indicates that the design options on ESC should be decided based on how fast and how much ambient energy changes over time.

7. PRACTICAL ISSUES

This section completes our ESC design by discussing several practical design issues, such as time synchronization and multiple transmissions within a time instance.

7.1. Low-cost Time Synchronization

For the sake of clarity, we introduce ESC design in a synchronized mode. Clearly, the operation of ESC depends on neither neighbor time instance nor global synchronization. It is sufficient for ESC to know only the *wake-up time interval* of predecessor and successor nodes. To know those wake-up time intervals, simple and low-cost lo-

cal synchronization techniques [Maroti et al. 2004] can achieve an accuracy of $2.24\mu s$ with the cost of exchange a few bytes of packets among neighboring nodes every 15 minutes. Because an active instance typically ranges from $2{,}000\mu s$ to $20{,}000\mu s$, the accuracy of $2.24\mu s$ is far more than sufficient. In addition, ESC does not require that the transmission starts at the beginning of an active instance, which further relaxes the requirement of accuracy for time synchronization.

7.2. Multiple Transmissions within a Time Instance

While describing our network model in Section 3, we assume there is at most one packet transmission during an active instance. This is true if nodes are equipped with slow radio. However, if a fast radio is used, it is possible to transmit multiple packets within an active instance. To accommodate such scenarios in the modeling of cross-traffic delay, we can simply rewrite the bidirectional link quality between two nodes as $p'_{ab} = 1 - (1 - p_{ab})^m$, where m is the maximum number of transmissions allowed in an active instance. Essentially, the new p'_{ab} represents the probability that the receiver received the packet by m transmissions.

8. IMPLEMENTATION AND EVALUATION

In order to validate the performance and feasibility of ESC in practice, we fully implement ESC on the TinyOS-2.1/Mote platform in nesC. To compare the performance of ESC, we also implement a random schedule synchronization scheme that randomly adjusts active instances of the original node working schedule with increasing or decreasing node duty cycles.

8.1. Experiment Setup

In the experiment, 30 MicaZ nodes are randomly placed in our test-bed, which can accommodate at most 360 sensor nodes. The transmission power at MicaZ motes is tuned down to -24 dBm (by setting the power level as 3) to form a multi-hop network. To reduce the implementation overhead, we synchronize neighboring nodes using FTSP [Maroti et al. 2004], which is a standard component in TinyOS 2.1. The length of each active instance is set to $20\ ms$. The available energy budget over time at each node is derived from the actual energy harvesting profile measured at our running prototype [Zhu et al. 2009]. The range of duty cycle varies from around 0.2% to 10%. According to the available energy budget, each node turns on or off its radio based on the energy synchronized working schedule. To investigate the flexibility of ESC design, we choose two state-of-the-art solutions as underlying routing protocols:

— Link-Quality-based: ETX [Couto et al. 2003] in MobiCom'03
— Sleep-Latency-based: DESS [Lu et al. 2005] in INFOCOM'05

8.2. Performance Comparison

In this section, we compare the E2E delay for both ESC and the randomized scheme. During the experiment, over 1000 packets for each routing protocol are transmitted from a random source to the sink, which is randomly selected from the 30 motes and kept fixed during the evaluation. To minimize the impact of temporal energy variation and link quality fluctuation, nodes in the network periodically switch between ESC and randomized energy synchronization protocols so as to ensure fair performance comparisons.

In Figure 12 and Figure 13 we plot two snapshots of the routing topologies for both ETX and DESS on our testbed. From Figure 12 and Figure 13, we can see the routing topology significantly varies for ETX and DESS even for a 30-node testbed.

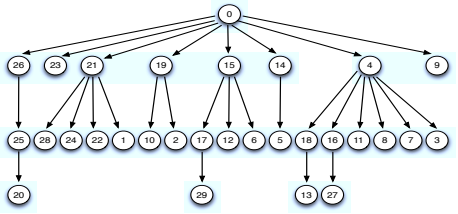


Fig. 12. ETX topology.

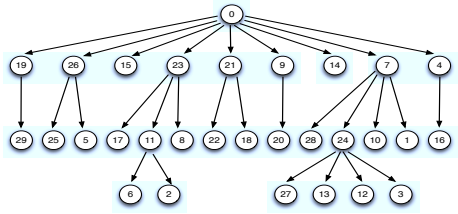


Fig. 13. DESS topology.

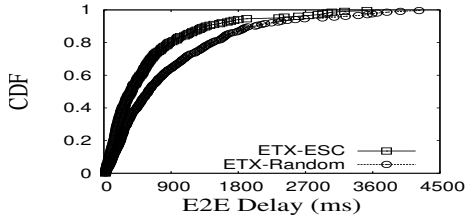


Fig. 14. ETX E2E Delay

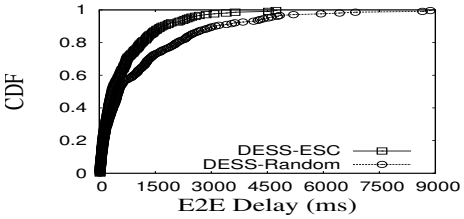


Fig. 15. DESS E2E Delay

In Figure 14 we study the E2E delay for ESC and the randomized scheme under ETX. Clearly, ESC performs much better than the randomized scheme. While 80% of ESC packets reach their destinations within $877\ ms$, the corresponding percentile for the randomized scheme is $1451\ ms$, which is about a 65% increase. Because ETX picks the route with the minimum number of expected transmissions, the performance gap between ESC and the randomized scheme therefore is mainly due to the minimized cross-traffic delay for ESC.

Similar to the results for ETX, ESC also significantly outperforms the randomized scheme under DESS in Figure 15. While the 80-th percentile for ESC is $1131\ ms$, the corresponding delay for the randomized scheme is $2,091\ ms$, which almost doubles the delay of ESC. In addition, the randomized scheme has much longer tail than ESC. While all packets for ESC reach the destinations within $6,299\ ms$, the longest E2E delay for the randomized scheme is $12,343\ ms$. The reason for such a long tail of the randomized scheme is because the penalty of a failed transmission for the randomized scheme is much larger than for ESC, as ESC has carefully scheduled the radio activity to minimize the impact of the failed transmissions.

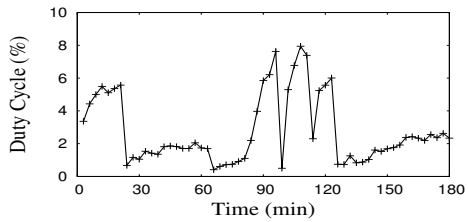


Fig. 16. Node Duty-Cycle Over Time

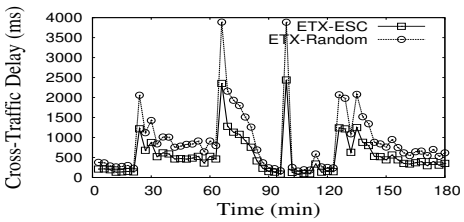


Fig. 17. Cross-Traffic Delay Over Time

To further reveal the performance of ESC over the time dimension, Figure 16 and Figure 17 show the duty cycle of a deployed node and its corresponding cross-traffic delay under ESC and the randomized scheme over a period of three hours. By comparing Figure 16 and Figure 17, we can see the cross-traffic delay matches the available duty-cycle well. For example, the peaks of delay occur when the node duty cycle drops to around 0.4% at time 65 and 100. In addition, although both ESC and the randomized scheme react to the duty cycle change promptly, the cross-traffic delay for ESC is always smaller than that of the randomized scheme. This consistent smaller cross-traffic delay of ESC over time further explains the smaller E2E delay for ESC in Figure 14.

9. SIMULATION EVALUATION

In addition to testbed evaluations in Section 8, in this section we provide simulation results with over 4000 sensor nodes to understand the system performance of ESC under various network settings.

9.1. Simulation Setup

In the simulation, except where otherwise specified, we deploy up to $4,200$ sensor nodes randomly in a $400m \times 400m$ square field. A sink is positioned in the center of the field, and each sensor node sends its packet to the sink over multiple hops. The communication range of sensor nodes is $25\ m$. The radio model was implemented according to [Zuniga and Krishnamachari 2004], which considers the oscillatory nature of the radio links and has several adjustable parameters. During the simulation, we set these parameters strictly according to the CC2420 radio hardware specification [CC2420 2012]. Specifically, the PRR (Packet Reception Rate) for a link of length d is given by

$$PRR(d) = \left(1 - \frac{1}{2}\exp^{-\frac{\gamma(d)}{2}\frac{1}{0.64}}\right)^{8f} \tag{9}$$

where $\gamma(d)$ is the SNR (Signal-to-Noise ratio) which can be determined by the RSSI measurements, and f is the frame size. The RSSI readings are simulated as Gaussian distributed variables, whose mean is determined by the path-loss model with a path loss exponent of 3.0, and the shadowing standard deviation is set to 3.8. The frame size is set to 50 B. Identical to the testbed evaluation, the energy model used in simulation evaluation is also based upon our empirical measurement and energy-harvesting model in [Zhu et al. 2009].

Each experiment was repeated 100 times with different random seeds, node deployments, and node harvested energy. Data collected at each node were obtained by averaging over 10000 source-to-sink communications. The 95% confidence intervals are within 1$\sim$4% of the means.

9.2. System Performance Over Time

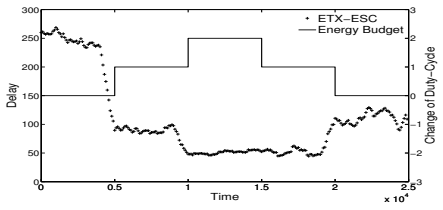


Fig. 18. Delay Over Time

In this section, we reveal the effectiveness of ESC over time in terms of communication delays. Figure 18 shows the average cross-traffic delay at a node over a period of $25,000$ units of time. At time 0, active instances are allocated randomly within nodes (hence not optimally). The node increases its duty cycle at time $5,000$ and $10,000$ and decreases its duty cycle at time $15,000$ and $20,000$. It is clear that after the node increases its duty cycle at time $5,000$, the delay at the node significantly drops. For example, within time interval $[0, 5000]$, the average delay is 249.26 units of time. In contrast, during time $[5000, 10000]$, the average delay drops to 90.51, which is around only 36.3% of the original delay. After increasing the duty-cycle again at time $10,000$, the delay at the node further reduces to 50.32 during time $[10000, 15000]$, almost half the previous delay. When the duty cycle decreases at time $15,000$, the average delay only slightly

increases to 51.36 units of time within time interval $[15000, 20000]$. Finally, when we further reduce the duty-cycle at time $20,000$, the delay increases to 113.66 which is only around 45.6% of the initial delay during time $[0, 5000]$, when allocation is not optimal. From this figure, it is clear that ESC effectively reduces the delay at the node when its duty cycle increases while it minimally increases the delay when the node decreases its duty cycle, converging gradually from an initial sub-optimal allocation into an optimal allocation.

9.3. Comparing Shuffle with Adjustment

Previously, we discussed the per-hop delay impact of shuffle and adjustment. In this section, we systematically compare the system performance for adjustment-based and shuffle-based energy synchronization methods in terms of multi-hop E2E data delivery delay and data delivery ratio.

In both Figure 19 and Figure 20, we deploy sensor nodes in a $150m \times 150m$ area with an average 1% node duty cycle. The energy variation rate, which is defined in Section 6.3.1, is set to be 1 here, which favors shuffle for the single hop scenario. Figure 19 shows the E2E delay and data delivery ratio for adjustment and shuffle under a different maximum numbers of retransmissions. From Figure 19, we can see that under all maximum number of retransmissions, adjustment has a smaller delivery delay and a larger delivery ratio than does the shuffle. For example, when the maximum number of retransmissions is three within the network, the average delay for adjustment is 648.6 units of time while the delay for shuffle is 699.2 units of time, an 8% difference. For the data delivery ratio, under three maximum number of retransmissions, adjustment delivers 86% of data while shuffle delivers only around 70% of messages. Similarly, in Figure 20 adjustment outperforms shuffle in terms of delivery delay and delivery ratio for all node densities.

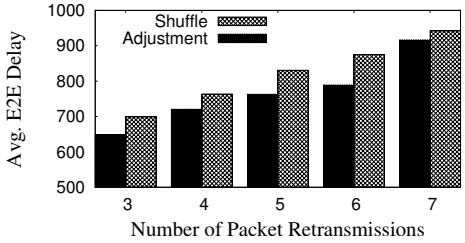
By comparing the performance of these two schedule update approaches, it is clear that *adjustment is favorable in a multi-hop network*. This is because as network size increases, the impact of inconsistent views of node working schedules at individual hops along the data path increases exponentially for E2E delays in the network.

9.4. Impact of Changing Cross-Traffic Delay on the Path Delay

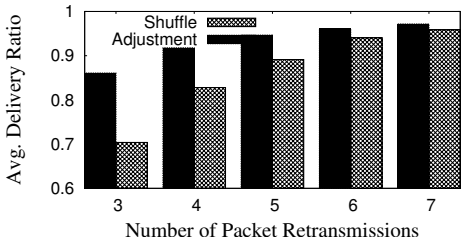
For our ESC design, in order to provide a transparent middleware and support various existing routing protocols, we focus on minimizing the cross-traffic delay at individual nodes. However, it is also essential to study the effectiveness of reducing cross-traffic delay on the actual E2E communication delays. To reveal the correlation of delay reductions for cross-traffic delays and corresponding E2E path delays, Figure 23 plots the expected delay reduction from cross-traffic delays and actual E2E path delays under different numbers of active instance augmentations. From Figure 23, we can see the reduction of expected cross-traffic delay at individual nodes indeed leads to the reduction of E2E path communication delays. In addition, from Figure 23 we can observe that the expected delay reductions from expected cross-traffic delays at individual nodes along the E2E communication path fairly accurately predict the actual E2E delay reductions. Consequently, we can conclude that the cross-traffic delay at individual nodes along a path are positively correlated with the E2E communication delay and can fairly accurately predict the E2E delay changes along the path.

9.5. Comparison with Global Synchronization

To demonstrate the advantage of energy synchronization, we compare ESC with the global synchronization, where the active instances schedule of nodes do not change with the energy supply. In the global synchronization, each node is labeled according to its hop count from the sink, and the route path is constructed based on the label.

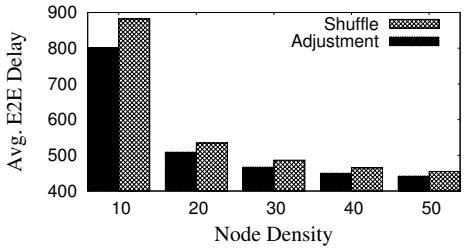


(a) Delay vs. Number of Retransmissions

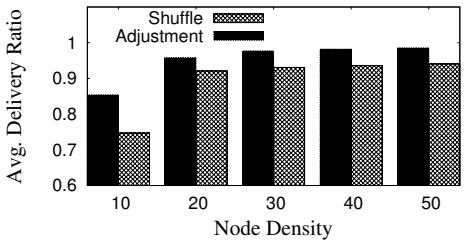


(b) Delivery Ratio vs. Number of Retransmissions

Fig. 19. Performance of Shuffle vs. Adjustment under Different Maximum Retransmissions

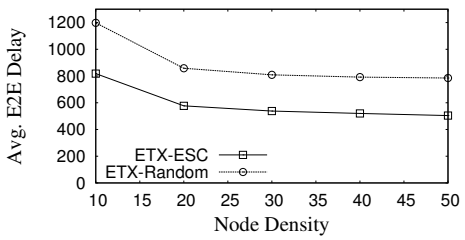


(a) Delay vs. Node Density

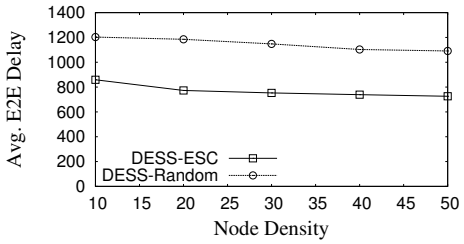


(b) Delivery Ratio vs. Node Density

Fig. 20. Performance of Shuffle vs. Adjustment under Different Node Densities

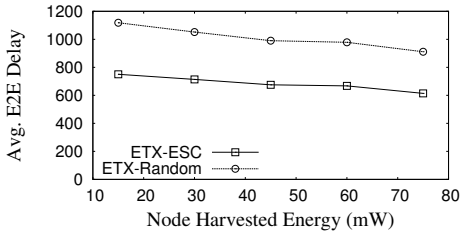


(a) ETX Delay vs. Node Density

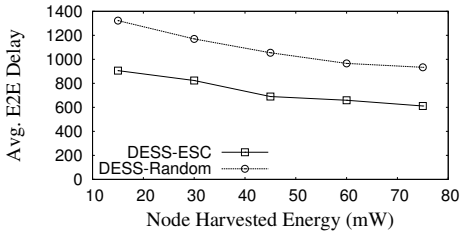


(b) DESS Delay vs. Node Density

Fig. 21. Impact of Node Densities (ESC vs. Randomized Control)



(a) ETX Delay vs. Node Harvested Energy



(b) DESS Delay vs. Node Harvested Energy

Fig. 22. Impact of Node Harvested Energy (ESC vs. Randomized Control)

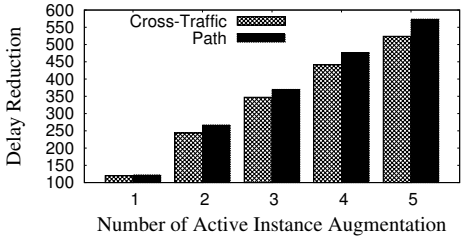


Fig. 23. Number of Active Instance Augmentation vs. Delay Reduction

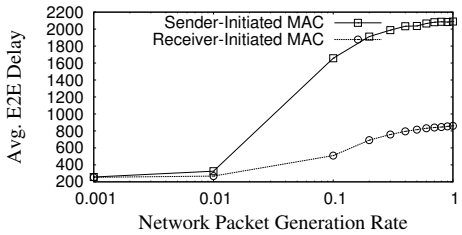


Fig. 24. E2E Delay vs. Data Rate

For the purpose of a fair comparison, the duty cycle of nodes in the global synchronization is set to the average duty cycle of nodes in ESC, or more specifically, 1.5% in our simulation. The comparison results are shown in Figure 25. Not surprisingly, we can see that both ESC and the randomized schedule adjustment outperform the global synchronization. This demonstrates the advantages of the adaptive scheduling adjustment over the fixed scheduling. To be explicit, the reduction in the communication delay with ESC and the randomized adjustment, when compared with the global synchronization, is around 30%–40% and 50%–60%, respectively.

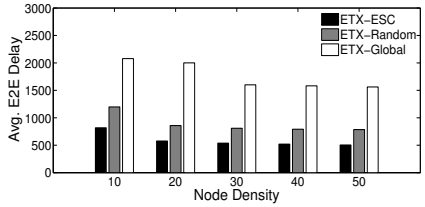


Fig. 25. Comparison with Global Synchronization

9.6. Impact of MAC Layers

In this section, we study the impact of MAC layer protocol selection on the E2E communication delays for ESC. Specifically, we are interested in seeing the performance difference between sender-initiated MAC protocols (B-MAC [Polastre and Culler 2004]) and receiver-initiated MAC protocols (A-MAC [Dutta et al. 2010]). Figure 24 shows the E2E delay of sender-initiated MAC and receiver-initiated MAC under different data rates in energy-harvesting sensor networks. From Figure 24, we can see receiver-initiated MAC protocols are generally more preferable for low duty cycle energy-harvesting sensor networks, especially when the data rate in the network is high. This is because in low duty cycle networks, a sender of sender-initiated MAC protocols such as B-MAC sends long preambles to ensure successful communications with its intended receiver, and such long preambles suppress other potential communications in the surrounding area. In contrast, a node of receiver-initiated MAC protocols, such as A-MAC, listens only to the probe message from its intended receiver and does not interfere with other potential communications from neighboring nodes. Therefore, we would recommend the adoption of well implemented sender-initiated MAC protocols over widely used sender-initiated MAC protocols for low duty cycle energy-harvesting sensor networks.

9.7. Impact of Node Densities

In this section, we examine the impact of node densities on E2E delay for both ETX and DESS networks with varying energy supplies over time. During the simulation, the energy profiles for specific nodes are the same for ESC and the randomized energy

synchronization scheme to ensure fair comparison. As can clearly be seen from both Figure 21(a) and Figure 21(b), ESC has a much smaller delay than the randomized scheme at all node densities for both ETX and DESS. For example, at node density 30, ETX-ESC has a delay of 538 units of time while ETX-Random has a delay of 809 units of time, which is about 50% larger than the delay for ETX-ESC.

9.8. Impact of Node Harvested Energy

In this section, we study the impact of node harvested energy on E2E delay by applying ESC to ETX and DESS in energy-varying sustainable sensor networks. For this specific evaluation, we tweak the energy-harvesting model so as to get the desired average node harvested energy for each experiment. In both Figure 22(a) and Figure 22(b), we can see that ESC outperforms the randomized scheme at all node harvested energies. As the node harvested energy increases, E2E delays for both ETX and DESS under ESC and the randomized scheme are decreasing. This is because with a higher node energy-harvesting rate in the network, the sleep latency between a sender and a receiver is reduced, as there are more active instances for the receiver to receive incoming packets from sending nodes.

10. RELATED WORK

Several technologies have been developed to extract energy from the environment, including solar, motion, biochemical and vibrational [Meninger et al. 1999; Wright et al. 2000]. Building on those energy-harvesting technologies, researchers have designed various types of platforms to collect ambient energy from the environment with optimal efficiency [Dutta et al. 2006; Rahimi et al. 2003; X. Jiang, J. Polastre and D. Culler 2005; Gorlatova et al. 2009]. To fully utilize the harvested energy and ensure the sustainable operation of sensor node, Kansal et al. [Kansal et al. 2007; Kansal et al. 2004] and Vigorito et al. [Vigorito et al. 2007] have presented both theoretical and experimental results on deciding the appropriate working duty cycle of sensor nodes with information on current energy-harvesting rates. To further study the impact of energy leakage for energy-harvesting sensor networks, the TwinStar system suggests node duty cycles based on user specified lifetime and energy information including energy-harvesting rate, remaining energy in the system and energy leakage rate [Zhu et al. 2009]. To explore collaborative energy management, IDEA [Challen et al. 2010] allows individual nodes to evaluate their own impact on other nodes and enables awareness of the connection between the behavior of each node, the application's energy goals, and system performance.

MAC layer design has been a major focus for supporting low duty cycle networking. In general, existing MAC protocols can be categorized into two categories. One category is synchronous MAC protocols, including S-MAC [Ye et al. 2002], T-MAC [van Dam and Langendoen 2003], RMAC [Du et al. 2007] and DW-MAC [Sun et al. 2008a], which synchronize neighboring nodes in order to align their active or sleeping periods. The other category is asynchronous MAC protocols, including B-MAC [Polastre and Culler 2004], X-MAC [Polastre and Culler 2004], WiseMAC [El-Hoiydi and Decotignie 2004], RI-MAC [Sun et al. 2008b] and A-MAC [Dutta et al. 2010], which allow the sensor node to operate individually with its own working schedule through techniques such as Low-Power-Listening. More recently, some hybrid MAC protocols such as SCP-MAC [Ye et al. 2006], Z-MAC [Rhee et al. 2008] and Funneling-MAC [Ahn et al. 2006] are designed to take advantage of two traditional approaches. On top of MAC protocols that focus on allowing multiple sensor nodes to share the physical medium, ESC aims to capture the most generic many-to-many communication pattern and minimize the cross-traffic delay at individual nodes.

Due to the growing gap between limited energy and increasing energy demand in long-term applications, there has been a surge of research interest in low duty cycle networking. For scenarios with mobile nodes, a number of effective solutions has been proposed for data communication [Lindgren et al. 2004; Spyropoulos et al. 2008; Musolesi et al. 2005]. For scenarios with low duty cycle nodes, by assuming perfect link qualities, both works [Lu et al. 2005] and [Keshavarzian et al. 2006] introduce several techniques for minimizing communication latency while providing energy-efficient periodic node working schedules. To address both low duty cycle and unreliable communication links, Gu and He [Gu and He 2007] introduce a dynamic switch-based forwarding using optimized forwarding sequences. Su et al. [Su et al. 2008] propose both on-demand and proactive algorithms for routing packets in low duty cycle networks. Efficient flooding protocols have been introduced to tackle the challenges in low duty cycle sensor networks [Guo et al. 2009; Wang and Liu 2009]. More recently, Gu et al. [Gu et al. 2009a] studied the delay control for low-duty-cycle sensor networks.

Routing algorithms incorporating the energy-harvesting feature exist in the literature [Lattanzi et al. 2007; Lin et al. 2007; Hasenfratz et al. 2010]. A methodology for assessing the energy efficiency of routing algorithms for energy-harvesting networks is proposed in [Lattanzi et al. 2007]. Lin et al. model and characterize the performance of multihop radio networks in the presence of energy constraints in [Lin et al. 2007]. A modified version of the R-MPRT algorithm is proposed in [Hasenfratz et al. 2010]. These existing results mainly focus on the energy-aware optimal routing design, and our work advances the state-of-the-art by optimizing the network performance in the time dimension, and thus complements the existing achievements.

However, none of those prior works investigates how changing the duty cycle of sensor nodes affects communication performance in sustainable sensor networks and how we can adaptively synchronize node working schedules with specified duty cycle budgets. Acting as a transparent middleware service between the MAC layer and network layer, ESC adjusts only RF activities at individual nodes while taking routing and link quality information from those two layers for minimizing cross-traffic delays. In this work, we advance state-of-the-art solutions for both energy-harvesting and low-duty-cycle sensor networks and provide effective methods of synchronizing node working schedules with varying duty cycle budgets.

11. CONCLUSION

In this work, we reveal that cross-traffic delay through a duty cycled node is determined only by the number of active instances at intervals, partitioned by active instances of predecessor and successor nodes. This allows us to design energy-synchronized control with $\mathcal{O}(d^3)$ time complexity for sustainable networks in which energy supplies and demands are in balance, where d is the node degree in the network. In a low duty cycle network, updating neighbors' working schedules would be slow, leading to inconsistent views on active instances. To address this issue, we investigate the impact of obsolete working schedules on logical link quality and demonstrate the tradeoff between shuffle-based and adjustment-based allocation under different energy variation rates. Our evaluation demonstrates that ESC can effectively reduce delay and increase delivery ratios, while synchronizing radio activity with available energy.

Acknowledgement

This work was supported in part by Singapore-MIT International Design Center IDG31000101, Grant SUTD SRG ISTD 2010 002, SUTD-ZJU/RES/03/2011, and NSF grant CNS-1217791.

REFERENCES

AHN, G.-S., HONG, S. G., MILUZZO, E., CAMPBELL, A. T., AND CUOMO, F. 2006. Funneling-mac: a localized, sink-oriented mac for boosting fidelity in sensor networks. In *SenSys '06*.

ALLAART, P. C. AND KAWAMURA, K. 2006. Extreme Values of some Continuous Nowhere Differentiable Functions. *Math. Proc. Camb. Phil. Soc 140,* 2.

CC2420 2012. *CC2420 Product Information and Data Sheet*. CC2420 2012. Available at http://www.ti.com.cn/product/cn/cc2420, 2012.

CHALLEN, G. W., WATERMAN, J., AND WELSH, M. 2010. Idea: integrated distributed energy awareness for wireless sensor networks. In *MobiSys '10*.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms*. MIT Press.

COUTO, D. S. J. D., AGUAYO, D., BICKET, J., AND MORRIS, R. 2003. A High Throughput Path Metric for Multi-Hop Wireless Routing. In *MOBICOM'03*.

DU, S., SAHA, A. K., AND JOHNSON, D. B. 2007. Rmac: A routing-enhanced duty-cycle mac protocol for wireless sensor networks. In *INFOCOM'07*.

DUTTA, P., DAWSON-HAGGERTY, S., CHEN, Y., LIANG, C.-J. M., AND TERZIS, A. 2010. Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless. In *SenSys '10*.

DUTTA, P., HUI, J., JEONG, J., KIM, S., SHARP, C., TANEJA, J., TOLLE, G., WHITEHOUSE, K., AND CULLER, D. 2006. Trio: Enabling Sustainable and Scalable Outdoor Wireless Sensor Network Deployments. In *IPSN'06*.

EL-HOIYDI, A. AND DECOTIGNIE, J.-D. 2004. Wisemac: an ultra low power mac protocol for the downlink of infrastructure wireless sensor networks. *IEEE Symposium on Computers and Communications 1*.

EU, Z. A., TAN, H. P., AND SEAH, W. K. G. 2010. Opportunistic routing in wireless sensor networks powered by ambient energy harvesting. *Computer Networks 54,* 17, 2943–2966.

FAFOUTIS, X. AND DRAGONI, N. 2011. Odmac: An on-demand mac protocol for energy harvesting wireless sensor networks. In *Proc. of ACM PE-WASUN'11*.

GELAL, E., JAKLLARI, G., KRISHNAMURTHY, S. V., AND YOUNG, N. E. 2005. Topology control to simultaneously achieve nearoptimal node degree and low path stretch in ad hoc networks. In *In IEEE SECON05*.

GORLATOVA, M., KINGET, P., KYMISSIS, I., RUBENSTEIN, D., WANG, X., AND ZUSSMAN, G. 2009. Challenge: ultra-low-power energy-harvesting active networked tags (enhants). In *MobiCom '09*.

GU, Y. AND HE, T. 2007. Data Forwarding in Extremely Low Duty-Cycle Sensor Networks with Unreliable Communication Links. In *SenSys '07*.

GU, Y., HE, T., LIN, M., AND XU, J. 2009a. Spatiotemporal delay control for low-duty-cycle sensor networks. In *RTSS*.

GU, Y., ZHU, T., AND HE, T. 2009b. ESC: Energy Synchronized Communication in Sustainable Sensor Networks. In *ICNP '09*.

GUI, C. AND MOHAPATRA, P. 2004. Power Conservation and Quality of Surveillance in Target Tracking Sensor Networks. In *MobiCom'04*.

GUO, S., GU, Y., JIANG, B., AND HE, T. 2009. Opportunistic Flooding in Low-Duty-Cycle Wireless Sensor Networks with Unreliable Links. In *MobiCom '09*.

HASENFRATZ, D., MEIER, A., MOSER, C., JIA CHEN, J., AND THIELE, L. 2010. Analysis, comparison, and optimization of routing protocols for energy harvesting wireless sensor networks. *Proc. of SUTC/UMC*.

HE, S., CHEN, J., YAU, D., SHAO, H., AND SUN, Y. 2009. Energy-efficient capture of stochastic events by global- and local-periodic network coverage. In *MobiHoc*.

HOANG, A. T. AND MOTANI, M. 2007. Collaborative broadcasting and compression in cluster-based wireless sensor networks. *ACM TOSN 3,* 3.

KANSAL, A., HSU, J., ZAHEDI, S., AND SRIVASTAVA, M. B. 2007. Power Management in Energy Harvesting Sensor Networks. *TECS 6,* 4.

KANSAL, A., POTTER, D., AND SRIVASTAVA, M. B. 2004. Performance Aware Tasking for Environmentally Powered Sensor Networks. In *SIGMETRICS '04*.

KESHAVARZIAN, A., LEE, H., AND VENKATRAMAN, L. 2006. Wakeup Scheduling in Wireless Sensor Networks. In *MobiHoc*.

LANGENDOEN, K. AND MEIER, A. 2010. Analyzing MAC protocols for low data-rate applications. *ACM Trans. on Sens. Netw. 7,* 1.

LATTANZI, E., REGINI, E., ACQUAVIVA, A., AND BOGLIOLO, A. 2007. Energetic sustainability of routing algorithms for energy-harvesting wires sensor networks. *Computer Communications 30,* 14–15.

LIN, L., SHROFF, N. B., AND SRIKANT, R. 2007. Asymptotically optimal energy-aware routing for multihop wireless networks with renewable energy sources. *IEEE/ACM Trans. on Netw. 15,* 5.

LINDGREN, A., DORIA, A., AND SCHELEN, O. 2004. Probabilistic routing in intermittently connected networks. *LNCS*, 239–254.

LIU, R.-S., FAN, K.-W., ZHENG, Z., AND SINHA, P. 2011. Perpetual and fair data collection for environmental energy harvesting sensor networks. *Networking, IEEE/ACM Transactions on PP,* 99, 1.

LU, G., SADAGOPAN, N., KRISHNAMACHARI, B., AND GOEL, A. 2005. Delay Efficient Sleep Scheduling in WIreless Sensor Networks. In *INFOCOM'05*.

MAROTI, M., KUSY, B., SIMON, G., AND LEDECZI, A. 2004. The Flooding Time Synchronization Protocol. In *SenSys'04*.

MENINGER, S., MUR-MIRANDA, J., AMIRTHARAJAH, R., CHANDRAKASAN, A., AND LANG, J. 1999. Vibration-to-electric Energy Conversion. In *ISLPED*.

MUSOLESI, M., HAILES, S., AND MASCOLO, C. 2005. Adaptive routing for intermittently connected mobile ad hoc networks. In *WoWMoM'05*.

PARK, C. AND CHOU, P. 2006. AmbiMax: Autonomous Energy Harvesting Platform for Multi-Supply Wireless Sensor Nodes. In *SECON'06*.

POLASTRE, J. AND CULLER, D. 2004. Versatile Low Power Media Access for Wireless Sensor Networks. In *SenSys'04*.

RAHIMI, M., SHAH, H., SUKHATME, G., HEIDEMANN, J., AND ESTRIN, D. 2003. Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network. In *ICRA'03*.

RHEE, I., WARRIER, A., AIA, M., MIN, J., AND SICHITIU, M. 2008. Z-mac: A hybrid mac for wireless sensor networks. *IEEE/ACM Transactions on Networking 16,* 3.

SONG, W.-Z., HUANG, R., XU, M., MA, A., SHIRAZI, B., AND LAHUSEN, R. 2009. Air-dropped sensor network for real-time high-fidelity volcano monitoring. In *MobiSys '09*.

SPYROPOULOS, T., PSOUNIS, K., AND RAGHAVENDRA, C. 2008. Efficient routing in intermittently connected mobile networks: The multiple-copy case. *IEEE/ACM TON 16,* 1, 77–90.

STECK, J. AND ROSING, T. 2009. Adapting performance in energy harvesting wireless sensor networks for structural health monitoring applications. In *IWSHM*.

SU, L., LIU, C., SONG, H., AND CAO, G. 2008. Routing in intermittently connected sensor networks. In *ICNP*.

SUN, Y., DU, S., GUREWITZ, O., AND JOHNSON, D. B. 2008a. Dw-mac: a low latency, energy efficient demand-wakeup mac protocol for wireless sensor networks. In *MobiHoc '08*.

SUN, Y., GUREWITZ, O., AND JOHNSON, D. B. 2008b. Ri-mac: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks. In *SenSys '08*.

VAN DAM, T. AND LANGENDOEN, K. 2003. An adaptive energy-efficient mac protocol for wireless sensor networks. In *SenSys '03*.

VIGORITO, C., GANESAN, D., AND BARTOEEEE, A. 2007. Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks. In *SECON'07*.

WANG, F. AND LIU, J. 2009. Duty-cycle-aware broadcast in wireless sensor networks. In *INFOCOM'09*.

WRIGHT, S., SCOTT, D., HADDOW, J., AND ROSEN, M. 2000. The Upper Limit to Solar Energy Conversion. In *IECEC*.

WU, Y., FAHMY, S., AND SHROFF, N. B. 2007. Energy Efficient Sleep/Wake Scheduling for Multi-Hop Sensor Networks: Non-Convexity and Approximation Algorithm. In *INFOCOM*.

X. JIANG, J. POLASTRE AND D. CULLER. 2005. Perpetual Environmentally Powered Sensor Networks. In *IPSN'05*.

YE, W., HEIDEMANN, J., AND ESTRIN, D. 2002. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *INFOCOM*.

YE, W., SILVA, F., AND HEIDEMANN, J. 2006. Ultra-low duty cycle mac with scheduled channel polling. In *SenSys '06*.

ZAMALLOA, M., SEADA, K., KRISHNAMACHARI, B., AND HELMY, A. 2008. Efficient geographic routing over lossy links in wireless sensor networks. *ACM TOSN 4,* 3.

ZHAO, J. AND GOVINDAN, R. 2003. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *Sensys 03*.

ZHU, T., ZHONG, Z., GU, Y., HE, T., AND ZHANG, Z.-L. 2009. Leakage-Aware Energy Synchronization for Wireless Sensor Networks. In *MobiSys '09*.

ZUNIGA, M. AND KRISHNAMACHARI, B. 2004. Analyzing the Transitional Region in Low Power Wireless Links. In *IEEE SECON'04*.