Ad Hoc Networks 11 (2013) 1202–1220

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Energy-neutral scheduling and forwarding in environmentally-powered wireless sensor networks

Alvin C. Valera [a,b,*], Wee-Seng Soh [b], Hwee-Pink Tan [a]

[a] Sense and Sense-abilities Programme, Institute for Infocomm Research, Singapore 138632, Singapore
[b] Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576, Singapore

ARTICLE INFO

Article history:
Received 9 July 2012
Received in revised form 20 November 2012
Accepted 11 January 2013
Available online 23 January 2013

Keywords:
Energy-harvesting
Wireless sensor network
Dynamic duty cycling
Dynamic wakeup scheduling
Sleep latency
Routing

ABSTRACT

In environmentally-powered wireless sensor networks (EPWSNs), low latency wakeup scheduling and packet forwarding is challenging due to dynamic duty cycling, posing time-varying sleep latencies and necessitating the use of dynamic wakeup schedules. We show that the variance of the intervals between receiving wakeup slots affects the expected sleep latency: when the variance of the intervals is low (high), the expected latency is low (high). We therefore propose a novel scheduling scheme that uses the *bit-reversal permutation sequence* (BRPS) – a finite integer sequence that positions receiving wakeup slots as evenly as possible to reduce the expected sleep latency. At the same time, the sequence serves as a compact representation of wakeup schedules thereby reducing storage and communication overhead. But while low latency wakeup schedule can reduce per-hop delay in ideal conditions, it does not necessarily lead to low latency end-to-end paths because wireless link quality also plays a significant role in the performance of packet forwarding. We therefore formulate *expected transmission delay* (ETD), a metric that simultaneously considers sleep latency and wireless link quality. We show that the metric is left-monotonic and left-isotonic, proving that its use in distributed algorithms such as the distributed Bellman–Ford yields consistent, loop-free and optimal paths. We perform extensive simulations using real-world energy harvesting traces to evaluate the performance of the scheduling and forwarding scheme.

1. Introduction

An environmentally-powered wireless sensor network (EPWSN) is a network consisting of sensor nodes powered by energy harvested from the environment. EPWSN recently gained traction due to breakthroughs in energy harvesting technologies and ultra low power computing and communication devices [1–4]. A major appeal of EPWSN is its potential to address the problem of limited lifetime which is a major drawback of battery-powered wireless sensor network (WSN). By powering nodes with renewable energy, EPWSN can operate perpetually without the need for battery replacement which is not only laborious or expensive but also infeasible in certain scenarios.

But while EPWSN can theoretically enable perpetual network operation, the use of energy harvesting poses a major constraint on energy availability: the amount of energy available for consumption at any given instant is unpredictable and changes significantly over time [5–7]. Thus, unlike battery-powered WSN where the aim is to minimize energy consumption [8], the key objective in EPWSN is to efficiently utilize available energy. The new guiding principle in EPWSN is *energy neutral operation*, which consists of two simultaneous goals: (i) optimizing

* Corresponding author at: Sense and Sense-abilities Programme, Institute for Infocomm Research, Singapore 138632, Singapore.
E-mail addresses: acvalera@i2r.a-star.edu.sg (A.C. Valera), elesohws@nus.edu.sg (W.-S. Soh), hptan@i2r.a-star.edu.sg (H.-P. Tan).

1570-8705/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2013.01.005

the network performance while (ii) ensuring that energy supply and energy demand are balanced [5,6,9,7,10].

To achieve energy neutral operation in the face of unpredictable and dynamic energy availability, adaptive duty cycling algorithms have been proposed [5,6,9,10]. These algorithms aim to dynamically adjust a node's duty cycle given its energy supply, energy buffer capacity as well as current and predicted future harvesting rates. Duty cycled operation necessitates the use of wakeup schedules which specify the times at which a node listens for transmissions from its neighbors [11–15].

A major challenge in both static and dynamic duty cycling networks is *sleep latency* which is the delay incurred when a transmitting node must wait for the receiving node to wakeup before it can commence packet transmission [16,14,17]. In battery-powered WSN where duty cycles are static, static wakeup schedules that minimize sleep latency and end-to-end delay can be computed prior to the operation of the network [14,15]. These pre-computed and fixed wakeup schedules cannot provide optimal performance in EPWSN where duty cycles are dynamic and vary from node to node. Essentially, dynamic wakeup scheduling needs to be adopted wherein every node must compute its low latency wakeup schedule according to its instantaneous duty cycle.

In this paper, we introduce an energy neutral scheduling and forwarding scheme for EPWSN that aims to reduce end-to-end delay due to sleep latency while maintaining low computational complexity and low communication and storage overheads. The major contributions of this work are as follows:

- We show analytically that the expected sleep latency of a wakeup schedule is related to the variance of the intervals between receiving wakeup slots. In particular, when the variance of the interval is low (high), the expected latency is low (high). Hence, the ideal scheduling scheme is the one where the receiving wakeup slots are positioned at equal intervals since its variance is 0.
- To reduce the overhead for storing and exchanging schedules, we propose sequence-based scheduling – a compact representation of dynamic wakeup schedules where the receiving wakeup slots can be obtained using an integer sequence. We design a sequence-based scheduling scheme that uses *bit-reversal permutation sequence* (BRPS) and analytically obtain its worst-case sleep latency which is slightly worse than the ideal scheme but better than schemes where the receiving wakeup slots are spaced uniformly or exponentially.
- To enable the selection of low latency and high reliability paths, we formulate a metric called *expected transmission delay* (ETD), which simultaneously considers sleep latency (due to duty cycling), and wireless link quality. We show that the metric is left-monotonic and left-isotonic, proving that its use in distributed algorithms such as the distributed Bellman–Ford will yield consistent, loop-free and optimal paths.
- We evaluate our design in simulations using real energy harvesting traces to verify and compare the performance of the scheduling and forwarding scheme with existing schemes.

The rest of the paper is organized as follows: Section 2 presents a review of related work and challenges in EPWSN while Section 3 discusses the models and assumptions that are used in the development of the proposed scheme. Sections 4 and 5 provide a detailed presentation of the scheduling and forwarding schemes, respectively. The simulation models and parameters are discussed in Section 6 while the simulation results are presented in Section 7. We finally state our conclusions and future work in Section 8.

2. Related work and motivation

Numerous data delivery protocols have been proposed for battery-powered WSN [18]. The key objective of these protocols is to conserve as much energy as possible, knowing that the energy supply is finite and will eventually be depleted [8]. Evidently, this design principle is not suitable for EPWSN where energy supply can be replenished. In this section, we briefly discuss why dynamic duty cycling is needed in EPWSN and the challenges that it poses to wakeup scheduling and packet forwarding.

2.1. Dynamic duty cycling

The new guiding principle in EPWSN is the notion of *energy neutral operation*, which means operating the nodes such that energy supply and energy demand are balanced [5,6,9,7,10]. To achieve energy neutral operation in the face of dynamic energy availability, adaptive duty cycling algorithms have been proposed [5,6,9,10]. Adaptive duty cycling algorithms aim to dynamically adjust a node's duty cycle given its energy supply, energy buffer capacity and current harvesting rate. Some algorithms also consider future harvesting rates, in which case an accurate prediction scheme (e.g. [19,20]) needs to be employed to ensure optimal algorithm performance. Duty cycling itself is not new and has been proposed as an energy conservation method in battery-powered WSN because radio transceivers consume significant amounts of energy even when idle [21,14,22,17,15].

2.2. Wakeup scheduling

In duty cycled operation, a node "sleeps" (powers off its transceiver) and "wakes up" (powers on its transceiver) according to a schedule[1]. A node's wakeup schedule can be generated using either *random* [11–13] or *coordinated* [14,15] scheduling. In the former approach, a node randomly and independently generates its schedule without considering the schedules of other nodes. In the latter approach, a node considers the schedule of the other nodes in the network in the computation of its own schedule such that a performance metric is optimized. Sleep scheduling can either be *locally* or *globally* coordinated. Locally coordinated

[1] Duty cycling can also be applied to other aspects of a sensor node including its sensors. But in this paper, our focus is on duty cycling of radio transceivers which affects network connectivity and consequently packet forwarding.

scheduling only considers local schedules while globally coordinated scheduling considers the schedules of all the other nodes in the entire network.

While numerous wakeup scheduling schemes have been proposed for fixed duty cycling networks (e.g., [21,14,22,15]), few have been proposed for dynamic duty cycling networks [7]. Compared with [7], our proposed scheme does not require a fixed routing graph and can therefore be used in conjunction with any routing protocol. Furthermore, our scheme has low computational complexity and requires lower communication and storage overhead.

2.3. Energy neutral operation and its challenges

The use of dynamic duty cycling to support energy neutral operation in EPWSN poses several challenges to scheduling and data delivery schemes. In this paper, we specifically tackle the following two major challenges:

2.3.1. Dynamic wakeup scheduling

Sleep latency is a delay incurred in duty cycling networks due to the fact that a transmitting node must wait for the receiving node to wakeup before it can commence packet transmission [16,14,17]. In battery-powered WSN where duty cycles are static, the problem of sleep latency is easily tackled by performing optimization prior to the operation of the network [14,15]. However, this is not possible in EPWSN where duty cycles are dynamic and vary from node to node. We therefore need to devise a dynamic wakeup scheduling scheme (wherein every node computes its wakeup schedule according to its instantaneous duty cycle) that can reduce sleep latency to the least extent possible. In addition, the scheme must also address the following major issues: (i) For the scheme to be useful in practice, it must be designed to be executed in a distributed manner, i.e., in each of the EPWSN nodes. Given their limited computational power, the computational complexity of the scheduling scheme must therefore be kept as low as possible. (ii) When a node's schedule changes, it must be conveyed to its neighbors. Every node must also store the schedules of its neighbors so that it can schedule packet transmissions correctly. As EPWSN nodes are bandwidth- and memory-constrained, the overhead in the exchange of schedules as well as their storage in memory must be kept as low as possible.

2.3.2. Low latency and reliable forwarding

When a path needs to traverse multiple hops, determining the path that provides the least delay can be difficult because the nodes may have different duty cycles and therefore pose different sleep latencies. Existing metrics (e.g., hop count, ETX [23]) do not consider sleep latency even though it is known to be a significant factor in the high end-to-end latency in EPWSN. An equally important criteria in path selection is reliability which is mainly determined by the quality of the wireless links along a path. We therefore need to formulate a metric that simultaneously considers sleep latency and link quality to enable the selection of low latency and high reliability end-to-end paths.

3. Models and assumptions

3.1. Network model

We consider an EPWSN composed of N static nodes. No assumptions are made on the deployment or distribution of the nodes over the area of interest so long as the resulting network is connected. Every node is assigned a unique identifier and has a finite queue which is used for storing packets that need to be forwarded. For easy reference, we provide a summary of the notational conventions used in this paper in Table 1.

3.1.1. Application and traffic model

We consider an environmental monitoring application wherein the sensor network is tasked to monitor the environmental conditions (e.g., temperature, humidity, air quality) of an area of interest. Every node performs periodic readings every T_s which are sent to a common data collection point. Similar to prior work [14,17,15], we assume that the data generation rate is low (i.e., T_s is large) and does not cause significant congestion. Note that this assumption is reasonable as certain environmental monitoring applications require at most 1 reading every 5 min [24].

3.1.2. Cross-layer implementation approach

The proposed scheme has two main components: (i) a dynamic wakeup scheduling scheme; and (ii) a packet forwarding algorithm. In terms of implementation, the components are more appropriately implemented in separate layers. The first component can be implemented in the data link layer on top of a MAC protocol while the second component and can be implemented in the network layer.

3.1.3. Slot synchronization

The proposed scheme operates in a slotted fashion, thereby requiring slot synchronization or alignment. The slot duration is defined prior to the operation of the network and is given by $\tau = \frac{T}{S}$, where T is the duration of one cycle and S is the number of slots in a cycle. τ is defined such that it can accommodate the transfer of one maximum-length data packet and a corresponding ACK packet.

Table 1
Summary of notations used in the paper.

Notation	Meaning
S	Number of slots in one cycle
T	Duration of one cycle
$\tau = \frac{T}{S}$	Duration of one slot
u, v, w	Variables used to denote nodes
k	Variable used to denote a cycle or epoch
$\delta_v(k)$	Duty cycle of v on the kth cycle
$n_v(k)$	Number of receive slots in v's schedule on the kth cycle
$\mathbf{\Gamma}_v(k)$	Receiving wakeup schedule of v on the kth cycle
$\mathbf{N}_v$	Single-hop neighbors of v
S_{vt}	Successor node at v to sink node t

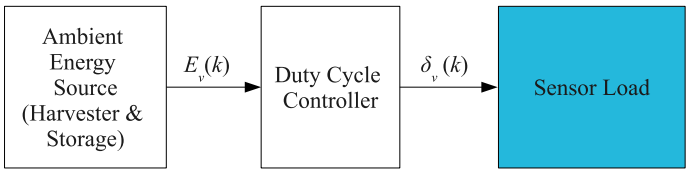


Fig. 1. Node model showing the duty cycle controller, which takes energy level $E_v(k)$ from the ambient energy source as input and provides the duty cycle $\delta_v(k)$ as output.

3.1.4. Medium access control and link estimation

The proposed scheme requires the underlying MAC to support broadcast and two-way handshake packet transmission (unicast data followed by ACK). If the sending node fails to receive an ACK, the MAC layer informs the network layer of the failure. In addition, we assume that the underlying transceiver provides a link quality estimate. Note that this assumption is reasonable since in the IEEE 802.15.4 standard [25], link quality indication (LQI) is a feature required on radio transceivers.

3.1.5. Network initialization

As the proposed scheme focuses on the design of energy neutral wakeup scheduling and forwarding, it assumes that the network has been initialized, i.e., (i) every node v is assumed to have established its set of one-hop neighbors $\mathbf{N}_v$; and (ii) nodes have aligned their slots. We will not propose schemes for accomplishing these functions but the reader can refer to existing work on neighborhood discovery [26] and slot synchronization [27–29].

3.2. Energy-harvesting node model and duty cycle

Every node, except for the sink node, is powered by ambient energy and uses adaptive duty cycling [5,6,9,7,10] to optimize its utilization of available energy for communication. The sink node does not perform duty cycling (i.e., it is always awake) and has unlimited energy supply.

3.2.1. Energy-harvesting node model

Fig. 1 illustrates the energy-harvesting node model used in this paper. The duty cycle controller model, which is inspired by Vigorito et al. [9], requires the energy level of the buffer $E_v(k)$ as input and provides the duty cycle $\delta_v(k) \in [0,1]$ as control output.[2] In simple terms, the duty cycle controller [9] chooses $\delta_v(k)$ as the duty cycle that minimizes $|E_v^*(k) - E_v(k)|$, where $E_v^*(k)$ is a preset target energy level. (A more detailed discussion of this model is presented in Section 6.2.) In general, it can be said that the resulting duty cycle $\delta_v(k) \propto E_v(k)$.

3.2.2. Duty cycle

As mentioned, the duty cycle $\delta_v(k)$ indicates the fraction of time that v can be active at k, i.e., all of its components (microcontroller, radio, sensors and miscellaneous peripherals) are powered up. To emphasize, this means that each of the components can have a duty cycle of $\delta_v(k)$. As our objective in this study is to perform wakeup scheduling of the radio, from hereon, we will only consider $\delta_v(k)$ as allocated to the radio component.

3.2.3. Transmit and receive slots allocation

At the kth cycle, the total amount of time for the radio of v to be active is $T\delta_v(k)$. As v must also perform relaying and not just transmitting its own readings, it must allocate a fraction of its active time for reception as well. We propose the following simple allocation: First, v reserves a certain amount of time for transmitting its own readings. If there is excess time, v divides the remaining time such that the number of transmit and receiving wakeup slots are equal. The rationale behind this is to ensure that v will have a chance to forward all packets generated (its own readings) and received within a cycle. Note that if v allocates less transmit slots and more receive slots, then packets will potentially accumulate if v receives more packets than its transmit slots. Let $n_v(k)$ be the number of receive slots of v at k. If T_s is the sensing interval, then at every cycle v has an average of $\frac{T}{T_s}$ readings which requires $\frac{T}{T_s}\tau$. Whereas, the time needed for receiving and transmitting transit packets is $2n_v(k)\tau$. Then $T\delta_v(k) = \frac{T}{T_s}\tau + 2n_v(k)\tau$. Solving for $n_v(k)$ and forcing the result to be an integer, we obtain

$$n_v(k) = \begin{cases} \left\lfloor \frac{T}{2}\left(\frac{\delta_v(k)}{\tau} - \frac{1}{T_s}\right) \right\rfloor & \text{if } \delta_v(k) > \frac{\tau}{T_s}, \\ 0 & \text{if } \delta_v(k) \leqslant \frac{\tau}{T_s}. \end{cases} \quad (1)$$

3.3. Wakeup schedule

A duty-cycled node requires wakeup schedules for data transmission and reception. When node v needs to forward a packet to node w, v needs to know the receiving wakeup schedule of w so that it can wakeup at the appropriate slot in the future to perform the actual transmission. We define the receiving wakeup schedule as follows.

Definition 1 (*Receiving wakeup Schedule*). Every node v has a *receiving wakeup schedule* $\mathbf{\Gamma}_v(k)$ for the kth cycle which contains $n_v(k)$ time slots indicating the times at which v wakes up to listen for transmissions from its neighbors. A receiving wakeup slot is represented by an integer which ranges from 0 to $S-1$. A number $\eta \in \mathbf{\Gamma}_v(k)$ means that v should wake up in the interval $[\eta\tau, (\eta+1)\tau]$ relative to the start of the cycle.

[2] Of course, other controller models (which may require more information such as power requirements of transmission and reception and current harvesting rate) can also be used as long as they can provide the energy neutral duty cycle which indicates the fraction of time that a node can be active.

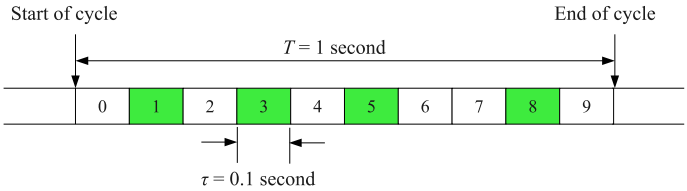


Fig. 2. An example of a receiving wakeup schedule of v where $\Gamma_v(k) = \{1, 3, 5, 8\}$, $S = 10$ and $T = 1$ s. The shaded slots are the time slots where v listens for transmissions from its neighbors.

To clarify this definition, consider the receiving wakeup schedule of node v, $\Gamma_v(k) = \{1, 3, 5, 8\}$, where $S = 10$ and $T = 1$ s as shown in Fig. 2. It must be noted that $\Gamma_v(k)$ only contains the slots at which v wakes up to receive packets from its neighbors. To transmit its packets, v can do so in any unused slot $\beta \notin \Gamma_v(k)$ but must ensure that the intended receiver node w is awake to listen for packet transmissions, i.e., $\beta \in \Gamma_w(k)$.

We complete the discussion on wakeup schedule by considering the following example. Suppose that v (with receiving wakeup schedule given in Fig. 2) intends to transmit a packet to w which has a receiving wakeup schedule of $\Gamma_w(k) = \{1, 3, 4, 7\}$. Node v can choose either slot 4 or 7 to transmit its packet but it cannot choose slots 1 and 3 because they are part of its own receiving wakeup schedule. If v chooses slot 4, then v must transmit the data packet the moment slot 4 begins and should w correctly receive the data packet, it must send back an ACK packet within the same slot. This is possible because the slot duration τ has been defined to accommodate the transfer of one data packet and a corresponding ACK packet.

4. Dynamic wakeup scheduling

We now begin the development of a dynamic wakeup scheduling scheme that can reduce sleep latency to the least extent possible. Because it has to be ultimately executed on sensor nodes which have limited computational power, we want the scheduling scheme to have low computational complexity and low communication and storage overhead. We begin our discussion with a formal definition of sleep latency as it is a key concept in the paper.

Definition 2 (*Sleep Latency*). The sleep latency from node u to v, denoted by W_{uv}, is the delay from the time that a packet becomes ready for transmission at u until the actual packet transmission from u to v. The latency occurs because u must schedule its transmission in the future when v is awake to receive its transmission.

As highlighted in Section 2, sleep latency is a major challenge in both battery-powered and environmentally-powered WSN as it is the main cause of high end-to-end delay. To reduce sleep latency, existing scheduling schemes [14,7,15] perform "tight" coordination wherein the receive slots of v are positioned at times that are close to the receive slots of its successor node w. The key idea is that when v receives a packet in its receive slot, it can quickly forward the packet to w thereby reducing sleep latency. While such an approach may work in perfect conditions, it will face difficulties in situations where link qualities are not ideal. Because of lossy links, v may not be able to successfully transmit at the "nearest" receive slot of w. The retransmission of the lost packet will surely increase its delay which depends on how the receive slots of w are distributed over the cycle. Furthermore, because of the dependence of one node's schedule on another node's schedule, a change in one node may unnecessarily trigger a change in the schedule of other nodes.

We instead propose a "loose" coordination approach wherein the receive slots of v are distributed as evenly as possible within the cycle duration T without regard for the position of the receive slots of its neighbors. The basis of this simple approach is the following lemma which shows that minimizing the variance of intervals between receive slots leads to reduced sleep latency.

Lemma 1 (*Expected Sleep Latency*). *Suppose that n is the number of receiving wakeup slots of node v at a particular cycle. Let D_i denote the ith interval between two successive receiving wakeup slots, where $1 \leqslant i \leqslant n$ (c.f. Fig. 3). Suppose that the packet ready times at some neighbor node u is uniform in [0,T]. Whenever u has a packet to transmit to v, the expected sleep latency from u to v is*

$$E(W_{uv}) = \frac{1}{2}E(D)\left[1 + \frac{V(D)}{E^2(D)}\right], \tag{2}$$

where E(D) and V(D) are the mean and variance, respectively, of all the intervals $\{D_i\}$.

Proof. Suppose that intervals $\{D_i\}$ are independent and identically distributed and let D be a random variable with distribution $F_D(x)$ and that $P(D_i \leqslant x) = F_D(x)$, where $1 \leqslant i \leqslant n$. With $\{D_i\}$ being i.i.d. sequence of positive random variables, we can use results from renewal theory, in particular renewal reward processes (see [30], p. 441) to obtain the expected sleep latency.

Let $N(t) = \sup\{m \geqslant 0 : S_m = D_1 + D_2 + D_3 + \cdots + D_m \leqslant t\}$. The sleep latency from the current time t until the next receiving wakeup slot at node v is simply the residual time $B(t)$ given by

$$B(t) = S_{N(t)+1} - t.$$

Since the packet ready times at node u are uniform, the expected sleep latency from u to v, denoted by $E(W_{uv})$, is equal to

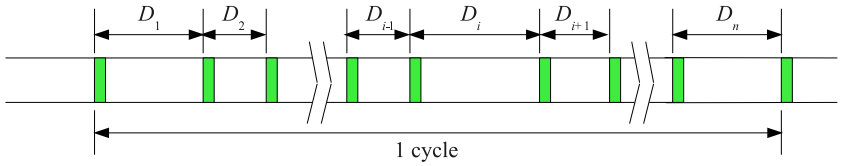


Fig. 3. Model used in the derivation of expected sleep latency. Given a node with n receive slots, D_i is the interval between two successive receiving wakeup slots, where $1 \leqslant i \leqslant n$.

$$E(W_{uv}) = \lim_{t\to\infty} \frac{\int_0^t B(\tau)d\tau}{t}$$

which is the average residual time. To obtain $E(W_{uv})$ using renewal reward theory, we simply associate a reward that is equal to the residual time $B(t)$. Let $R(t)$ denote the total accumulated reward until time t. Then we have

$$R(t) = \int_0^t B(\tau)d\tau.$$

Using Proposition 7.3 in [30] (p. 433), we have

$$E(W_{uv}) = \lim_{t\to\infty} \frac{R(t)}{t} = \frac{E(R)}{E(D)}, \tag{3}$$

where $E(R)$ is the expected reward per renewal cycle or interval and $E(D)$ is the expected duration of an interval. The former quantity can be easily obtained as follows:

$$E(R) = E\left\{\int_0^D (D-t)dt\right\} = \frac{E(D^2)}{2}. \tag{4}$$

Substituting (4) in (3), we have

$$E(W_{uv}) = \frac{E(D^2)}{2E(D)}. \tag{5}$$

Noting that $E(D^2) = E^2(D) + V(D)$, where $V(D)$ is the variance of $\{D_i\}$, we finally obtain (2). □

Note that the result in Lemma 1 is not unique to EPWSN. In fact, a similar result on bus waiting times have been shown in transportation studies [31]. The above result can also be explained by the waiting time paradox (also known as "inspection" paradox) [32] in renewal theory. In terms of packets, the paradox states that it is more likely for a packet to become ready for transmission at a larger interval than a shorter interval. The net effect is that the average waiting time will be higher than the typical value.

For a cycle k with duration T and $n_v(k)$, (5) can be rewritten as

$$E(W_{uv}) = \frac{T}{2n_v(k)}\left(1 + C^2(D)\right), \tag{6}$$

where $C(D) = \frac{\sqrt{V(D)}}{E(D)}$ is the coefficient of variation. Eq. (6) seems to imply that $E(W_{uv})$ can be reduced by decreasing the duty cycle duration T. Recall however from (1) that $n_v(k) \propto T\delta_v(k)$ which really implies that $E(W_{uv}) \propto \frac{1}{\delta_v(k)}$. Hence, given a duty cycle to operate on, the only other way to reduce $E(W_{uv})$ is by reducing the variance or the coefficient of variation of the interval between receive slots.

Using Lemma 1, the ideal schedule is composed of equally-spaced receiving wakeup slots. For a node v with $n_v(k)$ receive slots, the schedule

$$\mathbf{\Gamma}_v(k) = \left\{\left\lfloor \frac{S}{n_v(k)} \right\rfloor n, n = 0, 1, 2, \ldots, n_v(k) - 1\right\} \tag{7}$$

is the most ideal as it provides zero variance. However, one major disadvantage of (7) is that it is not *robust* to changes in $n_v(k)$.

4.1. Schedule robustness

Consider two consecutive cycles k and $k+1$. Suppose that $n_v(k) \neq n_v(k+1)$, then the receive slots in $\mathbf{\Gamma}_v(k+1)$ may be entirely different from the receive slots in $\mathbf{\Gamma}_v(k)$. The implication is that if some other node u fails to receive a schedule update from v, u will transmit at a slot where v is not likely to be awake. We formally define this concept as follows.

Definition 3 (*Schedule Robustness*). Given two cycles k and k', where $k \neq k'$, $n_v(k) \neq n_v(k')$ and $n_v(k),\ n_v(k') \neq 0$, let $\rho(k,k')$ be defined as

$$\rho(k,k') = \frac{|\mathbf{\Gamma}_v(k) \cap \mathbf{\Gamma}_v(k')|}{\min[n_v(k), n_v(k')]}. \tag{8}$$

A schedule $\mathbf{\Gamma}_v$ is robust if for any k and k', $\rho(k,k') = 1$.

Note that in general, $0 \leqslant \rho(k,k') \leqslant 1$. A schedule with $\rho(k,k') = 0$ means that the receive slots in $\mathbf{\Gamma}_v(k)$ are entirely different from $\mathbf{\Gamma}_v(k')$ while a schedule with $\rho(k,k') = 1$ means that the receive slots in $\mathbf{\Gamma}_v(k)$ and $\mathbf{\Gamma}_v(k')$ are the same except for additional slots in one of the schedules.

It is easy to see that the schedule defined in (7) is not robust according to Definition 3. Consider two cycles k and k'. Let $a = \lfloor \frac{S}{n_v(k)} \rfloor$ and $b = \lfloor \frac{S}{n_v(k')} \rfloor$. Without loss of generality, suppose $a > b$ and let m be the least common multiple of the two numbers. Then $\mathbf{\Gamma}_v(k) \cap \mathbf{\Gamma}_v(k') = \{mn, n = 0, 1, 2, \ldots, r-1\}$. We then obtain r as follows. Since $a > b$, then $|\mathbf{\Gamma}_v(k)| < |\mathbf{\Gamma}_v(k')|$. The maximum slot number in $\mathbf{\Gamma}_v(k)$ is therefore the maximum possible common slot in the two schedules. That is,

$$\begin{aligned} m(r-1) &\leqslant a[n_v(k) - 1] \\ r &\leqslant \frac{a[n_v(k)-1]}{m} + 1 \end{aligned} \tag{9}$$

Solving for $\rho(k,k')$ yields

$$\rho(k,k') \leqslant \frac{a}{m} + \frac{1}{n_v(k)}\left(1 - \frac{a}{m}\right),$$

which can only be at most 1 when either $a = m$ (i.e., the schedules are the same) or $n_v(k) = 1$ (i.e., one of the schedules has only one receive slot). The following lemma clarifies why a robust schedule is desirable.

Lemma 2. *Let k and k' be two cycles where $k < k',\ n_v(k) \neq n_v(k')$ and $n_v(k),\ n_v(k') \neq 0$. Consider a node u with knowledge of $\mathbf{\Gamma}_v(k)$ alone and needs to transmit at k'. For a robust schedule, the probability that u transmits at a slot where v is awake, denoted by $P(c)$, is given by*

$$P(c) = \begin{cases} 1 & \text{if } n_v(k) < n_v(k'), \\ \frac{n_v(k')}{n_v(k)} & \text{if } n_v(k) > n_v(k'). \end{cases} \tag{10}$$

Proof. If $n_v(k) < n_v(k')$, the new schedule $\mathbf{\Gamma}_v(k')$ has more slots and that $\mathbf{\Gamma}_v(k) \subset \mathbf{\Gamma}_v(k')$. Since u knows $\mathbf{\Gamma}_v(k)$, it will transmit at $\eta \in \mathbf{\Gamma}_v(k) \Rightarrow \eta \in \mathbf{\Gamma}_v(k')$. Hence $P(c) = 1$.

If $n_v(k) > n_v(k')$ the new schedule $\mathbf{\Gamma}_v(k')$ has fewer slots. Since u knows $\mathbf{\Gamma}_v(k)$ which has $n_v(k)$ slots and that $n_v(k')$ of these slots are contained in $\mathbf{\Gamma}_v(k')$, then $P(c) = \frac{n_v(k')}{n_v(k)}$. □

A subtle implication of Lemma 2 is that if a node u does not know the new schedule of v but has knowledge of its old schedule, u can improve its chances of successfully sending to v by being conservative with its estimate of the number of slots in the new schedule.

4.2. Sequence-based wakeup schedule

From Definition 1, we can view a receiving wakeup schedule $\mathbf{\Gamma}_v(k)$ as a set of integers with cardinality of $n_v(k)$. Exchanging and storing raw schedules (i.e., the entire contents of $\mathbf{\Gamma}_v(k)$) may therefore entail high overhead especially if $n_v(k)$ is high. To address this, we propose a *sequence-based wakeup schedule* pattern for exchanging and storing schedules in a compact manner.

Definition 4 (*Sequence-Based Wakeup Schedule*). Let t_n be an integer sequence that satisfies the following conditions:

(a) $0 \leqslant t_n \leqslant S-1,\ n = 0,1,2,\ldots,S-1$
(b) $t_m \neq t_n,\ \forall m \neq n$

A wakeup schedule of v at k is sequence-based if $\mathbf{\Gamma}_v(k) = \{t_n, n = 0,1,2,\ldots,n_v(k)-1\}$.

Conditions (a) and (b) will ensure that the generated sequence will contain every possible slot number $\eta = 0,1,2,\ldots,S-1$ exactly once. With a sequence-based schedule, $\mathbf{\Gamma}_v(k)$ can be effectively specified by the tuple $\{t_n, n_v(k)\}$. If the sequence t_n is the same for all nodes, then only $n_v(k)$ is needed to completely specify $\mathbf{\Gamma}_v(k)$. Another important property of a sequence-based schedule is that it is robust according to Definition 3. We clarify this fact in the following lemma.

Lemma 3. *A sequence-based wakeup schedule is robust.*

Proof. Let k and k' be two cycles, where $k \neq k'$ and $n_v(k),\ n_v(k') \neq 0$. Without loss of generality, assume that $n_v(k) < n_v(k')$. Then $\mathbf{\Gamma}_v(k) = \{t_n, n = 0,1,2,\ldots,n_v(k)-1\}$ and $\mathbf{\Gamma}_v(k') = \{t_n, n = 0,1,2,\ldots,n_v(k')-1\}$, that is, the two schedules have the same terms up to $n_v(k)$ (since $n_v(k) < n_v(k')$). Hence, $|\mathbf{\Gamma}_v(k) \cap \mathbf{\Gamma}_v(k')| = n_v(k)$. Using (8), $\rho(k,k') = n_v(k)/n_v(k) = 1$ for any k and k'. □

4.3. Bit-reversal permutation sequence

Definition 4 provides two conditions for an integer sequence t_n to be usable as a wakeup schedule generator. If we can obtain such t_n, then the schedule is guaranteed to be robust. We have also shown in Lemma 1 that a schedule with low variance can reduce the expected sleep latency. Thus, aside from satisfying the two conditions in Definition 4, we must also formulate t_n such that the generated schedule yields the minimum variance.

To obtain a suitable sequence, we proceed as follows. Let S be the number of slots in one cycle. If $n_v = 1$ (we drop the parameter k in this discussion as it is clear that we are at a specific cycle), then we can simply decide to position the slot, which we label as $\eta(0)$ at 0. (For the purpose of labeling the receive slots, we use the notation $\eta(i)$, where i is the index.) If $n_v = 2$, then we just add an active slot in the middle of the cycle at $S/2$ which we label $\eta(1)$. If $n_v = 3$, then we add a slot in the middle of $\eta(0)$ and $\eta(1)$. This new slot labeled $\eta(2)$ is at $S/4$. Fig. 4 shows an example up to $n_v = 8$. Observe that for this method to work, $S, S/2, S/4, S/8, \ldots, S/(S/2), S/S$ must be integers. In other words, S must be a power of 2.

Table 2 provides a summary of the slot positions using the method above. We also show in columns 3–5 that when the numerator coefficient is represented in 3-bit binary, its bit-reversal yields the index of the slot label (column 1). The sequence in column 3 can be actually generated using *bit-reversal permutation* [33].

Definition 5. Consider an integer $n \in [0, 2^r - 1]$ with binary representation $(b_{r-1}, b_{r-2}, \ldots, b_1, b_0)$, where $b_i \in \{0,1\}$. Then its bit-reversal in r bits is $(b_0, b_1, \ldots, b_{r-2}, b_{r-1})$. If $B(n,r)$ denotes the decimal value of the bit-reversal of n in r bits, then

$$B(n,r) = \sum_{i=0}^{r-1} b_i 2^{r-1-i}.$$

We are now ready to generalize the method for obtaining the slot position of $\eta(n),\ n = 0,1,2,\ldots,n_v - 1$. Let $S = 2^s$. Suppose that $2^{a-1} \leqslant n_v < 2^a \leqslant 2^s$. Then we have

$$\eta(n) = B(n,a)\frac{S}{2^a} = B(n,a)\frac{2^s}{2^a}.$$

It is straightforward to show that $\eta(n)$ satisfies the conditions in Definition 4. The minimum value of $B(n,a)$ is the bit-reversal of $000\cdots000$ which is 0. Hence the minimum value of $\eta(n)$ is also 0. Whereas, the maximum value of $B(n,a)$ is the bit-reversal of $111\cdots111$ which is $2^a - 1$. Hence the maximum value of $\eta(n)$ is $(2^a - 1)\frac{2^s}{2^a} = 2^s - \frac{2^s}{2^a} \leqslant 2^s - 1$. Also, every n has a unique a-bit representation. Bit-reversal also yields a unique a-bit representation which yields a unique value. Multiplying this value

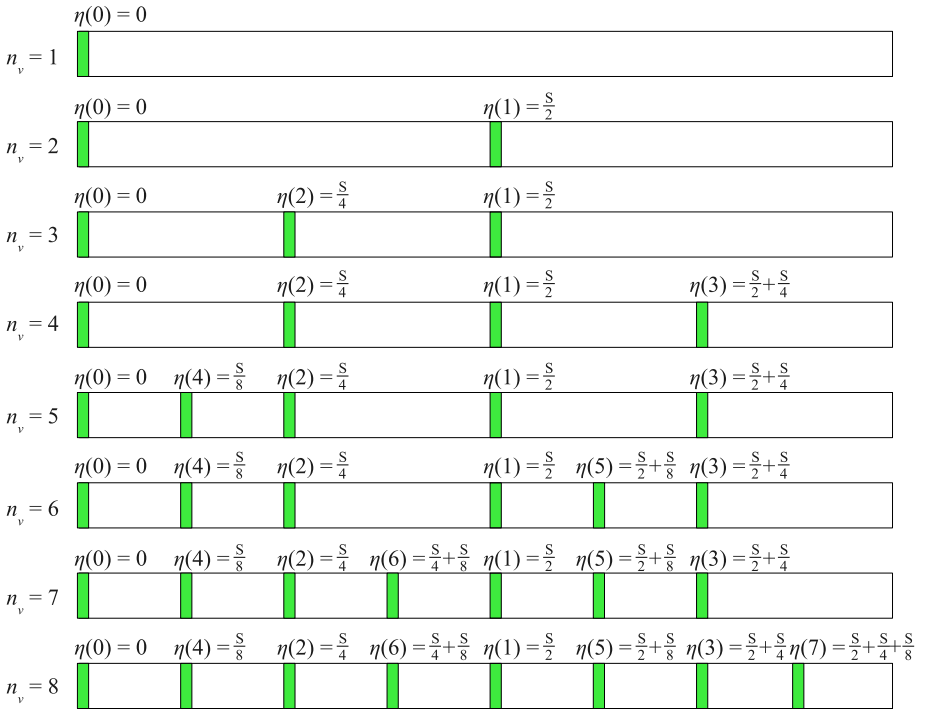


Fig. 4. Illustrating the idea behind the proposed sequence for generating wakeup schedules.

Table 2
Summary of the example sequence for generating schedule.

Label	Slot position	Numerator coefficient	Numerator coefficient (3-bit binary)	Bit reversal and decimal value
$\eta(0)$	0	0	000	$000 \Rightarrow 0$
$\eta(1)$	$\frac{S}{2} = \frac{4S}{8}$	4	100	$001 \Rightarrow 1$
$\eta(2)$	$\frac{S}{4} = \frac{2S}{8}$	2	010	$010 \Rightarrow 2$
$\eta(3)$	$\frac{S}{2}+\frac{S}{4} = \frac{6S}{8}$	6	110	$011 \Rightarrow 3$
$\eta(4)$	$\frac{S}{8}$	1	001	$100 \Rightarrow 4$
$\eta(5)$	$\frac{S}{2}+\frac{S}{8} = \frac{5S}{8}$	5	101	$101 \Rightarrow 5$
$\eta(6)$	$\frac{S}{4}+\frac{S}{8} = \frac{3S}{8}$	3	011	$110 \Rightarrow 6$
$\eta(7)$	$\frac{S}{2}+\frac{S}{4}+\frac{S}{8} = \frac{7S}{8}$	7	111	$111 \Rightarrow 7$

by $\frac{2^s}{2^a}$ does not affect its uniqueness. Hence, $\eta(n)$ is an admissible sequence.

4.3.1. Expected waiting time

We now obtain the expected sleep latency that results when bit-reversal permutation is used to generate a schedule. For conciseness, we drop the parameter k in $n_v(k)$ as it is clear that the results apply to a specific cycle or epoch k.

Theorem 1. *The expected sleep latency incurred using a bit-reversal permutation sequence schedule is*

$$E(W_{uv}) = \frac{T}{2n_v}\left(1 + \frac{1}{2^{2a+1}}(n_v - 2^a)(2^{a+1} - n_v)\right), \tag{11}$$

where $n_v > 0$ is the number of receive slots and $a = \lfloor \log_2 n_v \rfloor$.

Proof. $a = \lfloor \log_2 n_v \rfloor \Rightarrow 2^a \leqslant n_v < 2^{a+1}$. The first 2^a slots divide T equally such that the interval between slots is $T/2^a$. The excess slots $x = n_v - 2^a$ further divides x of the 2^a intervals into 2. Thus, finally there are $2x$ intervals with duration $T/2^{a+1}$ and $n_v - 2x = 2^{a+1} - n_v$ intervals with duration $T/2^a$. Since $E(D) = T/n_v$, the variance $V(D)$ of the intervals is

$$\begin{aligned} V(D) &= \left(\frac{2n_v - 2^{a+1}}{n_v}\right)\left(\frac{T}{2^{a+1}} - \frac{T}{n_v}\right)^2 \\ &\quad + \left(\frac{2^{a+1} - n_v}{n_v}\right)\left(\frac{T}{2^a} - \frac{T}{n_v}\right)^2 \\ &= \left(\frac{1}{2^{2a+1}}\right)\left(\frac{T}{n_v}\right)^2 (n_v - 2^a)(2^{a+1} - n_v). \end{aligned} \tag{12}$$

Solving for $C^2(D) = V(D)/E^2(D) = V(D)/(T/n_v)^2$ and substituting in (6), we obtain (11). □

Note that when n_v is a power of 2, $n_v = 2^a$ which results in the coefficient of variation to vanish. The following corollary further shows that $C^2(D)$ has an upper bound and that $E(W_{uv})$ therefore has an upper bound.

Corollary 1. *The expected sleep latency incurred using a bit-reversal permutation sequence schedule is bounded and* $0.5\frac{T}{n_v} \leqslant E(W_{uv}) \leqslant 0.5625\frac{T}{n_v}$.

Proof. Since $2^a \leqslant n_v < 2^{a+1}$, let $d = 2^{a+1} - 2^a = 2^a$. Then for $0 \leqslant m < 2^a$, we can rewrite $(n_v - 2^a)(2^{a+1} - n_v)$ as $f(m) = (d - m)m = (2^a - m)m = 2^a m - m^2$. Solving for $f'(m)$ and $f''(m)$,

$f'(m) = 2^a - 2m$

and

$f''(m) = -2.$

Since $f''(m) < 0, f(m)$ is concave and setting $f'(m) = 0$ will yield the maximum value of $f(m)$. Doing this gives us $m = 2^{a-1}$. Hence, $n_v = 2^a + 2^{a-1}$ will give the maximum $C^2(D)$. Therefore,

$$C^2(D) = \frac{(2^a + 2^{a-1} - 2^a)(2^{a+1} - 2^a + 2^{a-1})}{2^{2a+1}} = \frac{1}{8} = 0.125.$$

Substituting this in (11), we finally have the upper bound $E(W_{uv}) = 0.5625\frac{T}{n_v}$. □

In Table 3, we compare the upper bound of the expected sleep latency under different distributions. For the uniform distribution, we let the slot intervals range from 0 to $2T/n_v$ such that the expected value is still T/n_v. The coefficient of variation of exponential distribution is always 1. Note that bit-reversal permutation sequence schedule yields lower expected sleep latency compared with uniform or exponential.

4.3.2. Per node sequence

The disadvantage of using the same sequence $\eta(n)$ in all of the nodes is that they will have common wakeup schedules. This is not a desirable situation because more nodes may transmit at the same slot even though their intended receivers are different. Although traffic flow is assumed to be low, this may still result in higher occurrence of packet collisions. To reduce common receiving wakeup slots across nodes, we introduce an offset to $\eta(n)$ for every node v as follows:

$$t_n = [v + \eta(n)] \mod S,$$

where v is the ID of node v. The modulo operation is necessary to ensure that t_n does not exceed $S - 1$.

4.3.3. Computational complexity

One of the key advantages of the proposed scheme is its low computational complexity. As a matter of fact, it does not require the computation of any schedule as it only needs to compute the number of receiving wakeup slots. Hence, its complexity is $O(1)$.

4.3.4. Scheduling overhead

In terms of communication overhead, the advantage of a sequence-based wakeup schedule is that for any node v in the network, it only needs to send $n_v(k)$. In terms of storage overhead, the compact representation of the wakeup schedule in terms of $n_v(k)$ requires considerably lower overhead compared to schemes that require the storage of the entire schedule. If v has to store the schedule of every neighbor, then the overhead of a sequence-based wakeup schedule is $O(|\mathbf{N}_v|)$. Whereas, the overhead of schemes that require the storage of entire schedules is $O(|\mathbf{N}_v|S)$, where S is the number of slots in the cycle.

Table 3
Upper bound of the expected sleep latency under different distributions.

Slot distribution	Upper bound
Ideal (equally-spaced slots)	$0.5\frac{T}{n_v}$
Bit-reversal permutation sequence	$0.5625\frac{T}{n_v}$
Uniform	$0.6667\frac{T}{n_v}$
Exponential	$\frac{T}{n_v}$

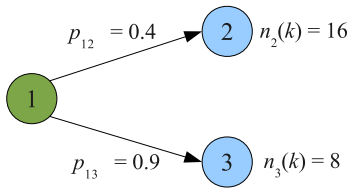


Fig. 5. An example illustrating the adverse effect of neglecting link quality in forwarder selection. p_{vw} denotes the packet delivery probability from v to w.

5. Low latency and reliable forwarding

So far, our development assumed that the links are ideal and that packet transmissions are always successful on the first transmission. However in practical deployments, links in wireless sensor networks are far from being ideal [17]. The impact of lossy links is to essentially increase the expected sleep latency as will be shown in the following motivating example.

5.1. Motivating example

In the example shown in Fig. 5, the number of receiving wakeup slots of nodes 2 and 3 are indicated. p_{vw} denotes the packet delivery probability from v to w. Thus, we can see that the link (1,3) has better quality than link (1,2). If only sleep latency were considered, then the selected forwarder would be node 2 since $E(W_{12}) = \frac{T}{2n_2(k)} = \frac{T}{32} < E(W_{13}) = \frac{T}{2n_3(k)} = \frac{T}{16}$. But because the delivery probability across (1,2) is only 0.4, this implies that a packet needs to be transmitted 1/0.4 = 2.5 on the average[3] before being successfully received by node 2. Hence, the total latency would be roughly $(2.5)E(W_{12}) = \frac{2.5T}{32}$. Whereas, for link (1,3) which has a link quality of 0.9, the total latency would be roughly $(1.1)E(W_{13}) = \frac{2.2T}{32}$.

5.2. Expected transmission delay

The example above clearly shows that considering sleep latency alone in selecting the next hop may result in poor performance. On the other hand, considering link quality alone will also not necessarily lead to the best performance. This is because the next hop node with the highest

[3] This argument is explained as follows. Each transmission attempt across link (u, v) can be considered as a Bernoulli trial. Because the probability of success is p_{uv}, then from elementary probability, the expected number of transmission attempts is simply $1/p_{uv}$.

link quality might have a very high sleep latency. We therefore define a new metric that combines sleep latency and link quality.

Definition 6 (*Transmission Delay*). The transmission delay between adjacent nodes u and v, denoted by $\mathscr{D}_{uv}$, is the delay from the time a packet becomes ready for transmission at u until it is successfully received by v.

We now obtain the expected transmission delay (ETD) $E(\mathscr{D}_{uv})$ from u to v. For a transmission to be deemed successful, the receiving node v must receive the data packet *and* the sending node u must receive the corresponding ACK packet. If p_{uv} and p_{vu} are the two-way delivery probabilities of link (u,v), then a packet is considered successfully transmitted with probability $p_{uv}p_{vu}$. Following the same argument as in [23], the expected number of transmissions is $\frac{1}{p_{uv}p_{vu}}$. Given that for every transmission attempt the delay is $E(W_{uv})$, we have

$$E(\mathscr{D}_{uv}) = \frac{E(W_{uv})}{p_{uv}p_{vu}}. \tag{13}$$

5.2.1. Expected transmission delay of a path

Finally, the ETD of a path $\mathbf{P}$ from s to t, denoted by $E(\mathscr{D}_{st})$, is defined as the expected delay measured from the time a packet becomes ready for transmission at s until it is successfully delivered to t. This is simply given by

$$E(\mathscr{D}_{st}) = \sum_{\forall(u,v)\in\mathbf{P}} E(\mathscr{D}_{uv}). \tag{14}$$

5.2.2. Distributed calculation

The search for the minimum ETD path from every sensor node v to the data collection point t can be performed in a distributed manner using the distributed Bellman–Ford algorithm. However, for such an algorithm to yield consistent, loop-free and minimum cost paths, we have to show that the ETD metric is both left-monotonic and left-isotonic [34].

Lemma 4. *The ETD metric is left-monotonic and left-isotonic.*

Proof. To show left-monotonicity, consider a path A from v to w with cost $E(\mathscr{D}_A)$. Suppose that we prepend a path C from u to v with cost $E(\mathscr{D}_C)$. Then the cost of the path from u to w through (C,A) is $E(\mathscr{D}_{CA}) = E(\mathscr{D}_C) + E(\mathscr{D}_A)$. Clearly, $E(\mathscr{D}_A) \leqslant E(\mathscr{D}_{CA})$, proving its left-monotonicity.

To show left-isotonicity, consider two paths A and B from v to w with costs $E(\mathscr{D}_A)$ and $E(\mathscr{D}_B)$, respectively, and with $E(\mathscr{D}_A) \leqslant E(\mathscr{D}_B)$. Suppose that we prepend a path C from u to v with cost $E(\mathscr{D}_C)$. Then the cost of the path from u to w through (C,A) is $E(\mathscr{D}_{CA}) = E(\mathscr{D}_C) + E(\mathscr{D}_A)$ while the cost of the path from u to w through (C,A) is $E(\mathscr{D}_{CB}) = E(\mathscr{D}_C) + E(\mathscr{D}_B)$. Clearly, $E(\mathscr{D}_{CA}) \leqslant E(\mathscr{D}_{CB})$, proving its left-isotonicity. □

5.3. Protocol overview

We now present the details of a forwarding scheme that incorporates the BRPS and ETD to perform minimum-cost path computation and packet forwarding. As a matter of notation, we maintain the variable naming conventions but affix the node ID as superscript to indicate that a variable is maintained by a node.

5.3.1. State variables and data structures

Every node v maintains two global state variables and a neighbor table. The global variables are S^v_{vt} and $E(\mathscr{D}^v_{vt})$ which are the ID of the next hop node and the minimum ETD, respectively, to the sink node t. The neighbor table contains an information tuple $(n^v_u, E(\mathscr{D}^v_{vu}), E(\mathscr{D}^v_{ut}), p^v_{vu}, p^v_{uv})$ about every node $u \in \mathbf{N}_v$, where n^v_u is the number of receive slots of u, $E(\mathscr{D}^v_{vu})$ is the ETD from v to u, $E(\mathscr{D}^v_{ut})$ is the minimum ETD from u to t, p^v_{vu} is the link delivery probability from v to u and p^v_{uv} is the link delivery probability from u to v.

5.3.2. Initial values

After a node has completed performing neighbor discovery using some suitable protocol (e.g., [26]), it initializes every neighbor information tuple as follows: $n^v_u \leftarrow 0$, $E(\mathscr{D}^v_{vu}) \leftarrow \infty$, $E(\mathscr{D}^v_{ut}) \leftarrow \infty$, $p^v_{vu} \leftarrow 0$, and $p^v_{uv}) \leftarrow 0$. More importantly, it initializes its state variable $E(\mathscr{D}^v_{vt})$ as follows:

$$E(\mathscr{D}^v_{vt}) = \begin{cases} 0 & \text{if } v \text{ is the sink node } t, \\ \infty & \text{otherwise.} \end{cases} \tag{15}$$

5.3.3. Control update and node update slot

At every cycle k, every node v must broadcast an update packet $\text{UPDATE}(n_v(k), E(\mathscr{D}_{vt}(k)), \{p_{uv}(k), \forall u \in \mathbf{N}_v\})$ at a designated *node update slot*. For simplicity, we assign the slot $(v \bmod S)$ as the node update slot of v. Hence, every neighbor of v must wakeup at slot $(v \bmod S)$ to listen for updates from v and must not use that slot for its own packet transmission.

Algorithm 1. Algorithm for processing an update packet.

1: **if** received UPDATE$(n_u(k), E(\mathscr{D}_{ut}(k)), \{p_{wu}(k), \forall w \in \mathbf{N}_u\})$ from u **then**
2: $n^v_u \leftarrow n_u(k)$
3: $E(\mathscr{D}^v_{ut}) \leftarrow E(\mathscr{D}_{ut}(k))$
4: $p^v_{vu} \leftarrow p_{vu}(k)$
5: $p^v_{uv} \leftarrow$ PHY LQI estimate of p_{uv}
6: **else**
7: $n^v_u \leftarrow \lfloor \alpha n^v_u \rfloor$
8: **end if**

5.3.4. Control update processing

When v receives an update packet from u, it updates the corresponding information tuple for u in its neighbor table as indicated in lines (2)–(5) of Algorithm 1. Because the radio transceiver provides link quality information, v can also obtain an estimate of p^v_{uv}. Studies [35,36] have shown that LQI is highly-correlated with packet delivery probability and can therefore be used to obtain the latter. If v does

not receive an update packet from u on its designated slot, v sets $n_u^v \leftarrow \lfloor \alpha n_u^v \rfloor$, where $0 < \alpha < 1$. We refer to the parameter α as the *receive slot discount factor*. By conservatively estimating $n_u(k)$, v is essentially improving its probability of successful transmission to u. This is possible because of the robustness property of the scheduling scheme as shown in Lemma 2.

5.4. Path computation

Algorithm 2 provides a listing of the algorithm that is executed at every node v to obtain the minimum cost path from v to t. The algorithm is executed at the end of every node update slot of every neighbor node u.

Algorithm 2. Distributed algorithm for path computation.

1: **if** $p_{uv}^v = 0$ or $p_{vu}^v = 0$ or $n_u^v = 0$ **then**
2: $E(\mathscr{D}_{vu}^v) \leftarrow \infty$
3: **if** $S_{vt}^v = u$ **then**
4: $E(\mathscr{D}_{vt}^v) \leftarrow \infty$
5: **for** $w \in \mathbf{N}_v \setminus u$ **do**
6: $M \leftarrow E(\mathscr{D}_{vw}^v) + E(\mathscr{D}_{wt}^v)$
7: **if** $M < E(\mathscr{D}_{vt}^v)$ **then**
8: $E(\mathscr{D}_{vt}^v) \leftarrow M$
9: $S_{vt}^v \leftarrow w$
10: **end if**
11: **end for**
12: **end if**
13: **else**
14: $a \leftarrow \lfloor \log_2 n_u^v \rfloor$
15: $E(W_{vu}^v) \leftarrow \frac{T}{2n_u^v}\left(1 + \frac{1}{2^{2a+1}}(n_u^v - 2^a)(2^{a+1} - n_u^v)\right)$
16: $E(\mathscr{D}_{vu}^v) \leftarrow \frac{E(W_{vu}^v)}{p_{uv}^v p_{vu}^v}$
17: $M \leftarrow E(\mathscr{D}_{vu}^v) + E(\mathscr{D}_{ut}^v)$
18: **if** $M < E(\mathscr{D}_{vt}^v)$ **then**
19: $E(\mathscr{D}_{vt}^v) \leftarrow M$
20: $S_{vt}^v \leftarrow u$
21: **end if**
22: **end if**

5.4.1. Infinite ETD to neighbor u

Lines (2)–(12) lists the steps that are executed when the ETD to a neighbor node u becomes infinite. The ETD to u becomes infinite whenever the number of receive slots of u becomes zero (which happens at initial state or after several non-reception of update packets) or link delivery probability estimates are not available. If u is the successor node to t, v searches for an alternative neighbor w that provides the minimum ETD path to t.

5.4.2. Finite ETD to neighbor u

Lines (14)–(21) lists the steps that are executed when the ETD to a neighbor node u is finite. Node v computes the ETD to u using (13). If the resulting ETD through u is less than the current minimum ETD, then u is chosen as the new next hop node and the corresponding ETD is set as the minimum ETD from v to t.

5.4.3. Convergence

Lines (5)–(11) and (18)–(21) ensure that after every execution of Algorithm 2, v either has infinite cost to t or a finite cost which satisfies the following:

$$E(\mathscr{D}_{vt}^v) = \min_{u \in \mathbf{N}_v} \left[E(\mathscr{D}_{vu}^v) + E(\mathscr{D}_{ut}^v)\right] \tag{16}$$

This equation is essentially the update rule of the distributed Bellman–Ford algorithm [37]. Together with the initial values in (15) and neighbor update processing in Algorithm 1, (16) ensures that the computation will converge to the correct minimum cost within finite time [37].

5.5. Packet forwarding

Algorithm 3 lists the packet forwarding algorithm of the protocol. Upon receipt of a data packet P from a neighbor node, from a local application, or from a previous unsuccessful transmission, v checks if it has a path to the sink. If it has no path, v drops P and the algorithm terminates. Otherwise, v checks the number of times that P has been transmitted previously. If P has been transmitted more than L, P is dropped and the algorithm terminates.

Algorithm 3. Packet forwarding algorithm.

1: **if** $E(\mathscr{D}_{vt}^v) = \infty$ **then**
2: Drop P
3: **return**
4: **end if**
5: $r \leftarrow$ Number of retransmissions of P
6: **if** $r \geqslant L$ **then**
7: Drop P
8: **return**
9: **end if**
10: $s \leftarrow$ Current slot
11: $w \leftarrow S_{vt}^v$
12: $\mathbf{\Gamma}_v \leftarrow \{t_n, n = 0, 1, 2, \ldots, n_v(k) - 1\}$
13: $\mathbf{\Gamma}_w \leftarrow \{t_w(n), n = 0, 1, 2, \ldots, n_w^v - 1\}$
14: $t_{\min} \leftarrow \infty$
15: **for** $\eta \in \mathbf{\Gamma}_w$ **do**
16: **if** $\eta = u \bmod S, \forall u \in \mathbf{N}_v$ or $\eta \in \mathbf{\Gamma}_v$ **then**
17: continue
18: **end if**
19: $t_{\text{diff}} \leftarrow \eta - s$
20: **if** $t_{\text{diff}} \leqslant 0$ **then**
21: $t_{\text{diff}} \leftarrow t_{\text{diff}} + S$
22: **end if**
23: bf if $t_{\text{diff}} < t_{\min}$**then**
24: $t_{\min} \leftarrow t_{\text{diff}}$
25: **end if**
26: **end for**
27: **if** $t_{\min} < \infty$ **then**
28: Schedule P for transmission at $s + t_{\min}$
29: **else**
30: Drop P
31: **end if**

5.5.1. Slot search

If P is eligible for transmission, v searches for a time slot that is closest to the current slot in the schedule of the next hop node w. Node v ensures that a chosen transmit slot does not conflict with an update slot from a neighbor node or that the slot is not in v's own schedule. If a slot is found, P is scheduled for transmission at the specified slot $s + t_{\min}$. Otherwise, a *scheduling failure* is considered to have occurred and P is dropped.

5.5.2. Packet transmission

At the appropriate slot, v wakes up to initiate the transmission of the scheduled packet P. The packet is sent unicast to w and v waits for an ACK from w. When an ACK is received, v proceeds to schedule the next packet in the queue, if there is any. When no ACK is received, the number of retries counter r is incremented, and Algorithm 3 is invoked to try to schedule P again.

6. Simulation models

We implemented a simulation model of the proposed wakeup scheduling and forwarding scheme in Qualnet [38], including other required components such as energy harvesting source and duty cycle controller. We set the slot duration τ to be 10 ms which is more than sufficient for the transmission of a 127-byte data packet (the maximum payload of an IEEE 802.15.4 PHY frame is 127 bytes [25]) and ACK.

6.1. Energy harvesting source model

Fig. 6 shows the energy harvesting source model which is composed of a solar panel, a charging circuit, and a supercapacitor. This model captures the state-of-the-art in circuit design for micro-solar energy harvesting systems [39,40]. The model requires solar irradiance data (in Watts per square meter) as input. For this purpose, we used real solar data traces from the National Renewable Energy Laboratory (NREL) [41]. We selected 2 days of solar radiation trace, representing "sunny" and "cloudy" scenarios, which are shown in Fig. 7. For each node, the actual irradiance value used is a random number between the diffuse and global irradiance values.

The solar panel is characterized by its surface area A_{SP} and conversion efficiency η_{SP}. The energy output $E_{SP}(k)$ of the solar panel at the kth epoch is given by $E_{SP}(k) = A_{SP}\eta_{SP}E_s(k)$, where $E_s(k)$ is the solar irradiance. Whereas, the charging circuit is characterized by its charging efficiency η_{CC}. The amount of energy that reaches the supercapacitor is simply the output of the charging circuit which is given by

$$E_{CC}(k) = \eta_{CC}E_{SP}(k) = \eta_{CC}A_{SP}\eta_{SP}E_s(k) = \eta_{EH}E_s(k).$$

We shall refer to $\eta_{EH} = \eta_{CC}A_{SP}\eta_{SP}$ as the *effective harvesting efficiency*. In the simulations, we used $A_{SP} = 0.01\ \text{m}^2$, $\eta_{SP} = 0.1$, and $\eta_{CC} = 0.5$, which yields $\eta_{EH} = 0.0005$. Note that these values are conservative compared to what is currently available or achievable in the literature. For instance, current photovoltaic cells can achieve as high as 25% energy conversion efficiency while state-of-the-art supercapacitor charging circuits can attain as high as 89% efficiency [39]. We also conduct simulations where η_{EH} is varied from 0.0001 to 0.0005. Finally, we used 25 Farad 4 Volt supercapacitor as energy buffer.

6.2. Duty cycle controller model

The duty cycle controller is modeled after the LQ-Tracker algorithm proposed by Vigorito et al. [9]. Our main motivation for using LQ-Tracker is that it represents the state-of-the-art in adaptive duty cycling algorithms and is easily implementable in simulations. We provide a brief description of the controller below while interested readers can refer to [9] for a detailed discussion of the controller.

The controllers's objective is to achieve ENO-Max, which entails two simultaneous objectives: (i) to ensure that energy consumed is always less than or equal to the energy harvested; and (ii) to maximize task performance (energy consumption). To derive the adaptive control law, the authors modeled the dynamics of the battery level as a first order, discrete time, linear dynamical system with colored noise which conforms to

$$y(k+1) = ay(k) + bu(k) + cw(k) + w(k+1),$$

where y is the battery level, u is the control, w is a zero-mean input noise, and a, b, c are real-valued coefficients. The objective of the control system is to minimize the error $|y(k) - y^*|$ for all k, where y^* is the target battery level. Note that this objective is equivalent to minimizing the average squared tracking error

$$\lim_{K\to\infty}\frac{1}{K}\sum_{k=1}^{K}[y(k) - y^*]^2, \tag{17}$$

which is the ENO-Max objective. The optimal control law that minimizes (17) is

$$u(k) = \frac{y^* - (a+c)y(k) + cy^*}{b}. \tag{18}$$

The authors proposed an on-line algorithm based on standard gradient descent techniques to estimate the coefficients a, b, and c.

6.3. Network parameters

The network consists of 200 static nodes that are uniformly-distributed in a 500 m × 500 m area. A single sink node is positioned at (0, 0). Fig. 8a shows the cumulative distribution of hop count to the sink of a typical scenario. We can see that the hop count ranges from 1 to 12 and that a large fraction of nodes are 5–10 hops away from the sink. The positioning of the sink at (0,0) results in a challenging scenario where data traffic converges to a "narrow spot" in the network.

The sensor node is modeled after a Libelium Waspmote equipped with XBee-802.15.4, an IEEE 802.15.4-compliant radio transceiver (see Fig. 8b). The transceiver is configured to send at a data rate of 250 kbps and a transmit power of 0 dBm. This yields a transmit range of approximately 100 m. With these configurations, the node consumes

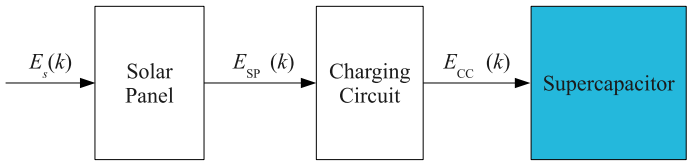


Fig. 6. Solar energy harvesting source model.

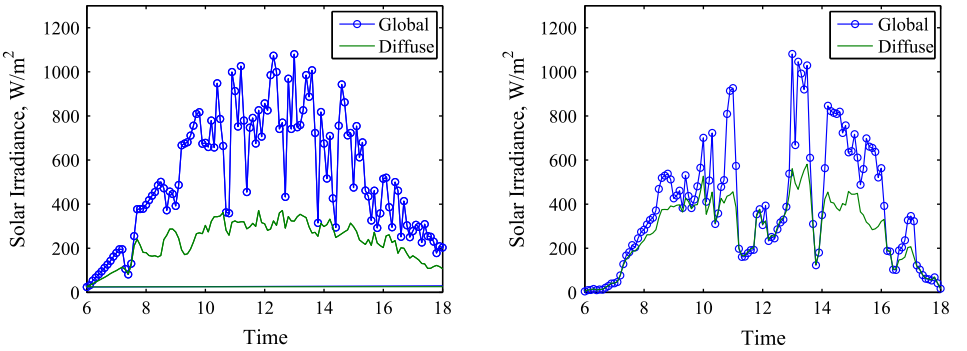


Fig. 7. Solar irradiance data (at 6-min resolution) from NREL Florida Solar Energy Center on July 2 and July 8, 2000.

approximately 180 mW, 195 mW, and 240 μW in transmit, receive (active and idle), and deep sleep modes, respectively at 4 V [42,43].

A constant bit rate (CBR) traffic generator is used to generate data traffic. Each data packet is 64 bytes and every node (except the sink) generates data at intervals of 60, 120, 180, 240, 300 s. These data generation rates are already considered to be high for certain environmental monitoring applications which require around 1 sample every 5 min [24]. Packet generation times are randomly staggered among different nodes. Each data point is obtained by averaging the results from 20 seed values, with every simulation run configured for 43,200 s (12 h) in simulation time.

7. Simulation results

We first evaluate the effect of the three design parameters, namely, the duty cycle duration T, maximum retry limit L, and receive slot discount factor α on the performance of the proposed scheduling and forwarding scheme.

Duty cycle duration T. To determine the effect of T, we varied the number of slots per cycle T/τ, such that $T/\tau \in \{256, 512, 1024, 2048, 4096\}$. Note that these values are exact powers of two as required by the bit-reversal permutation scheduling scheme. The smallest value of 256 is chosen since there are 200 nodes in the network. This

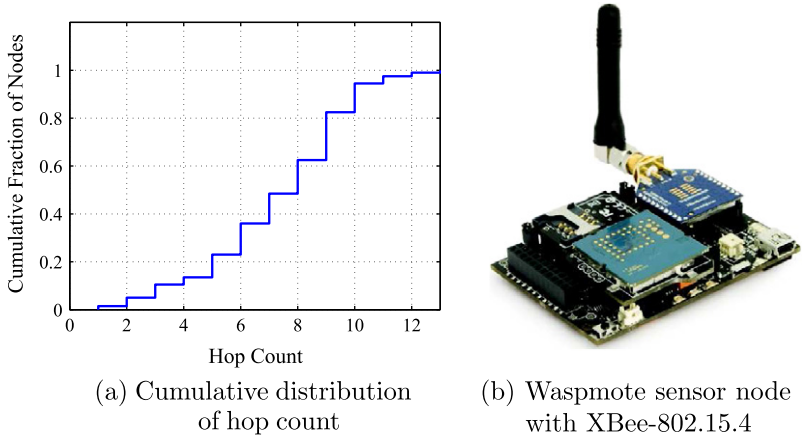


Fig. 8. Cumulative hop count distribution of a typical scenario and sensor node model.

ensures that every node will have its own slot for transmitting its update packet.

We first study the convergence time of the distributed path computation algorithm. To measure the convergence time, we forced the nodes to use fixed duty cycles throughout the simulation. The convergence time is affected by T since each node sends updates at an interval equal to T. As shown in Fig. 9a, the algorithm takes longer time to converge at higher T since the propagation of updates is slower.

However, one advantage of using a larger T is that it reduces the occurrence of scheduling failures (see Fig. 9b). As discussed in Section 5.5, a scheduling failure occurs when the forwarding algorithm fails to find a receiving wakeup slot at the intended receiver that does not conflict with neighbor update slots. Note that when T is small (i.e., number of slots per cycle is small), there are fewer receiving wakeup slots for the same duty cycle; hence the probability of successfully obtaining a non-conflicting receiving wakeup slot is lower.

Ultimately, the packet delivery ratio and end-to-end delay results (see Figs. 9c and 9d) show that using a smaller duty cycle duration provides better performance. The poor performance obtained when large T is used is due to the effect of slow convergence time. Especially in conditions where duty cycles are highly dynamic, slow convergence may result in the forwarding of packets through sub-optimal paths.

Maximum retry limit L. The maximum retry limit L determines the level of reliability provided by the forwarding scheme. The packet delivery ratio results (see Fig. 10a) demonstrate the positive effect of allowing higher number of retransmissions. The delivery ratio improves significantly when L is increased from 0 to 1; thereafter, the improvement is marginal. The downside of allowing higher retransmissions is increased end-to-end delay (see Fig. 10b). The increase is more dramatic in the cloudy scenario; this is due to the fact that every packet retransmission incurs a higher latency because of the lower node duty cycles in the cloudy scenario, in which the receive slots are generally spaced further apart.

Receive slot discount factor α. The receive slot discount factor α in the proposed forwarding scheme is used to estimate the number of receiving wakeup slots of a neighbor node u when node v fails to receive an update from u. A value of $\alpha = 0$ implies that whenever node v fails to receive an update from node u, the number of receive slots of u is set to 0 immediately while $\alpha = 1$ means that the number of receive slots of u is retained without any change. We can see that these values are extreme and as shown by the results (see Fig. 11), using either 0 or 1 does not provide the best performance. The optimal performance can be obtained when α is around 0.8–0.9 for both the sunny and cloudy scenarios. Using a value less than the optimal value underestimates the number of receive slots of u which unnecessarily increases the sleep latency. Likewise, using a value greater than the optimal value overestimates the number of receive slots

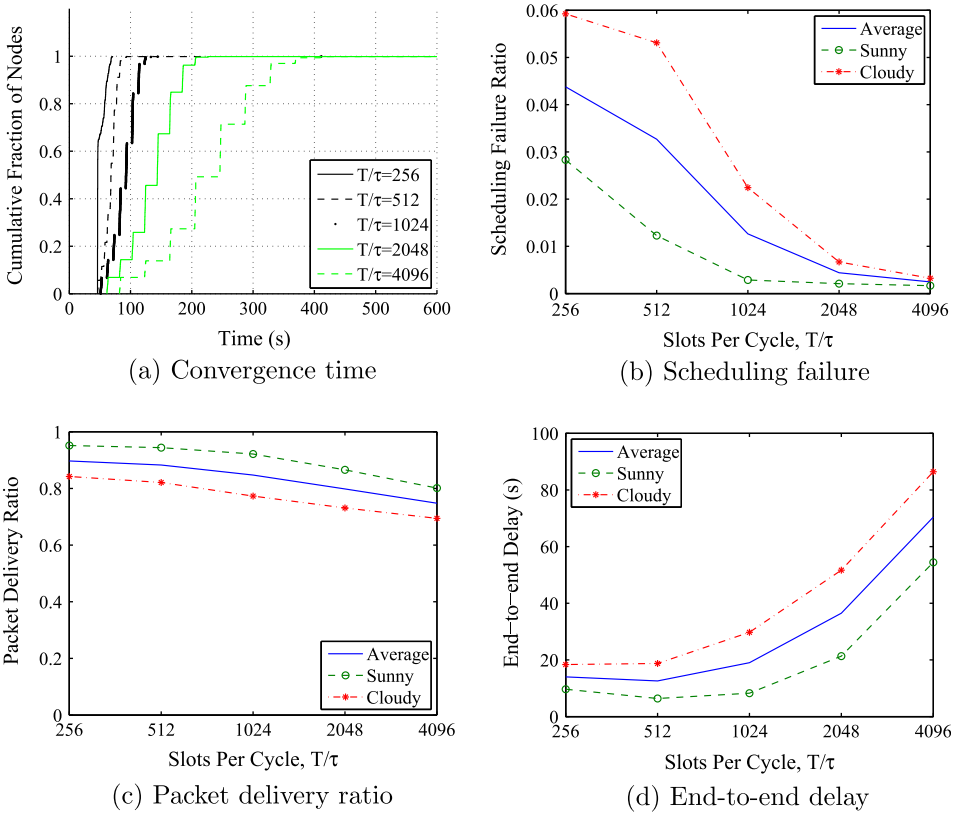
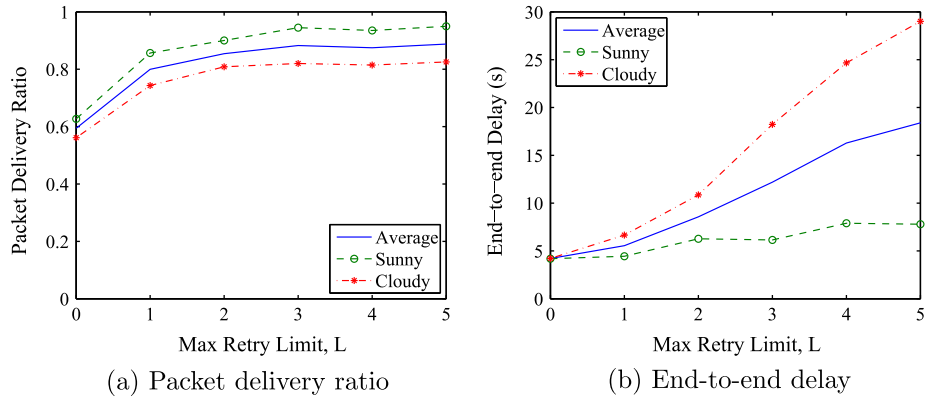
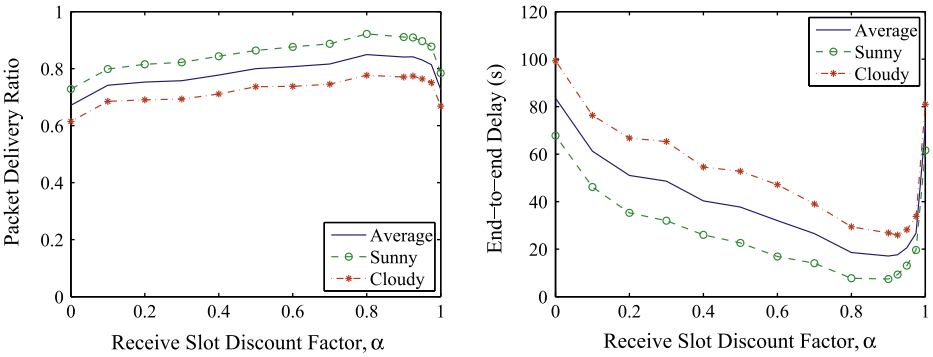


Fig. 9. Effect of duty cycle duration.



(a) Packet delivery ratio (b) End-to-end delay

Fig. 10. Effect of maximum retry limit.



(a) Packet delivery ratio (b) End-to-end delay

Fig. 11. Effect of slot discount factor.

of u which increases the scheduling errors thereby increasing the latency due to retransmission.

7.1. Scheduling performance comparison

We now compare the performance of the proposed wakeup scheduling scheme (bit-reversal permutation sequence based scheduling or BRPS) with the energy-synchronized communication (ESC) scheme [7]. ESC is one of the first schemes proposed for EPWSN and represents the state-of-the-art in dynamic wakeup scheduling. We implemented a simulation model of ESC as described in [7] with the following notable features: (i) To reduce storage and communication overhead, we represented the wakeup schedules as a bitmap instead of an integer array. Hence, a cycle with 512 slots only requires 512/8 = 64 bytes regardless of the number of active slots. Using integers, a cycle with n active slots requires $2n$ bytes. In the bitmap representation, a bit 1 at position i implies that slot i is active whereas a bit 0 implies that slot i is inactive. (ii) To further reduce communication overhead, schedule updates are piggybacked in neighbor updates. (iii) In ESC, every node requires the packet ready times at its predecessor nodes. As this is difficult to obtain *a priori*, we used the receiving wakeup slots of these predecessor nodes for this purpose. This is reasonable because packets are most likely to become ready for transmission after a node wakes up in its receiving wakeup slot. (iv) We implemented both the adjustment-based approach and the shuffle-based approach. Note that the former satisfies the properties of robustness while the latter does not. Table 4 provides a brief comparison of BRPS and the two variants of ESC. The variable n denotes the number of active slots while m denotes the sum of the active slots of a node's successors and the number of packets sent by its predecessors. The scheduling computation complexity of BRPS is $O(1)$ because it only requires knowledge of the number of active slots. Whereas, ESC requires $O(m)$ as it computes the optimal wakeup slot positions by considering all the individual active slots of a node's successors and the packet ready times at its predecessors.

Table 4
Comparison between BRPS and ESC (adjust and shuffle).

Criteria	BRPS	ESC-adjust	ESC-shuffle
Storage overhead per neighbor	$O(1)$	$O(n)$	$O(n)$
Communication overhead	$O(1)$	$O(n)$	$O(n)$
Schedule computation complexity	$O(1)$	$O(m)$	$O(m)$
Robustness	Yes	Yes	No

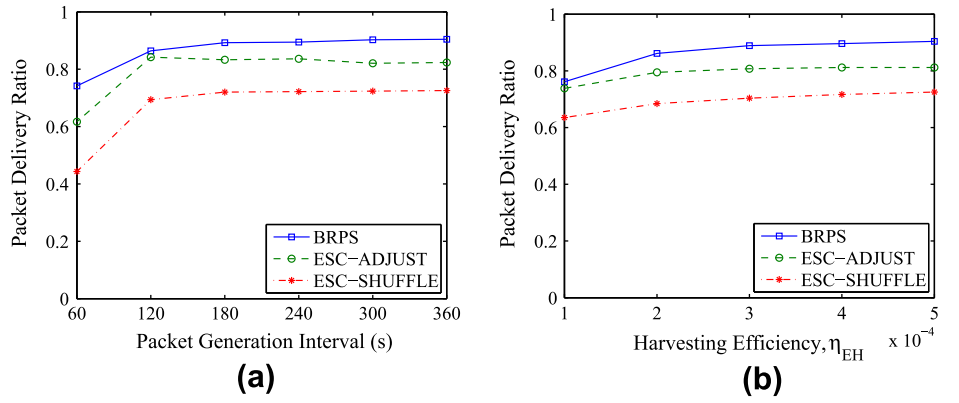


Fig. 12. Packet delivery ratio of different scheduling schemes.

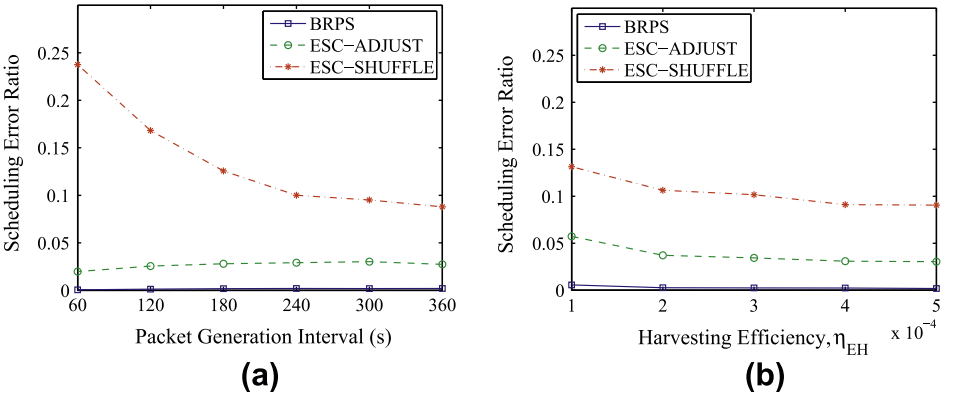


Fig. 13. Scheduling error ratio of different scheduling schemes.

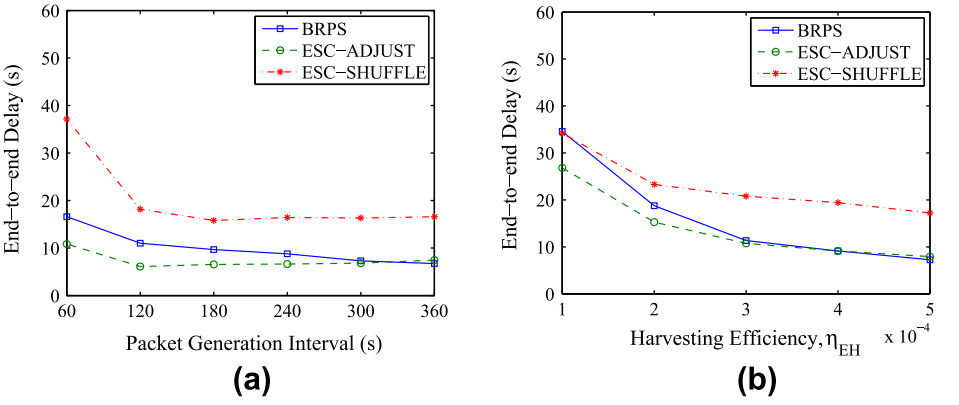


Fig. 14. End-to-end delay of different scheduling schemes.

In the comparison, we paired the above scheduling schemes with ETD. Note that using other routing metrics (ETX or hop count) yielded inferior performance. (Section 7.2 provides a comparison of the different routing metrics.) We fix the maximum retry limit L to 3 and the receive slot discount factor α to 0.8. The results plotted in Figs. 12 and 13 are the averages of the sunny and cloudy scenarios.

In terms of packet delivery ratio (see Fig. 12), we can see that BRPS outperforms ESC-SHUFFLE and ESC-ADJUST by 20% and 10%, respectively on the average. We did not expect to see significant difference in the data delivery performance because the schemes used the same routing metric. The performance advantage of BRPS is due to its robustness and the frequency of schedule updates. Note

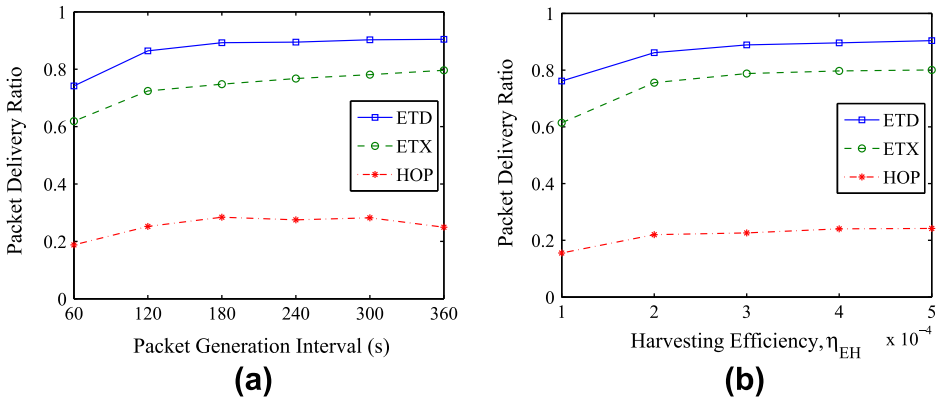


Fig. 15. Packet delivery ratio of different path metrics (in tandem with BRPS).

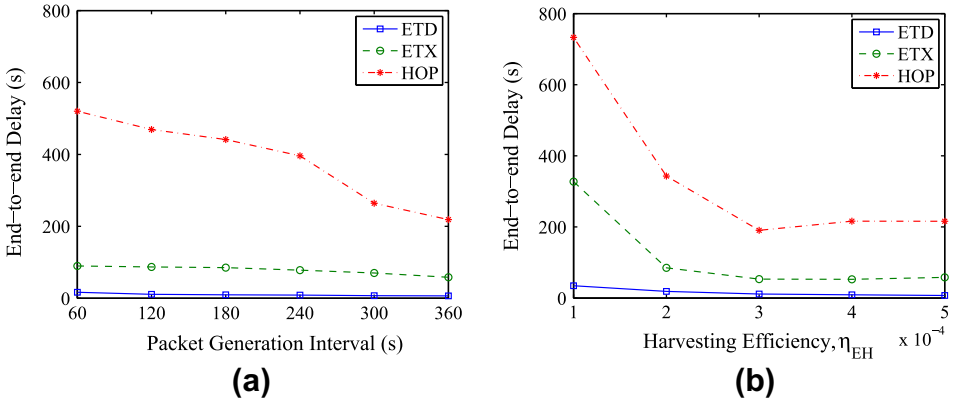


Fig. 16. End-to-end delay of different path metrics (in tandem with BRPS).

that the schedule information of BRPS (i.e., the number of receiving wakeup slots) is always included in every neighbor update. Because of the large overhead entailed by exchanging entire schedules in ESC, schedule updates are only transmitted when schedule changes have occurred. The net effect is that both ESC variants have higher scheduling error, as shown in Fig. 13. Between ESC-ADJUST and ESC-SHUFFLE, we can see the significant benefit of robustness as the former has significantly better performance.

In terms of delay (see Fig. 14), we can see that the performance of BRPS is well within the performance of the two ESC variants. ESC-ADJUST shows the best performance while ESC-SHUFFLE shows the worst performance. The higher delay of the latter can be attributed to its non-robustness which causes higher scheduling error ratio. A high scheduling error ratio implies more packet retransmissions which naturally leads to higher delay. Comparing BRPS and ESC-ADJUST, we can see that at moderate harvesting efficiency values and packet generation rates, BRPS can closely match the performance of ESC-ADJUST. Given the high complexity incurred by ESC in computing the optimal wakeup slot positions (it needs to consider the packet ready times of predecessor nodes and wakeup slots of successor nodes), the comparable performance of BRPS is a strong demonstration of the validity of the theory behind its design.

7.2. Routing metric performance comparison

To complete our simulation studies, we compare the performance of the proposed path metric (ETD) with ETX [23] and hop count in tandem with BRPS. Note that the relative performance of the three metrics remain the same when they are respectively paired with ESC-SHUFFLE and ESC-ADJUST. Likewise, the relative performance of the scheduling schemes remain the same in each of the three metrics.

As in the scheduling performance comparison, we also fix the maximum retry limit L to 3 and the receive slot discount factor α to 0.8. The results of the sunny and cloudy scenarios are averaged to obtain the packet delivery ratio and end-to-end delay results, which are shown in Figs. 15 and 16, respectively. We can see that in both performance metrics, ETD shows the best performance. Because of its awareness of both sleep latency and link quality, ETD considerably outperforms ETX and hop count by 10% and 60%, respectively, in most of the traffic conditions and harvesting efficiency values.

The end-to-end delay results (see Fig. 16) also show the significant advantage of ETD over ETX and hop count. Note that ETD's delay is less than 1/10 and 1/30 that of ETX and hop count, respectively. These results demonstrate the significant impact of both sleep latency and link quality on end-to-end packet forwarding. Note that ETX, a metric that considers link quality, performed much better than hop count but still significantly behind ETD. These results strongly suggest that blindness to sleep latency can cause considerable delay degradation.

8. Conclusions and future work

In environmentally-powered wireless sensor networks, low latency wakeup scheduling and packet forwarding is challenging due to dynamic duty cycling which necessitates the use of dynamic wakeup schedules and poses time-varying sleep latencies.

We have shown analytically that the expected sleep latency is affected by the variance of the intervals between receiving wakeup slots: when the variance of the intervals is low (high), the expected latency is low (high). This is because when the intervals between receiving wakeup slots are highly uneven, it is more likely for a packet to become ready for transmission at a larger interval than a shorter interval. We therefore introduced a scheduling scheme that aims to position receiving wakeup slots as evenly as possible. To reduce storage and communication overhead, the schedule of a node is represented compactly using an integer sequence.

We analytically obtained the worst-case sleep latency of a scheduling scheme that uses the bit-reversal permutation sequence (BRPS) and found it to be slightly worse than the ideal scheme (scheduling scheme where the receiving wakeup slots are equally-spaced) but better than schemes where the receiving wakeup slots are spaced uniformly or exponentially. But while the ideal scheme is not robust to changes in the duty cycle, the BRPS is robust. We performed simulations to compare the performance of BRPS with ESC, a scheduling scheme that represents the state-of-the art. Results show that BRPS provides low latency and can closely match the performance of ESC. Furthermore, BRPS has a lower scheduling error ratio which translates to better packet delivery ratio. Aside from having a lower storage and communication overhead, BRPS also has a lower computational complexity compared with ESC.

A low sleep latency schedule does not necessarily lead to low end-to-end latency paths because other factors such as link quality play a significant role in the performance of packet forwarding. We therefore formulated a metric called expected transmission delay (ETD) which simultaneously considers sleep latency (due to duty cycling), and wireless link quality. We showed that the metric is left-monotonic and left-isotonic, proving that its use in distributed algorithms such as the distributed Bellman–Ford will yield consistent, loop-free and optimal paths. Compared with ETX and hop count and used in tandem with BRPS, ETD provides the best performance in terms of packet delivery ratio and delay.

In our future work, we intend to study the performance of the scheduling and forwarding scheme using different adaptive duty cycle control algorithms and other energy harvesting source models.

References

[1] X. Jiang, J. Polastre, D. Culler, Perpetual environmentally powered sensor networks, in: Proc. IEEE IPSN, 2005.
[2] J. Paradiso, T. Starner, Energy scavenging for mobile and wireless electronics, IEEE Pervasive Computing 4 (1) (2005) 18–27.
[3] W. Seah, Z. Eu, H.-P. Tan, Wireless sensor networks powered by ambient energy harvesting (wsn-heap) – survey and challenges, in: Proc. Wireless VITAE, 2009.
[4] S. Sudevalayam, P. Kulkarni, Energy harvesting sensor nodes: survey and implications, IEEE Communications Surveys Tutorials (99) (2010) 1–19.
[5] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, V. Raghunathan, Adaptive duty cycling for energy harvesting systems, in: Proc. ACM ISLPED, 2006.
[6] A. Kansal, J. Hsu, S. Zahedi, M.B. Srivastava, Power management in energy harvesting sensor networks, ACM Transactions on Embedded Computing Systems 6 (2007) 1–38.
[7] Y. Gu, T. Zhu, T. He, Esc: Energy synchronized communication in sustainable sensor networks, in: Proc. IEEE ICNP, 2009.
[8] G. Anastasi, M. Conti, M.D. Francesco, A. Passarella, Energy conservation in wireless sensor networks: a survey, Ad Hoc Networks 9 (3) (2009) 537–568.
[9] C. Vigorito, D. Ganesan, A. Barto, Adaptive control of duty cycling in energy-harvesting wireless sensor networks, in: Proc. IEEE SECON, 2007.
[10] T. Zhu, Z. Zhong, Y. Gu, T. He, Z.-L. Zhang, Leakage-aware energy synchronization for wireless sensor networks, in: Proc. ACM MobiSys, 2009.
[11] C.-F. Hsin, M. Liu, Randomly duty-cycled wireless sensor networks: dynamics of coverage, IEEE Transactions on Wireless Communications 5 (2006) 3182–3192.
[12] K.-W. Lee, V. Pappas, A. Tantawi, Enabling accurate node control in randomized duty cycling networks, in: Proc. IEEE ICDCS, 2008, pp. 123–132.
[13] G. Ghidini, S. Das, An energy-efficient markov chain-based randomized duty cycling scheme for wireless sensor networks, in: Proc. IEEE ICDCS, 2011, pp. 67–76.
[14] G. Lu, N. Sadagopan, B. Krishnamachari, A. Goel, Delay efficient sleep scheduling in wireless sensor networks, in: Proc. IEEE INFOCOM, 2005.
[15] S. Guha, C.-K. Chau, P. Basu, Green wave: Latency and capacity-efficient sleep scheduling for wireless networks, in: Proc. IEEE INFOCOM, 2010.
[16] W. Ye, J. Heidemann, D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, in: Proc. IEEE INFOCOM, 2002.
[17] Y. Gu, T. He, Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links, in: Proc. ACM SenSys, 2007.
[18] K. Akkaya, M. Younis, A survey on routing protocols for wireless sensor networks, Ad Hoc Networks 3 (3) (2005) 325–349.
[19] N. Sharma, J. Gummeson, D. Irwin, P. Shenoy, Cloudy computing: Leveraging weather forecasts in energy harvesting sensor systems, in: Proc. IEEE SECON, 2010, pp. 1–9.
[20] O. Jumira, R. Wolhuter, N. Mitton, Prediction model for solar energy harvesting wireless sensors, in: Proc. IEEE AFRICOMM, 2012.
[21] R. Zheng, J.C. Hou, L. Sha, Asynchronous wakeup for ad hoc networks, in: Proc. ACM MobiHoc, 2003.
[22] A. Keshavarzian, H. Lee, L. Venkatraman, Wakeup scheduling in wireless sensor networks, in: Proc. ACM MobiHoc, 2006.
[23] D. De Couto, D. Aguayo, J. Bicket, R. Morris, A high-throughput path metric for multi-hop wireless routing, in: Proc. ACM MobiCom, 2003.
[24] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, W. Hong, A macroscope in the redwoods, in: Proc. ACM SenSys, 2005, pp. 51–63.
[25] IEEE 802.14.5-2006 Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), 2006.

[26] M.J. McGlynn, S.A. Borbash, Birthday protocols for low energy deployment and flexible neighbor discovery in ad hoc wireless networks, in: Proc. ACM MobiHoc, 2001.
[27] S. Ganeriwal, R. Kumar, M.B. Srivastava, Timing-sync protocol for sensor networks, in: Proc. ACM SenSys, 2003.
[28] H. Dai, R. Han, Tsync: a lightweight bidirectional time synchronization service for wireless sensor networks, SIGMOBILE Mobile Computing and Communications Review 8 (2004) 125–139.
[29] M. Maróti, B. Kusy, G. Simon, A. Lédeczi, The flooding time synchronization protocol, in: Proc. ACM SenSys, 2004.
[30] S. Ross, Introduction to Probability Models, ninth ed., Elsevier, 2007.
[31] E. Osuna, G. Newell, Control strategies for an idealized public transportation system, Transportation Science 6 (1) (1972) 52–72.
[32] S. Ross, Stochastic Processes, Wiley and Sons, 1982.
[33] A. Karp, Bit reversal on uniprocessors, SIAM Review 38 (1) (1996) 1–26.
[34] Y. Yang, J. Wang, Design guidelines for routing metrics in multihop wireless networks, in: Proc. IEEE INFOCOM, 2008, pp. 1615–1623.
[35] Q. Cao, T. He, L. Fang, T. Abdelzaher, J. Stankovic, S. Son, Efficiency centric communication model for wireless sensor networks, in: Proc. IEEE INFOCOM, 2006.
[36] V. Gungor, C. Sastry, Z. Song, R. Integlia, Resource-aware and link quality based routing metric for wireless sensor and actor networks, in: Proc. IEEE ICC, 2007.
[37] D.P. Bertsekas, Gallager, second ed., Data Networks, Prentice Hall, 1992.
[38] The Qualnet Simulator. <http://www.scalable-networks.com/products/developer.php> (accessed 31.05.12).
[39] S. Kim, K.-S. No, P. Chou, Design and performance analysis of supercapacitor charging circuits for wireless sensor nodes, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 1 (2) (2011).
[40] C.-Y. Chen, P. Chou, Duracap: a supercapacitor-based, power-bootstrapping, maximum power point tracking energy-harvesting system, in: Proc. ACM ISLPED, 2010.
[41] National Renewable Energy Laboratory. <http://www.nrel.gov> (accessed 31.05.12).
[42] Waspmote Datasheet. <http://www.libelium.com//waspmote/waspmote-datasheeteng.pdf> (accessed 31.05.12).
[43] XBee Datasheet. <http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf> (accessed 31.05.12).

Alvin Valera obtained the Bachelor of Science degree (computer engineering) from the University of the Philippines on 1998 and Master of Science (computer science) from the National University of Singapore in 2003. He is currently with the Institute for Infocomm Research (I2R), Singapore as a Senior Research Engineer. He is also a Ph.D. candidate at the National University of Singapore. His current research interests include optimization and control of wireless networks in general, and on scheduling, routing, and medium access control of energy-harvesting wireless sensor networks in particular.

Wee-Seng Soh received the B.Eng. (Hons) and M.Eng. degrees in electrical engineering from the National University of Singapore (NUS) in 1996 and 1998, respectively. In 1998, he was awarded the Overseas Graduate Scholarship by the National University of Singapore to study at Carnegie Mellon University, Pittsburgh, PA, where he received the Ph.D. degree in electrical and computer engineering in 2003. Since 2004, he has been with the Department of Electrical and Computer Engineering, National University of Singapore, where he is currently an Assistant Professor. Prior to joining NUS, he was a Postdoctoral Research Fellow in the Electrical Engineering and Computer Science Department, University of Michigan. He has served on the Technical Program Committees (TPC) of 20 conferences, and has also served as a TPC co-chair for ICCS'08 and WUnderNet'11. He is currently serving as an area editor of Computer Communications (Elsevier) journal. His current research interests are in wireless networks, underwater networks, and indoor tracking/localization techniques.

Hwee-Pink Tan is currently a Senior Scientist in the Institute for Infocomm Research, and is also the A☆STAR SERC Programme Manager for the "Sense and Sense-abilities" Programme that focuses on developing integrative technologies for data collection, transport and intelligence in large scale and heterogeneous sensor networks, and validating them via test-bedding based on sensing needs of targeted government agencies. His research has focused on the design, modeling and performance evaluation of networking protocols for wireless networks, and his past and current research interests include underwater acoustic sensor networks, wireless sensor networks powered by ambient energy harvesting and large scale and heterogeneous wireless sensor networks. He has been a PI for several research collaboration projects on wireless sensor networks, and has published more than 55 papers in peer-reviewed journals and conferences in the area of wireless networks.