Computer Communications 52 (2014) 21–36

Contents lists available at ScienceDirect

Computer Communications

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

Survey on wakeup scheduling for 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 117583, Singapore

ARTICLE INFO

Article history:
Available online 17 May 2014

Keywords:
Energy-harvesting
Wireless sensor network
Duty cycling
Wakeup scheduling

ABSTRACT

Advances in energy harvesting technologies and ultra low-power computing and communication devices are enabling the realization of environmentally-powered wireless sensor networks (EPWSNs). Because of limited and dynamic energy supply, EPWSNs are duty-cycled to achieve *energy-neutrality*, a condition where the energy demand does not exceed the energy supply. Duty cycling entails nodes to sleep and wakeup according to a *wakeup scheduling* scheme. In this paper, we survey the various wakeup scheduling schemes, with focus on their suitability for EPWSNs. A classification scheme is proposed to characterize existing wakeup scheduling schemes, with three main categories, namely, *asynchronous*, *synchronous*, and *hybrid*. Each wakeup scheduling scheme is presented and discussed under the appropriate category. The paper concludes with open research issues.

1. Introduction

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

While energy harvesting can theoretically enable perpetual network operation, it poses a major constraint on energy availability: the amount of energy available for consumption at any given instant can be 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 and adaptively utilize available energy to optimize the network throughput or end-to-end delay. The new guiding principle in EPWSN is *energy neutral operation*, which means operating nodes in a sustainable manner wherein 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 indicate the time intervals at which a node activates its radio transceivers to perform either packet transmission or reception. In this paper, we present a survey of the state-of-the-art in wakeup scheduling. Our ultimate aim is to characterize and differentiate the various schemes and determine their suitability for EPWSNs.

The rest of the paper is organized as follows: Section 2 motivates the survey with a presentation of the unique characteristics and challenges of EPWSNs. This section will also introduce the important factors that must be considered in designing wakeup scheduling schemes for EPWSNs. Section 3 presents the fundamental characteristics and properties of wakeup scheduling schemes and most importantly, the classification system that will be used to describe the various schemes. Sections 4–6 contain detailed descriptions and discussions of asynchronous, synchronous, and hybrid scheduling schemes, respectively. Section 7 concludes the paper with a qualitative assessment of their suitability for EPWSNs and open research issues.

* 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).

http://dx.doi.org/10.1016/j.comcom.2014.05.004
0140-3664/

2. Energy harvesting in sensor networks

To set the scene for this survey paper, we present an overview of energy harvesting in wireless sensor networks. We describe the various components needed to assemble an environmentally-powered wireless sensor node (or EPWSN node for short), followed by a discussion of the challenges faced by EPWSNs and the notion of energy-neutrality. We end the section with an enumeration of the important factors that must be considered in the design of wakeup scheduling schemes for EPWSNs.

2.1. Energy harvesting

Energy harvesting, also referred to as "*energy scavenging*" in the literature, is the process of converting ambient energy from the environment into electrical energy to power devices such as sensor nodes and mobile electronics [2]. Fig. 1 shows the various components of an EPWSN node: (i) *energy harvester* for converting ambient energy to electrical energy; (ii) *energy storage* for storing harvested energy; and (iii) *sensor load* which essentially consists of the sensor node electronics (mainboard, microcontroller, radio, sensors and other peripherals). Because ambient energy is readily available, energy harvesting could enable perpetual operation without the need for battery replacement [3,4].

There are numerous sources of ambient energy and they can be grouped into several classes according to their underlying physical process [2]:

- **Mechanical:** from sources such as wind, vibration, mechanical stress and strain and human body movement.
- **Light:** from sunlight or room (artificial) light.
- **Thermal:** waste energy from engines, furnaces, heaters and friction sources.
- **Electromagnetic:** from inductors, coils, transformers and radio frequency sources.
- **Others:** from chemical and biological sources.

The conversion of ambient energy to electrical energy requires the use of an energy harvester or transducer. Table 1 provides a summary of achievable energy harvesting rates of several state-of-the-art energy harvesting technologies [2,4,11,12]. Solar energy, which is one of the most abundant and readily available energy, can be harvested using photo-voltaic (PV) cells which can have 25% efficiency [11]. When such a PV cell is directly exposed to sunlight which has an irradiance of 1000 W/m^2 (this is a typical value of direct solar irradiance [13]), it can potentially generate 250 W/m^2 or 25 mW/cm^2.

2.2. Energy storage technologies

Energy storage or buffer is an important component of an EPWSN node. It serves two important functions [6]: (i) to act as storage for unused or excess harvested energy; and (ii) to act as additional energy supply when load consumption is not met by harvested energy. It is possible to power a sensor node directly from an energy harvester without any energy buffer but its operation will be severely constrained. In particular, such a node can only operate when the amount of harvested power is greater than or equal the required node consumption. When the amount of harvested power is not sufficient, the node will not operate and the harvested power will be wasted. In cases where the amount of harvested power exceeds the node consumption, the excess will likewise be wasted.

Currently, there are two dominant energy storage technologies that can be utilized in EPWSN [1,6,14,10,4,11]: (i) secondary or rechargeable batteries; and (ii) supercapacitors, also known as ultracapacitors or electrochemical double layer capacitors. Although there are many types of rechargeable batteries available in the market, nickel metal hydride (NiMH) and lithium ion (Li-ion) are considered to be more suitable for sensor nodes [14,4].

As far as EPWSN is concerned, the most important characteristics of an energy storage technology are energy storage capacity, number of full recharge cycles, and self-discharge rate or leakage. Table 2 provides a comparison of several energy storage devices in terms of the three characteristics [14]. In general, rechargeable batteries provide high energy capacity while supercapacitors can provide low to moderate energy capacity. In terms of self-discharge rate, Li-ion batteries are slightly better than supercapacitors. One major advantage of supercapacitors is the number of full recharge cycles which is three orders of magnitude higher than that of rechargeable batteries. This has significant impact on the lifetime of the storage device, enabling supercapacitors to last for 10–20 years compared to a maximum of 5 and 3 years for Li-ion and NiMH, respectively [14].

2.3. Challenges

As enumerated by Akyildiz et al. [15], WSNs pose numerous challenges including highly dynamic network topology due to failure-prone nodes and wireless links, limited memory and processing power and most importantly, limited network lifetime due to battery capacity limitations. Energy harvesting has the potential to eliminate the problem of limited network lifetime but it poses a major constraint on the amount and consistency of energy that can be supplied to the sensor node. Unlike a battery-powered WSN node where the energy supply is guaranteed (while its battery is not exhausted), the **energy supply of an EPWSN node can be unpredictable and varies over time** [5–7].

Unsuitability of energy conservation as a design objective. In battery-powered WSN, network protocols are designed to conserve as much energy as possible, knowing that the energy supply is finite and will eventually be depleted. Network lifetime can be maximized by minimizing the energy consumption of individual nodes while at the same time balancing the energy consumption across nodes [8]. In EPWSNs where the energy supply can be replenished, the notion of network lifetime is inappropriate and this renders energy conservation as an unsuitable design objective.

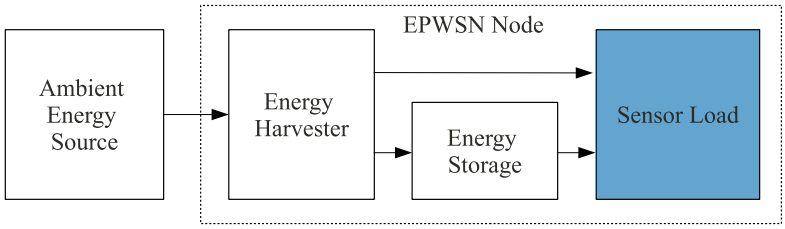


Fig. 1. Components of an EPWSN node.

Table 1
Efficiency of state-of-the-art energy harvesting technologies.

Energy source	Harvesting device/transducer	Efficiency	Achievable harvesting rate
Solar	Photo-voltaic cells	25%	25 mW/cm^2
Indoor light	Photo-voltaic cells	25%	25 μW/cm^2
Thermal	Thermoelectric generator	–	60 μW/cm^2
Wind	Anemometer	–	1200 mW h/day
Electromagnetic	RF antenna	–	10^{-5}–0.1 mW/cm^2
Indoor vibrations	EM induction	–	0.2 mW/cm^2
Vibrations (walking)	EM generator	–	0.95 mW
Vibrations (running)	EM generator	–	2.46 mW
Heel strike	Piezoelectric	7.5%	5 W

Table 2
Comparison of energy storage devices (adapted from [14]).

Device	Type	Capacity (mAh)	Recharge cycles	Self-discharge rate (%/month)
Maxwell BCAP350	350F supercapacitor	243	500,000	<30
Maxwell PC10	10F supercapacitor	6.9	500,000	<30
Panasonic HHR210AA/B	NiMH	2000	300	<30
Panasonic CGR17500	Li-ion	830	500	<10

Energy-neutrality. The new guiding principle in the design of EPWSN protocols is *energy neutral operation* which consists of two simultaneous goals: (i) optimizing the network performance while; and (ii) ensuring that energy supply and energy demand are balanced [5,6,9,7,10]. Several authors proposed essentially the same idea using different terms, namely, "energy-neutrality" [6], "energy-synchronized" [7], and "energetic sustainability" [16].

Dynamic duty cycling. 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 current energy level, energy buffer capacity as well as current and future (predicted) harvesting rates. 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 [17–21].

2.4. Considerations for wakeup scheduling

Wakeup scheduling schemes employed in EPWSNs must consider their unique characteristics and the underlying challenges posed by environmentally harvested energy supply. In the discussion of the various schemes, we examine how they measure up against the following important considerations:

Adaptation to environment dynamics. EPWSNs are highly dynamic in terms of topology, energy supply, and data traffic among others [15]. Schemes that employ *dynamic wakeup schedules* to respond to some or all of these dynamics are expected to perform better than those schemes that are oblivious. However, some non-adaptive schemes can be easily modified to respond to changes in the operating environment. Others, however, may not be amenable for use in dynamic environments.

Latency-aware. 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 [22,18,20]. Sleep latency is a major challenge in both battery-powered and environmentally-powered WSNs and significantly contributes to the end-to-end delay [20,7]. Schemes that explicitly tackle latency perform better than latency-oblivious schemes, but the awareness again comes at an additional cost.

Duty cycle range. Certain WSNs operate in very low duty cycles and as such, scheduling schemes that are designed with high duty cycle in mind may not work well in these regimes. For instance, schemes that rely on random schedules may have poor performance because the probability of the sender and receiver being awake at the same time is low. On the other hand, scheduling schemes tailored for low duty cycles may have poor performance in high duty cycle regimes.

Processing complexity. Because EPWSN nodes have limited processing capability [15], wakeup scheduling schemes that use complex algorithms may not be suitable. Their use may require the deployment of special nodes with sufficient processing capability to perform the complex computations and this entails some form of centralized processing.

Overhead and scalability. Finally, constraints on channel and storage capacities [15] imply that wakeup scheduling schemes must have low communication and storage overhead. Sources of overhead are mainly schedule exchange and storage. Notice that high overhead may imply that a scheme is not scalable in terms of the number of nodes.

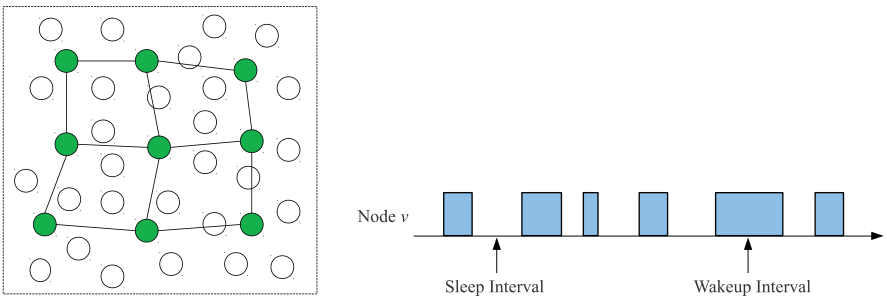
3. Fundamentals of wakeup scheduling

In this section, we study the various wakeup scheduling schemes in terms of the formation process, *i.e.*, how the wakeup schedules are obtained or computed, and the structure of the resulting wakeup schedule. But before delving into the details, let us first establish the relationship between *duty cycling* and *wakeup scheduling*.

In their paper, Anastasi et al. [8] split duty cycling into two approaches: *power management* and *topology control*. In the latter, the goal is to select a subset of nodes to be awake from a given set of nodes while in the former, the object is to select a subset of time intervals to be awake from a continuous time duration. Anastasi et al. further broke down power management into two categories, namely, sleep/wakeup scheduling and MAC-level duty cycling. In this paper, we refer to both approaches as wakeup scheduling since the objective is essentially to control the sleep and wakeup intervals of every node in the network. To further clarify, Fig. 2 illustrates the main difference between the two approaches.

3.1. Basic terminology

We put forth the following basic terminologies to avoid confusion in the discussions.



(a) Topology control (b) Wakeup scheduling

Fig. 2. Illustrating the main difference between topology control and wakeup scheduling. In topology control, the objective is to select a minimal subset of nodes that yields a connected network and covers the entire sensing area. In wakeup scheduling, the objective is to select a subset of time intervals to be awake (for every node v in the network) that minimizes the duty cycle while ensuring that throughput or latency (or both) requirement is achieved.

Definition 1 (*Wakeup Interval*). The time duration at which the radio is **switched on** to enable the node to either receive or transmit packets. The literature sometimes refer to this as **active** or **on interval**.

Definition 2 (*Sleep Interval*). The time duration at which the radio is **switched off** to enable the node to conserve energy. The literature sometimes refer to this as **inactive**, **off** or **dormant interval**.

Wakeup scheduling schemes can be broadly grouped into two types: those that divide time into equal-length intervals called *slots* and those that treat time continuously. The above definitions are usually applied to unslotted schemes. For *slotted* schemes, wakeup and sleep intervals are defined in terms of multiples of slots.

Definition 3 (*Wakeup schedule*). A sequence of wakeup and sleep intervals that is usually specified for one cycle and repeats every cycle until otherwise modified by the wakeup scheduling scheme. This is sometimes referred to as **sleep schedule** or **sleep/wakeup schedule** in the literature.

3.2. Data exchange

The ultimate aim of wakeup scheduling schemes is to enable nodes to exchange data during wakeup intervals. Most wakeup scheduling schemes use a simple design wherein every wakeup interval can accommodate at most one data frame or packet. Nevertheless, there are also several proposed schemes that can handle multiple data frames or packets in every wakeup interval. In schemes that can accommodate at most one packet per wakeup interval, wakeup schedules are considered to be *receiver-centric*, *i.e.*, the specified wakeup intervals in the schedule are meant for packet reception only.[1] In the latter, wakeup intervals are considered *bi-directional*, *i.e.*, wakeup intervals can be used for both packet transmission and reception.

3.3. Classification of wakeup scheduling schemes

To aid us in the study of the various wakeup scheduling schemes, we introduce a morphological classification system as shown in Fig. 3. As will become evident in the discussions, this taxonomy provides a good perspective on the effort required by a scheduling scheme to obtain a wakeup schedule. Fig. 3 also includes two or more representative wakeup scheduling schemes for each class that we have included in this survey.

At the most fundamental level, we can classify scheduling schemes based on their requirement for synchronization, *i.e.*, *synchronous* or *asynchronous*. As the name implies, synchronous scheduling schemes require that the time across nodes by synchronized. On the other hand, asynchronous scheduling schemes do not require any form of synchronization. Some schemes may operate using both approaches and we refer to them as *hybrid* schemes. Sections 4 and 5 present the various asynchronous and synchronous scheduling schemes, respectively, while Section 6 discusses the hybrid schemes.

4. Asynchronous schemes

Asynchronous schemes were the earliest protocols proposed for wakeup scheduling. Their main distinguishing feature is that they operate in an asynchronous manner, meaning that nodes wakeup to transmit without regard on whether other nodes are awake to receive. Because of this, asynchronous schemes do not require time synchronization. This is one of its major advantages because as Wu et al. [23] found in their study, efforts to periodically re-synchronize time across nodes can entail significant energy consumption. Another major advantage of asynchronous approaches is that they do not require any computation as well as communication and storage overhead since no schedules are exchanged and stored.

One of the main challenges of asynchronous scheduling is how to exchange data between two nodes which are not aware of each other's wakeup schedules. There are three possible approaches to do this: (i) *transmitter-initiated*; (ii) *receiver-initiated*; and (iii) *combinatorial* or *random*.

4.1. Transmitter-initiated

In transmitter-initiated protocols, a transmitting node v transmits a special frame to indicate to its neighbor nodes that it has data to transmit. When a neighbor node hears the special frame in one of its wakeup intervals, it awaits for the transmission of the data frame. A wakeup interval is receiver-centric and can accommodate at most one data frame. (It is possible for a wakeup interval to accommodate more than one data frame depending on the data frame duration or the wakeup interval duration. However, the use of preamble acts as a reservation mechanism whereby only one node has the right to transmit one or more data frames within the wakeup interval.)

B-MAC. The first protocol to use this approach is the B-MAC [24] protocol. In this protocol, nodes periodically wakeup for a duration of T_w and sleep for a duration of T_s. T_w is specified to be long

[1] In receiver-centric wakeup scheduling schemes, a node u with data to transmit to v must wakeup at an interval where v is awake as specified by the latter's wakeup schedule. Note that this interval at which u wakes up to transmit its packet is not considered part of its wakeup schedule.

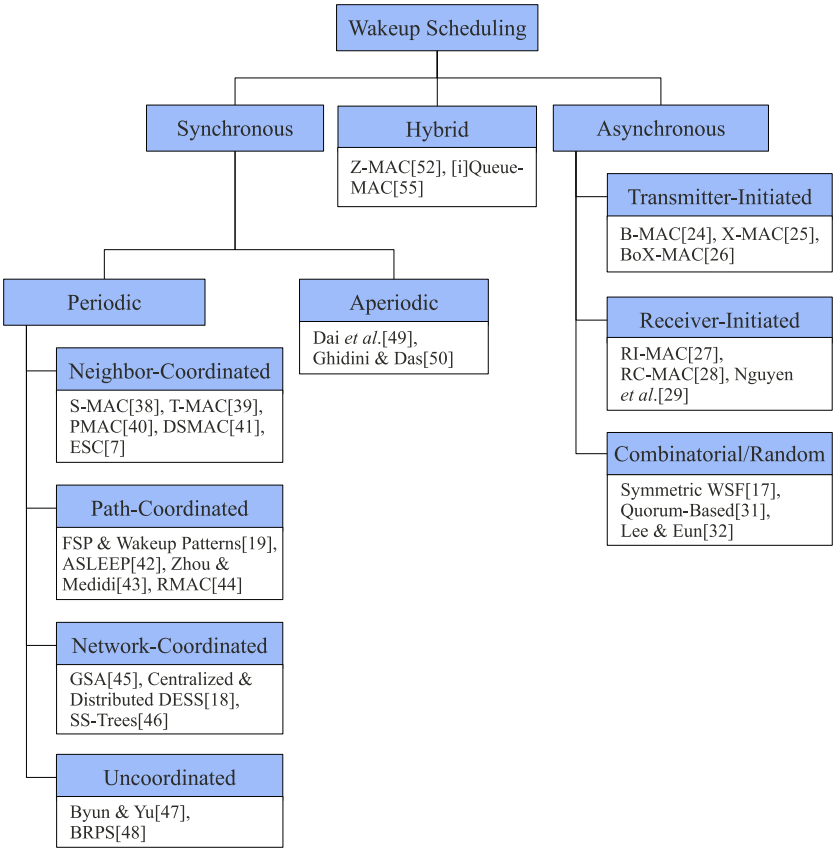


Fig. 3. Taxonomy of wakeup scheduling schemes.

enough for a node to detect the presence of a special signal known as *preamble*. When a node v has data to send, it immediately wakes up and transmits a preamble frame for duration of T_s followed by the data frame. A node w that wakes up and detects the preamble will then remain awake for the remaining preamble duration until it receives the data frame. This process is shown in Fig. 4.

B-MAC enhancements. B-MAC suffers from two major drawbacks. Firstly, sending nodes must transmit a long preamble which must be at least T_s and secondly, overhearing nodes (*i.e.*, not the intended receiver) will also have to be awake during the entire preamble transmission and possibly until data transmission is completed. Several enhancements have been proposed to address these deficiencies. X-MAC [25] tackles the long preamble problem by replacing it with a *strobe short preamble*. In addition, the strobe preamble includes the intended receiver address, thereby allowing overhearing nodes to go back to sleep the moment they receive a strobe. BoX-MAC [26] further improves on X-MAC by replacing the short preamble transmissions with data transmissions. This however assumes that data packets are short enough to be effective replacements of strobe preamble.

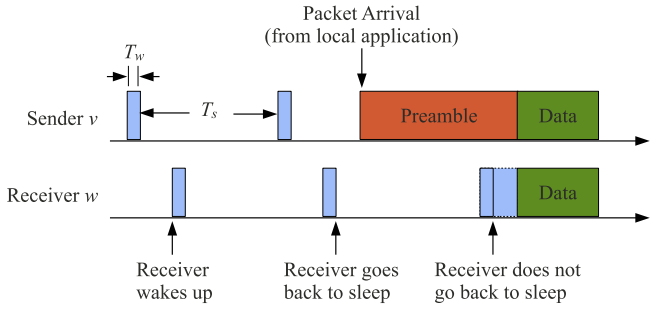


Fig. 4. B-MAC operation. Nodes independently sleep and wakeup periodically but with the same wakeup and sleep durations. When node v needs to send data (a packet arrives from local application/higher layer), it must first send a preamble which should be at least as long as sleep duration T_s (the *check interval*). When receiver node w detects the preamble, it remains awake to receive the preamble and data. If a node wakes up and does not detect a preamble within the wakeup duration T_w, it goes back to sleep.

Discussion. The main advantage of transmitter-initiated protocols is their low computation and storage complexity. Transmitter-initiated protocols also seem to be easily extensible to support dynamic operation. To support dynamic wakeup scheduling, the duration of the preamble needs to be as long as the maximum sleep interval (*i.e.*, the minimum duty cycle). Note however that these protocols will perform poorly in low-duty cycle regimes. In such environments, T_s will be significantly higher than T_w. This will waste a lot of energy on the part of transmitters as they need to transmit very long preambles. Another disadvantage of transmitter-initiated protocols is that since nodes independently sleep and wakeup, their sleep latency is non-deterministic.

4.2. Receiver-initiated

Receiver-initiated protocols essentially pass the burden of energy consumption for the overhead from transmitters to receivers. That is, a receiving node v transmits a special frame every time it wakes up to indicate to potential transmitters that it is ready to receive data frames. When a node u has pending data to transmit, it immediately wakes up and awaits for the transmission of the special frame from its neighbors. The moment it receives the special frame from another node v, u commences data transmission to v. Similar to the transmitter-initiated protocols, a wakeup interval

in receiver-initiated protocols is receiver-centric and can accommodate at most one data frame.

RI-MAC. Nodes periodically sleep for a duration of T_s and wakeup for a duration of T_w. Whenever a node wakes up, it transmits a *beacon* to indicate to potential transmitters that it is ready to receive data. If it does not receive a data frame after T_w, it goes back to sleep. From a transmitter perspective, if a node v has data to transmit, it waits for the beacon from the intended receiver before transmitting its data. Fig. 5 shows the operation of RI-MAC [27], the protocol that first proposed this approach.

RI-MAC enhancements. Huang et al. [28] proposed Receiver-Centric MAC (RC-MAC) that exploits the underlying routing tree structure to coordinate the transmission of a node's children. The coordination is done by piggybacking the ID of the next child that can transmit in the ACK. Meanwhile, Nguyen et al. [29] extended RI-MAC for EPWSNs by using energy harvesting rate and queue length to adjust the duty cycle of the nodes.

Discussion. Like transmitter-initiated protocols, receiver-initiated protocols also require low computational and storage overhead. These protocols are also easily adaptable to support dynamic wakeup scheduling. However, they may perform poorly in both low and high duty cycle situations. In the former, transmitters will need to be awake for long durations to be able to receive a beacon. In the latter, frequent beacon transmissions by receivers may cause excessive contention and power consumption. Another disadvantage of these protocols is that since nodes independently sleep and wakeup, their sleep latency is stochastic.

4.3. Combinatorial and random

The preceding approaches require the transmission of a special frame either by the receiver or transmitter. Combinatorial and random schemes obviate the transmission of such special frames which therefore translates to better energy efficiency. In a wakeup interval, nodes can exchange one or more data packets with each other, the number of which depends on the wakeup interval duration.

Symmetric WSF. Zheng et al. [17] proposed a totally different approach by formulating the schedule generation process as a *block design* problem in combinatorics. The main idea is that for any two nodes v and w, their respective wakeup schedules must overlap within a specified time interval to ensure that sleep latency is bounded. A *wakeup schedule function* (WSF) $f_v(x) = \sum_{i=0}^{T-1} a_i x^i$ is defined for every node v in the network, where T is the cycle duration in terms of slots and $a_i = 0$ if i is a sleep interval (slot) or $a_i = 1$, if i is a wakeup interval (slot), $\forall i \in [0, T-1]$. Now, we can perform a cyclic shift of the schedule of v by k slots and denote the shifted schedule by $f_v^k(x)$. It can be easily shown that

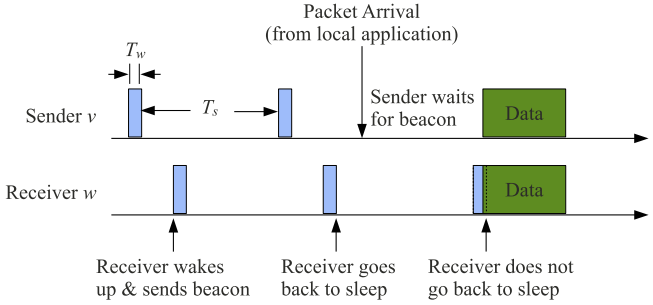


Fig. 5. RI-MAC operation. Nodes independently sleep and wakeup periodically but with the same wakeup and sleep durations. Whenever a node wakes up, it transmits a beacon frame and stays awake for a duration of T_w. It goes back to sleep if it does not start to receive any data within T_w. If node v has data to send, it waits for the beacon before transmitting its data.

$$f_v^k(x) = f_v(x) \cdot x^k \bmod (x^T - 1).$$

Let $C(v, w)$ denote the minimum number of common items of $f_v^k(x)$ and $f_w^l(x)$, for any integer l, $k \in [0, T-1]$. Then the problem of designing an optimal wakeup schedule can be formulated as follows:

$$\begin{aligned} &\text{minimize} \quad \bar{k} \\ &\text{subject to} \quad C(v,w) \geqslant m \quad \forall v \in \mathcal{A}, \forall w \in \mathbf{N}_v \end{aligned} \tag{1}$$

where $\mathbf{N}_v$ is the set of one-hop neighbors of v, $\bar{k}$ is the ensemble average of the number of wakeup slots in every T slots. Note that for a WSF $f_v(x), k_v = f_v(1)$ is the number of wakeup slots. Zheng et al. proved that for a given T and m, the necessary condition for $C(v, w) \geqslant m$ is $k_v k_w \geqslant mT$. Accordingly, they suggested two block designs, namely, symmetric WSF and asymmetric WSF. The main difference is that in the former, every node uses the same duty cycle while in the latter, different nodes can use different duty cycles. While the latter is attractive especially in EPWSN where nodes are likely to have heterogeneous energy availability, it is related to the vertex covering problem which is known to be NP-complete.

For a symmetric schedule, the necessary condition simplifies to $k^2 \geqslant mT \Rightarrow k \geqslant \sqrt{mT}$, where $k = k_v, \forall v \in \mathcal{A}$. To design a symmetric schedule, WSF uses (T, k, m)-difference set which by definition can asymptotically satisfy the condition $k \geqslant \sqrt{mT}$. A set $D = a_1, a_2, \ldots, a_k$ is said to be a (T, k, m)-difference set, if for every $d \in [0, T-1]$, there are exactly m ordered pairs (a_i, a_j) such that $d = a_j a_i \bmod T$. If $m = 1$ and k is a power of a prime number, the multiplier theorem [30] can be used to prove the existence of a $(k^2 + k + 1, k + 1, 1)$-design. Fig. 6 shows an example of a feasible schedule for $k = 2$, *i.e.*, a (7,3,1)-design with WSF $f(x) = 1 + x + x^3$.

Quorum-based wakeup scheduling. This approach uses cyclic block design technique [31] which was also used in the design of WSF [17]. Lai et al. [31] proposed two *heterogeneous* quorum-based schemes: *cyclic quorum system pair* (cqs-pair) and *grid quorum system pair* (gqs-pair). In the following description, let n denote the cycle length.

Cyclic quorum system. A cyclic quorum system is based on cyclic block design and cyclic difference sets in combinatorial theory [30]. Fig. 7a shows three quorums from a cyclic quorum system with cycle length $n = 7$.

Grid quorum system. Fig. 7b shows a grid quorum system where the slots are arranged in $\sqrt{n} \times \sqrt{n}$. A quorum can be any set containing a column and a row of elements in the array.

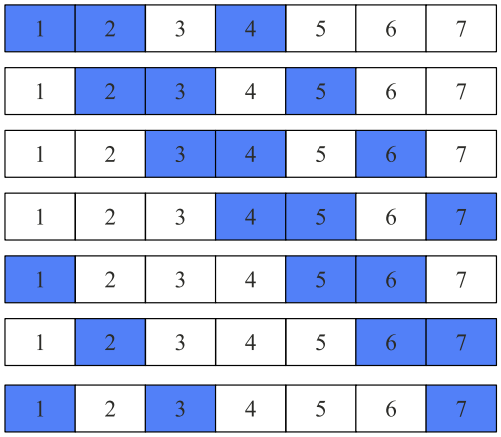
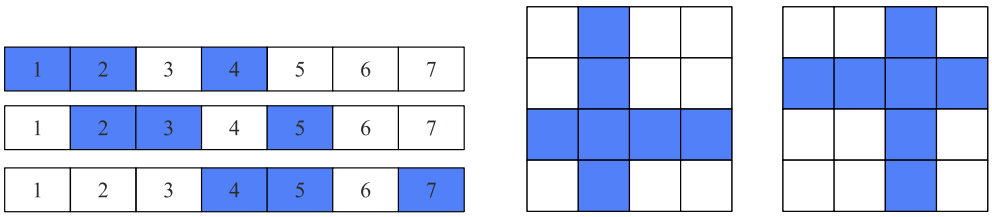


Fig. 6. Wakeup schedules of 7 nodes using the (7,3,1)-design and $f(x) = 1 + x + x^3$. Note that the schedule below is just a shifted form of the above. Each schedule has 3 active slots and any two schedules overlaps for at least 1 slot.



(a) Cyclic quorum system (b) Grid quorum system

Fig. 7. Illustrating quorum-based wakeup scheduling.

Thus, the quorum size in a square grid quorum system is $2\sqrt{n}-1$.

While prior work used homogeneous cycle length (*i.e.*, all nodes use the same cycle length and duty cycle), heterogeneous schemes allow nodes to use different cycle lengths but still guarantee that two nodes with heterogeneous quorums as their wakeup schedules can discover each other within bounded delay in the presence of clock drift.

Random. Random asynchronous wakeup scheduling does not require any effort on the transmitter or receiver to send special frames or does not require the computation of any schedule. Rather, every node wakes up in a probabilistic manner. In [32], every node v wakes up according to a Poisson process with rate λ_v. This Poisson wakeup rate essentially means that the sleep interval T_s is not constant but rather exponentially distributed. Let d_v denote the degree of node v (which v knows). Using the findings by Ikeda et al. [33], the following heterogeneous wakeup rate is proposed:

$$\lambda_v = \lambda_0 d_v^{-\beta}, \tag{2}$$

where λ_0 is an initial wakeup rate and β is a tunable parameter. Using (2), the authors proved that the worst-case delay (or equivalently, the maximum mean hitting time over all possible source–destination pairs in the network) denoted by $\overline{H}_{max}$ is

$$\overline{H}_{max} \leqslant \frac{3n^2\sqrt{d_{max}}}{\lambda_0},$$

where $n = |\mathcal{N}|, d_{max} = \max_{v\in\mathcal{N}} d_v$, and $\mathcal{N}$ is the set of all nodes in the network. To enable data transmission, every node v in the network transmits a short "hello" message every time it wakes up. This is akin to the method used by receiver-initiated protocols. For a node v that needs to transmit data, it simply wakes up and waits for a hello message from any of its neighbors. Node v then transmits its data immediately after receiving a hello message.

Discussion. A key advantage of the symmetric WSF is its low computational and storage complexity. However, this scheme cannot be easily adapted for dynamic operation. As mentioned, symmetric WSF requires all nodes in the network to use the same duty cycle. This is clearly not suitable for dynamic environments where duty cycles across nodes may differ significantly. Quorum-based wakeup scheduling is slightly better than symmetric WSF as the former allows the use of heterogeneous duty cycles across nodes. A common disadvantage of both schemes is that the sleep latency is non-deterministic (though bounds are guaranteed in static environments). Another downside that besets both schemes is that they will yield coarse-grained or low-resolution duty cycles because of limited block design options.

With regards to the random scheme, its main advantage is that it can be easily adapted to support dynamic environments through the tunable parameter β. However, because the underlying data transmission operation is similar to receiver-initiated protocols, it also suffers from the latter's weaknesses. That is, this scheme may perform poorly in both low and high duty cycle situations.

5. Synchronous schemes

Numerous synchronous schemes have been proposed because the use of time synchronization somehow eases up the analysis and design of wakeup scheduling schemes. Time synchronization in the context of wireless sensor networks is a well-studied area and numerous protocols have been proposed for this purpose [34–37]. Most synchronous schemes operate *periodically*, *i.e.*, a wakeup schedule repeats every *period*, *cycle* or *epoch* until a change is made by the scheduling algorithm. There are however *aperiodic* schemes that do not employ periodic schedules.

5.1. Periodic wakeup scheduling

Periodic wakeup scheduling schemes may operate either in a *slotted* or *unslotted* manner. In the former, the cycle is essentially broken up into equal-length slots as shown in Fig. 8. Slotted cycles are usually employed by schemes that use more than one wakeup instance per cycle.

The main problem in periodic wakeup scheduling is to determine which time interval (or intervals) in a cycle to activate so that a node can perform packet transmission or reception. This problem may look trivial from a node-level perspective but from a network-level point-of-view, selecting intervals across nodes to optimize a certain performance metric can be difficult. To achieve a desired performance, nodes must *collaborate* with each other in the process of schedule computation. We can divide the various schemes into five groups depending on the level of collaboration: (i) *neighbor-coordinated*; (ii) *path-coordinated*; (iii) *network-coordinated*; (iv) *independent*; and (v) *centralized*. The latter two are actually non-collaborative approaches.

5.1.1. Neighbor-coordinated

In this approach, a node establishes its own wakeup schedule by considering the wakeup schedules of its adjacent or neighbor nodes. To be precise, a node v calculates its wakeup schedule by consulting the schedule of all nodes $w \in \mathbf{N}_v$, where $\mathbf{N}_v$ is the set of one-hop neighbors of v. This is obviously the easiest among the collaborative schemes and requires the least effort.

The neighbor-coordinated schemes that we will be discussing in this survey are receiver-centric and every wakeup interval or slot can accommodate at most one data packet.

S-MAC. The simplest (and the first to be introduced) protocol that uses this approach is S-MAC [38]. As shown in Fig. 9, the wakeup interval is divided into three parts: (i) a portion for SYNC, (ii) a portion for RTS, and (iii) a portion for CTS. S-MAC uses *scheduled rendezvous* communication scheme wherein nodes exchange SYNC packets (in the first portion of the wakeup interval) to coordinate sleep/wakeup periods. Before a node can send a SYNC packet, it must ensure that the channel is idle by performing

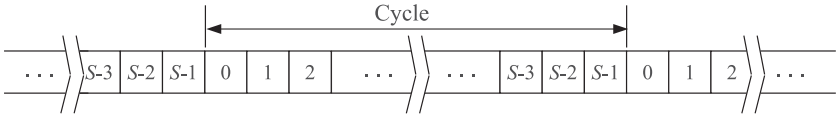


Fig. 8. A cycle with S slots.

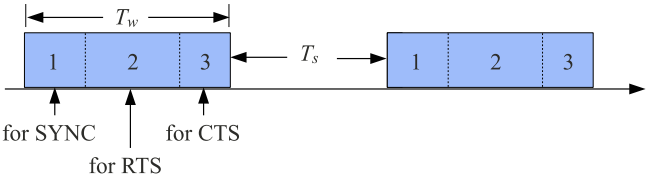


Fig. 9. Components of S-MAC wakeup interval.

carrier sensing. A node can either create its own schedule or follow a neighbor's schedule. When a node v has data to send to w, it performs carrier sensing and if the channel is idle, v sends an RTS to w in the second portion of the wakeup interval. If v detects the channel to be busy, it goes back to sleep. Node w sends back an RTS in the third portion of the wakeup interval. Data transmission occurs after the third portion of the wakeup interval. Fig. 10 illustrates the operation of the protocol.

S-MAC enhancements. Since the introduction of S-MAC, numerous improvements have been proposed to make it more energy-efficient or adaptive to changes in the network conditions.

- In Timeout MAC (T-MAC) [39], the wakeup interval is shortened with the adoption of *adaptive active time*. In S-MAC, a node's wakeup interval is essentially fixed regardless of whether it receives a packet or not. In T-MAC, if a node does not detect any further activity after a specified timeout T_A, it immediately goes back to sleep. The downside of this approach, however, is that nodes may prematurely go back to sleep which can result in higher latency and lower throughput.
- Another weakness of S-MAC is its use of fixed duty cycle which is wasted in light load conditions and insufficient in heavy load conditions. Zheng et al. [40] proposed Pattern-MAC (PMAC) to address the issues of fixed duty cycle through exchange of *sleep-wakeup patterns*. A sleep-wakeup pattern is a string of bits indicating the tentative wakeup schedule of a node over several slot times. The wakeup schedule of a node is derived from its own pattern and the patterns of its neighbor nodes.
- Dynamic S-MAC (DSMAC) [41] also attempts to make the duty cycle of S-MAC to be more dynamic, albeit in a limited matter. Specifically, DSMAC allows the duty cycle to be either 0.1, 0.2, or 0.4. These duty cycle choices ensure that any neighboring nodes will still have common wakeup intervals, as shown in Fig. 11. This can be generalized by stipulating duty cycles to be $2^k\delta$, where $k \in \{0, 1, 2, \ldots, K\}$ and $2^K\delta \leqslant 1$.

ESC. Energy-synchronized communication (ESC) [7] is one of the earliest wakeup scheduling schemes proposed for EPWSNs. ESC operates in a slotted manner. The key idea is simple: increase the number of wakeup slots when the energy supply increases and conversely, decrease the number of wakeup slots when the energy supply decreases. ESC refers to the former as (bursty) active instance increment and the latter as (bursty) active instance decrement. To facilitate the increment and decrement processes, ESC uses the notion of *cross-traffic delay* (CTD). For a node v with predecessor nodes $\mathbf{P}_v$ and successor nodes $\mathbf{S}_v$, the cross-traffic delay at v is the expected delay of every packet from any node in $\mathbf{P}_v$ to any node in $\mathbf{S}_v$ passing through v. CTD considers both sleep latency and retransmission delay. The authors demonstrated that for given schedules of predecessors and successors of node v, the CTD at v is not affected when a packet reaches v as long as the packet arrives within a certain interval. (An interval is just the contiguous set of slots between any two consecutive wakeup slots of the combined wakeup slots of $\mathbf{P}_v$ and $\mathbf{S}_v$.) This observation termed as the *stair effect* was used by the authors to design a localized $O(1)$ algorithm for schedule adjustment that minimizes the CTD at v. Note however that the $O(1)$ complexity hinges on the assumption that nodes use extremely low duty cycles.

Discussion. S-MAC and its variants rely on periodic scheduled rendezvous for synchronization, wherein the period or interval between rendezvous is determined by the duty cycle. In dynamically duty-cycled networks, the periodicity of these rendezvous will not hold and thus, communication among nodes will be difficult. As such, employing S-MAC or any of its variants in EPWSNs will be extremely challenging.

Meanwhile, ESC avoids the problem due to synchronization difficulties through the exchange of wakeup schedules among nodes.

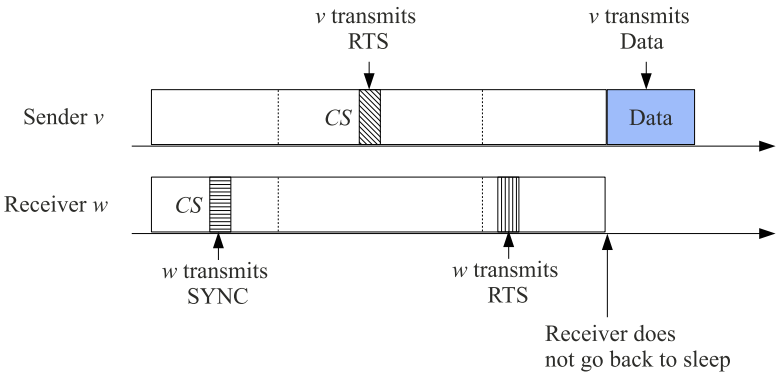


Fig. 10. S-MAC protocol operation. In the first portion, w performs carrier sensing (denoted by CS) before sending a SYNC packet. In the second portion, v sends an RTS after performing carrier sensing. In the third portion, w sends back a CTS to v. After the regular wakeup interval, v transmits its data while w continues to be awake to receive the data.

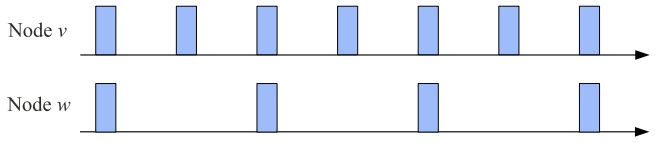


Fig. 11. Illustrating the common wakeup intervals in DSMAC even when nodes v and w choose different duty cycles. Node v's duty cycle is twice that of node w.

This enables any node u that receives a wakeup schedule advertisement from another node v to know the exact time instances at which v will be awake. ESC was designed for EPWSNs and as such, it adapts to changes in energy supply and more importantly, it minimizes sleep latency through the generation of wakeup schedules that minimizes the CTD. A major limitation of ESC is that it was designed for ultra low duty-cycled nodes. In high duty cycle scenarios, ESC will require excessive overhead to exchange and store wakeup schedules. In addition, the algorithm that computes the minimal CTD wakeup schedule will no longer be $O(1)$.

5.1.2. Path-coordinated

As mentioned in Section 2.4, sleep latency is a major problem in duty-cycled networks that significantly contributes to the end-to-end delay. Path-coordinated scheduling was therefore proposed to allow nodes along a path to coordinate their wakeup schedules such that packets traversing along the path will encounter as little delay as possible. An important requirement of this scheme is that either (i) a routing tree rooted at the sink node must already be in place; or (ii) nodes know the location of the sink node and their respective location.

Except for the Adaptive Staggered Sleep Protocol (ASLEEP) [42], all the other path-coordinated schemes were designed such that a wakeup interval or slot could accommodate the reception of at most one data packet. In ASLEEP, the wakeup interval duration is specified such that a node can communicate with all its child nodes as well as its parent node.

Wakeup patterns. Keshavarzian et al. [19] proposed several path-wide wakeup schedule patterns that aim to minimize the end-to-end sleep latency from every node to a common base station node (backward or uplink direction) and *vice versa* (forward or downlink direction). In the discussion of the wakeup patterns, it is assumed that the network is organized into levels, with the base station at level 0. The level of a node essentially indicates its minimum hop count to the base station node. Let $\mathbf{L}_k$ denote the set of nodes in level k and H denote the maximum number of hops (or maximum number of levels) in the network.

Fully-Synchronized Pattern (FSP). In this pattern, all nodes in the network sleep and wakeup at the same time. Though this is strictly not a path-coordinated pattern, it is used as the basis of the subsequent path-coordinated wakeup schedules. Fig. 12a shows FSP with a downlink transmission. Note that an uplink transmission will encounter the same latency.

Shifted Even and Odd Pattern. This pattern is derived from FSP by shifting the wakeup pattern of the nodes in even levels by $T/2$, as shown in Fig. 12b. The figure also shows an uplink transmission. Note that a downlink transmission will also experience the same latency.

Ladder pattern. This pattern is similar to the idea of *green wave* traffic light scheduling, *i.e.*, synchronizing traffic lights to turn green just in time for the arrival of vehicles from the previous intersections. This pattern can also be derived from FSP by shifting the wakeup schedule of nodes in level k by τ from that of level $k-1$.

Two-ladders pattern. One problem of the ladder pattern is that only the latency of the downlink traffic is reduced. To improve the latency of both traffic directions, two-ladders pattern is proposed, combining a forward ladder with a backward ladder.

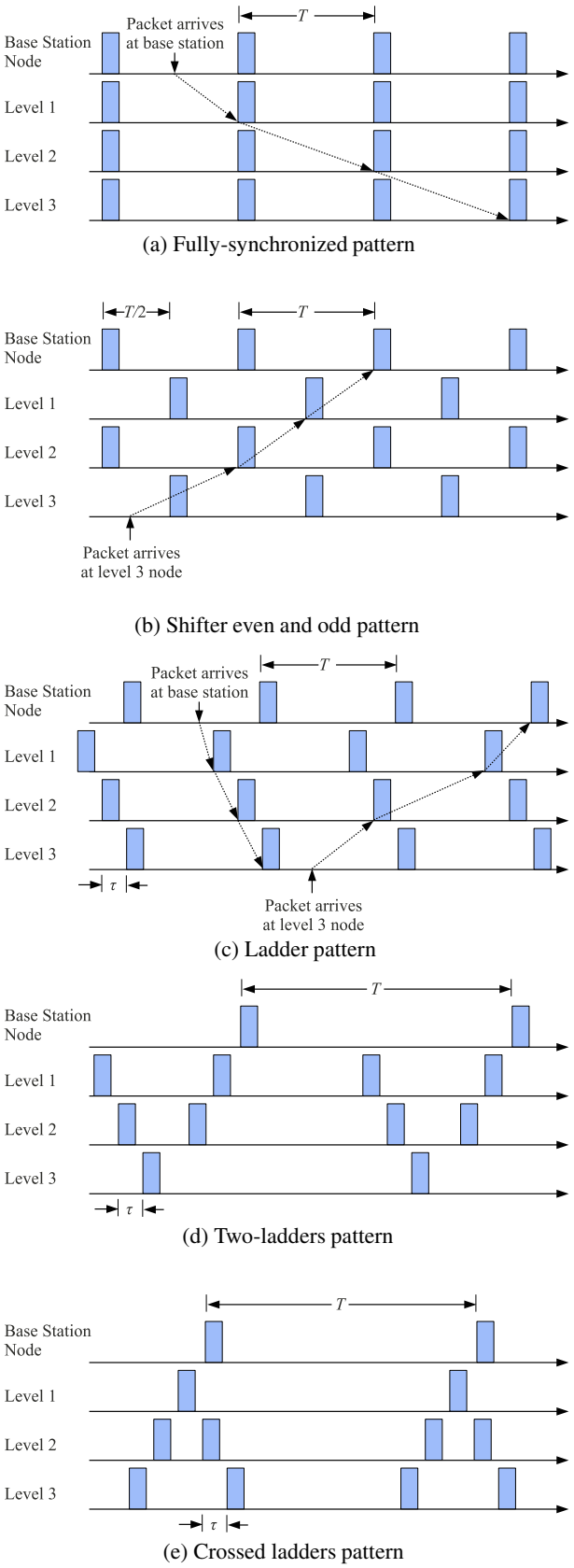


Fig. 12. Wakeup patterns.

Note that nodes in the middle levels (*i.e.*, nodes in levels $1, 2, 3, \ldots, H-1$) wakeup twice in every period T. Fig. 12d shows a two-ladders pattern.

Crossed-ladders pattern. This is an enhancement of the preceding wakeup schedule pattern where the two ladders are crossed so that the same wakeup is used for both downlink and uplink directions. The cross point can be in any of the middle levels (*i.e.*, levels $1, 2, 3, \ldots, H-1$). Fig. 12e shows a crossed ladders pattern with level 1 as cross point.

In addition to the five wakeup patterns, Keshavarzian et al. [19] also proposed the *multi-parent method* which can be independently applied to any of the five wakeup patterns. In terms of latency, the crossed-ladders and two-ladders pattern provide the best performance for both uplink and downlink traffic. The ladders pattern yields the same latency for downlink traffic but worse latency for uplink traffic. Compared to the ladders pattern, the shifted even and odd and pattern provides slightly better latency for uplink traffic but worse latency for downlink traffic. Finally, FSP performs the worst for both traffic directions.

ASLEEP. Adaptive Staggered Sleep Protocol (ASLEEP) [42] uses a staggered approach wherein nodes at the lower levels in the routing tree wakeup earlier than their ancestors. To clarify this, consider the wakeup schedules of nodes u, v, and w as shown in Fig. 13. In this illustration, u is a child of v and v is a child of w. Note that the wakeup interval of any node can actually be split into two parts which are called *talk intervals*. The first part is meant for talking to its children while the second part is meant for talking to its parent. To establish a wakeup schedule, ASLEEP uses two control messages known as *direct beacon* and *reverse beacon*. The messages are used to propagate schedule information to downstream and upstream nodes, respectively.

Staggered wakeup scheduling with multiple parents. Unlike ASLEEP which requires routing tree, Zhou and Medidi [43] proposed the use of location information to derive a staggered wakeup schedule. Prior to the computation of the wakeup schedule, the network is divided into concentric rings with the sink node located at the center. Every node must be able to identify its *ring level* with respect to the sink. Suppose that the K wakeup intervals in one cycle are $\{s_1, s_2, s_3, \ldots, s_K\}$, then a node belonging to ring n would choose to wakeup in intervals $\{s_1 + n\Delta, s_2 + n\Delta, s_3 + n\Delta, \ldots, s_K + n\Delta\}$ where Δ is an estimated packet transmission delay. One advantage of using this approach is that a node belonging to ring level n can forward its packets to any of the nodes in ring level $n-1$, hence the approach is also called *multi-parent*.

RMAC. Routing-enhanced MAC (RMAC) [44] is actually an enhancement of S-MAC protocol that uses a setup control frame known as *pioneer control frame* (PION). When a node has data to send, it initiates a request at the start of a DATA period. A PION is then generated by the sending node and this frame travels along the packet route to the sink. The PION frame essentially prepares all nodes along the route for the upcoming data from the source.

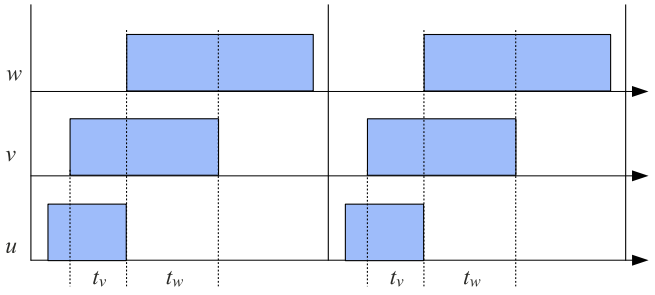


Fig. 13. ASLEEP staggered scheduling. Wakeup interval of any node can actually be split into two parts which are called *talk intervals*. For node v in the figure, t_v is meant for talking to its children (one of which is u) while t_w is meant for talking to its parent w.

Discussion. A common advantage of path-coordinated schemes is that they provide low end-to-end sleep latency for all nodes in the network. However, the need to perform path-wide coordination makes the adaptation of these schemes to EPWSNs difficult. This is because a schedule change in one node causes all the other nodes in the path to possibly re-compute their wakeup schedules. In highly dynamic environments, this may lead to excessive communication and computational overhead and in the worst case, the scheduling algorithms may fail to converge to an optimal schedule.

5.1.3. Network-coordinated

In network-coordinated scheduling, all nodes in the network collaborate to arrive at either a global wakeup schedule [45] or a per-node schedule that satisfies certain optimality goals [18]. Note that network-coordinated scheduling may either be *distributed* [45,18] or *centralized* [18,46]. In the latter, a single node is responsible for computing the wakeup schedules of all nodes in the network or at least a subset of nodes while in the former, every node is involved in the computation of their respective wakeup schedules.

Except for Sense-Sleep Trees (SS-Trees) [46], all the other network-coordinated schemes presented below were designed such that a wakeup interval or slot can be used to receive at most one data packet. In the former, nodes can transmit or receive one or more data packets in every wakeup interval.

GSA. In the global schedule algorithm (GSA) proposed by Li et al. [45], every schedule is tagged with a *schedule age* which indicates how long a schedule has existed in the network. Now consider a node v which uses a schedule with age A_v. When v receives a schedule from node w with age A_w and that $A_w > A_v$, then v adopts the schedule from w. GSA was proposed to enhance the performance of S-MAC protocol, *i.e.*, reduce the number of different schedules. This is because if a node v has neighbors with different schedules (*i.e.*, v is a border node), v must wakeup in all of its neighbors schedules which will result in higher energy consumption. Thus, by following the oldest schedule, after sufficient time, all nodes in the network will converge to a single schedule which is the oldest schedule.

Distributed DESS. Lu et al. [18] proposed two distributed algorithms to compute a wakeup schedule that minimizes the end-to-end delay. More specifically, the goal of the two algorithms is to find a single slot $s \in \{0, 1, 2, \ldots, S-1\}$ that minimizes the end-to-end delay for every source–destination pair in the network. The algorithms are called *Local-Neighbor* and *Local-DV*.

Local-neighbor: Denote s_w as the slot used by node w. Every node v in the network knows s_w, for all neighbor node w of v. Then, v chooses a slot s_v that minimizes the maximum of its delays to and from all its neighbors. This process can be repeated for several iterations.

Local-DV: Every node v in the network maintains two distance-vector tables, namely, FDV_v which stores the minimum delays from v to all other nodes, and BDV_v which stores the minimum delays from all other nodes to v. Every node v also knows FDV_w and BDV_w, for all neighbor node w of v. Node v calculates the two distance-vector tables for all possible values of s and takes note of d_{max}, the maximum value from the two distance-vector tables. Then v will choose the slot assignment that yields the lowest d_{max}.

Centralized DESS. Aside from the distributed DESS, Lu et al. [18] also proposed a centralized approach for computing wakeup schedule that minimizes sleep latency. In particular, the goal of the algorithm is to find a slot $s_v \in [0, 1, 2, \ldots, S-1], \forall v \in \mathcal{N}$ that

minimizes the *delay diameter*. The delay diameter D_f induced by a particular slot assignment f is defined as $D_f = \max_{v,w} P_f(v,w)$, where $P_f(v,w)$ is the delay along the shortest delay path between nodes v and w under the given slot assignment f. The listing in Algorithm 1 provides a pseudo-code of the centralized DESS algorithm.

Algorithm 1. Centralized DESS algorithm.

```
1: for v ∈ 𝒩 do
2:    s_v ← 0
3: endfor
4: d ← D_f
5: for i ∈ [1, n] do
6:    for v ∈ 𝒩 do
7:       for s ∈ [0, S − 1]
8:          d_v ← D_f
9:          if d_v < d then
10:            d ← d_v
11:            σ ← s
12:         else if d_v = d
13:            σ ← s with probability 0.5
14:            σ ← s_v with probability 0.5
15:         end if
16:         s_v ← σ
17:      end for
18:   end for
19: end for
```

SS-Trees. Ha et al. [46] proposed the use of Sense-Sleep Trees (SS-Trees) as a means for reducing overhearing and packet collisions in wireless sensor networks. To illustrate, consider the network topology shown in Fig. 14a. Note that if all nodes use the same wakeup schedule as in S-MAC or FSP, considerable energy might be wasted due to overhearing and packet collisions. To address these issues, the network topology can be logically partitioned into two SS-Trees, as shown in Fig. 14b and c. With the interleaved wakeup schedule of each tree shown in Fig. 14, overhearing and packet collisions will be reduced. The determination of SS-Trees and computation of their corresponding wakeup schedules is performed by the sink node. The following are the steps in the computation:

1. Every node v discovers its 1-hop neighbors.
2. Every node v forwards its link state information to the data sink t.
3. Data sink t computes the optimal SS-Tree structures and wakeup schedules with respect to the global connectivity map and application requirements.
4. Data sink t disseminates computed wakeup schedules to every node v through source routing.
5. Every node v exchanges wakeup schedule with its 1-hop neighbors.
6. Each node v follows its wakeup schedule.

Discussion. For the distributed and centralized DESS schemes, the computation of minimal sleep latency paths is a big advantage. However, this comes at a high cost in terms of communication, storage and computational overhead. In general, all the above-mentioned network-coordinated wakeup scheduling schemes suffer from several drawbacks including long convergence time, high communication and computational overhead and low scalability. Between centralized and distributed schemes, the latter schemes are more feasible as they do not require the propagation of control information to a single node which can be prohibitively expensive. Except for SS-Trees and GSA, all the network-coordinated schemes presented above can be easily adapted for dynamic wakeup scheduling. However, in highly dynamic environments, these schemes may fail to converge to an optimal wakeup schedule. As for SS-Trees, the wakeup schedules are dependent on the routing tree (which is assumed to be fixed) and is totally independent from energy availability. In the case of GSA, its objective is to come up with a common global schedule which is opposite to the objective of dynamic wakeup scheduling.

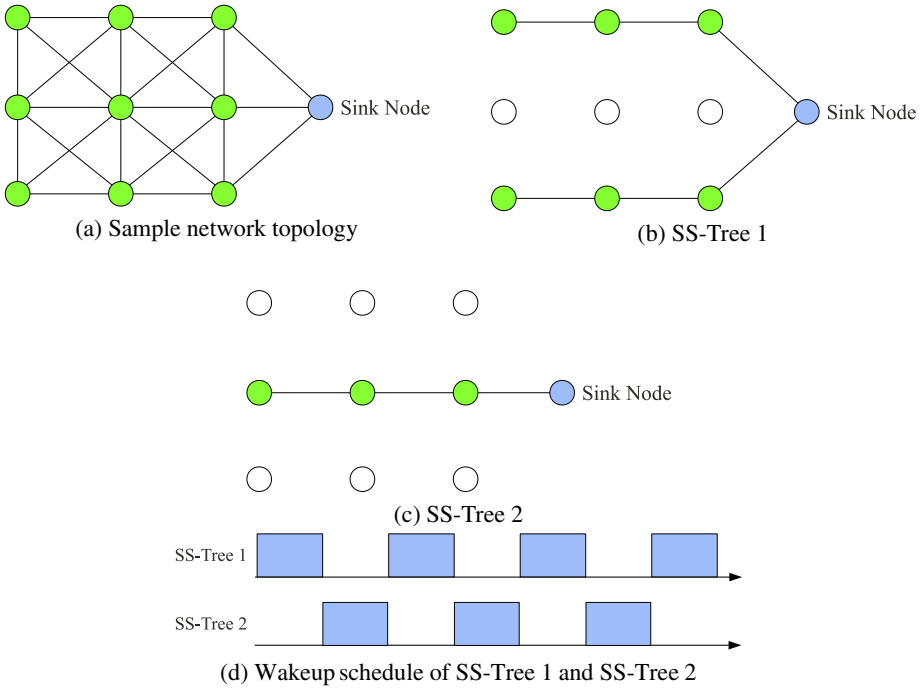


Fig. 14. Illustrating SS-Trees and wakeup schedule of each SS-Tree.

5.1.4. Uncoordinated

In uncoordinated or non-collaborative schemes, a node does not use schedule information from other nodes to compute its own wakeup schedule. Rather, a node employs control theory, or other techniques that only require local information (*i.e.*, information within the node such as queue length or duty cycle.) The uncoordinated schemes discussed below are receiver-centric, *i.e.*, every wakeup slot can accommodate the reception of at most one data packet.

Adaptive duty cycle control with queue management. Byun and Yu [47] proposed the use of a control-based technique to dynamically adjust a node's sleep interval at every cycle (and hence its wakeup schedule). Let $c_v(k)$ denote the sleep interval of a node v during the kth cycle. Then we have the following difference equation that can be used as a basis for designing a feedback controller:

$$c_v(k+1) = c_v(k) + \beta[q_v^{th} - q_v(k+1)] - \gamma[q_v(k+1) - q_v(k)], \qquad (3)$$

where $q_v(k)$ is the queue length at node v during the kth cycle, q_v^{th} is a specified queue length threshold for node v, and β and γ are control parameters that must be chosen. Note that as the queue length becomes smaller than the queue threshold, the sleep interval time increases linearly. Whereas, as the forward difference of queue length exceeds zero (because the increased forward difference of queue length induces a longer latency) the sleep interval time decreases. We highlight that the scheme only requires the local queue length information.

BRPS. Valera et al. [48] studied the sleep latency between a sender node v and receiver node w when the wakeup schedules are generated independently. The authors have shown that the expected sleep latency from v to w denoted by $E(W_{vw})$ is given by

$$E(W_{vw}) = \frac{1}{2}E(D)\left[1 + \frac{V(D)}{E^2(D)}\right], \qquad (4)$$

where $E(D)$ and $V(D)$ are the mean and variance, respectively, of all the intervals $\{D_i\}$. An interval is the number of slots between two consecutive wakeup slots in w. The result in (4) essentially means that expected sleep latency of a wakeup schedule is related to the variance of the intervals between receive wakeup slots. In particular, when the variance of the interval is low (high), the expected latency is low (high). Valera et al. [48] also introduced *schedule robustness*, which indicates the number of common wakeup slots between two schedules of the same node in different cycles. A schedule design is said to be robust if the generated schedule at cycle k is a subset or superset of the generated schedule at cycle k'. The motivation of schedule robustness is to ensure that a node v will be able to communicate with another node w even if v does not receive a schedule update from w but provided v knows an earlier schedule of w. With the above conditions in mind, the authors proposed a schedule design using bit-reversal permutation sequences (BRPS). Fig. 15 illustrates how to generate wakeup schedules for different duty cycles.

Discussion. Uncoordinated wakeup scheduling schemes have two major advantages: (i) they do not require information from other nodes to compute their wakeup schedules resulting in low communication overhead; and (ii) schedule changes in other nodes will have no effect on a node resulting in low computational overhead. As such, these schemes are very agile and are therefore suitable for EPWSNs. One major disadvantage is that the schemes may generate wakeup schedules with high sleep latencies. BRPS remedies this situation by generating wakeup schedules with low sleep interval variance to minimize the expected sleep latency.

5.2. Aperiodic wakeup scheduling

In periodic wakeup scheduling, a node's wakeup schedule usually repeats every cycle unless otherwise modified by the scheduling algorithm. In contrast, such repetition does not occur in aperiodic wakeup scheduling because the decision to wakeup or sleep in every slot is random. The periodic wakeup scheduling schemes presented in this section use bi-directional wakeup slots.

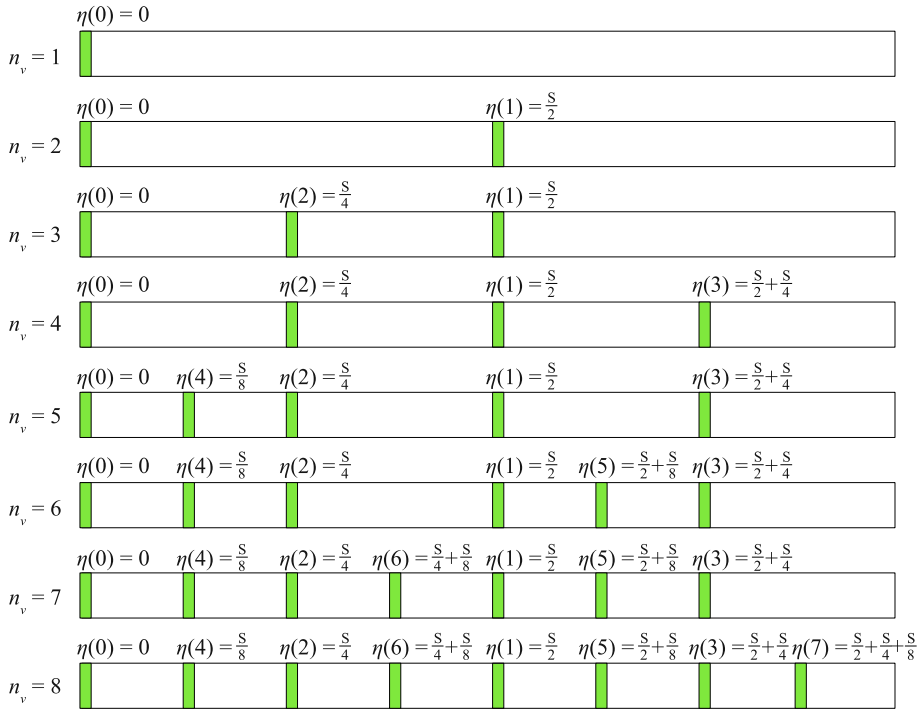


Fig. 15. Generating wakeup schedules for different duty cycles, where n_v denotes the number of wakeup slots of node v. If $n_v = 1$, then we can simply decide to position the slot, which we label as $\eta(0)$ at 0. (For the purpose of labeling the wakeup slots, we use the notation $\eta(i)$, where i 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 $S/4$. The illustration shows an example up to $n_v = 8$. Note that since there are S slots per cycle, the duty cycle is n_v/S.

Recall that a bi-directional wakeup slot can accommodate the transmission or reception of one or more data packets.

Dai and Basu. Dai and Basu [49] proposed several random wakeup scheduling schemes where every node exchanges minimal schedule information with its neighbors to determine whether they are asleep or awake in a particular slot. In particular, every node exchanges its pseudo-random number generator (pRNG) seed and cycle position. A node can be in any one of the following states in a slot: ON-RX, ON-TX and OFF, with corresponding probabilities p_{rx}, p_{tx} and $p_{off} = 1 - p_{rx} - p_{tx}$, respectively. Note that with knowledge of the pRNG seed and cycle and the probabilities, any node v will be able to know the state of any other node w. Dai and Basu [49] proposed six random wakeup scheduling schemes, described in the following:

S1: Random scheme without schedule exchange. In every slot, nodes independently choose states in a probabilistic manner. Suppose a node v is in the ON-TX state in a slot. For unicast traffic, v randomly selects one of the nodes w, where $w \in \mathbf{N}_v$ and $\mathbf{N}_v$ is the set of one-hop neighbors of v. For broadcast traffic, v simply transmits.

S2: One-hop schedule exchange. In this scheme, nodes exchange pRNG seed and cycle prior to the network operation. A node v that is in ON-TX selects a node w that is in ON-RX to transmit its packet to.

S3: One-hop schedule exchange & nodes turn OFF. This scheme improves on S2 through opportunistic energy saving strategies. If a node w is in the ON-RX state in a slot and none of its neighbors are in the ON-TX state in the same slot, then w changes its state to OFF. Similarly, if a node v is in the ON-TX state and none of its neighbors are in the ON-RX state, then v modifies its state to OFF. Also, a node v in the ON-RX state modifies its state to OFF if more than one of its neighbors is in the ON-TX state.

S4: Two-hop schedule exchange and nodes turn OFF. This scheme improves on S3 through two-hop schedule exchange. Each node exchanges the pRNG seed and cycle position and those of its neighbors to all of its neighbors. Since a node has two-hop schedule information, it will be able to calculate the modified schedule of all of its neighbors. After determining the modified neighbor schedule, a node modifies its state from ON-TX to OFF if there are no neighbors in the ON-RX state in that slot. A node v in the ON-TX state *randomly* selects a neighbor node w that is in the ON-RX state under the modified schedule to transmit to.

S5: Modified seedex. Similar to S4, each node exchanges two-hop schedule information and is able to obtain the modified schedule of all of its neighbors. In each slot, a node v in the ON-TX state selects one of its neighbors w in the ON-RX state as a potential receiver. The actual transmission is dictated by a probability p that is inversely proportional to the number of neighbors of w that are in the ON-TX state. In addition, (i) nodes that do not transmit (due to probabilistic backoff) are modified to the OFF state, and (ii) nodes modify their states from ON-RX and ON-TX to OFF if there are no nodes in the ON-TX or ON-RX state, respectively, in a node's neighborhood.

S6: Combination of S4 and S5. This scheme combines the features of S4 and S5.

Ghidini and Das. Ghidini and Das [50] proposed a random scheme that does not require any form of information exchange. Nodes therefore rely on the probability of being simultaneously awake to effect data transfer. To motivate the design of their random wakeup scheduling scheme, Ghidini and Das introduced the notion of *connection delay* for nodes v and w which is the time interval between the current slot and the first slot at which both v and w are simultaneously awake. A related concept is *connection duration* which is the time interval between first and last slot when v and w are simultaneously and continuously awake. The authors proposed a Markov Chain-based duty cycling scheme with control vector $[\delta, \tau, \gamma]'$ where δ is the target duty cycle, τ is the slot duration, and γ is the memory coefficient of the Markov Chain. The last parameter affects the transition probabilities α (transition probability from sleep to wakeup) and β (transition probability from wakeup to sleep) as follows:

$$\alpha = \gamma\delta \tag{5}$$

and

$$\beta = \gamma - \alpha. \tag{6}$$

Note that $\gamma \in [0, 1/(1-\delta)]$, and setting $\gamma = 1$ means that the decision at every slot is totally independent from the previous decisions.

Discussion. Aperiodic schemes are essentially random wakeup scheduling schemes where the decision to sleep or wakeup is performed at the beginning of every slot. As such, these schemes may potentially have higher computational overhead. But because no schedule is exchanged among the nodes, communication overhead is either zero or minimal. In terms of adaptability to dynamic environments, the schemes proposed by Dai and Basu [49] are not amenable for adaptation because of their use of pRNG. Note that a node's wakeup slots are determined by its pRNG which is totally independent from the dynamics of the node's environment. As for the scheme proposed by Ghidini and Das [50], it can be easily adapted through the control parameter δ. In terms of sleep latency, the use of random wakeup slots in aperiodic schemes results in stochastic sleep latency as well. The scheme by Ghidini and Das [50] is slightly better as it provides a mechanism to improve the sleep latency, *i.e.*, through the minimization of connection delay.

6. Hybrid schemes

Hybrid schemes switches between synchronous and asynchronous scheduling depending on the prevailing network conditions. The switching between the two modes is motivated by the fact that asynchronous schemes perform well under light load conditions while synchronous approaches are usually better in heavy load conditions [51].

Z-MAC. Z-MAC [52] combines the strengths of TDMA and CSMA while "offsetting" their weaknesses. Nodes employ neighbor discovery and slot assignment schemes to enable every node to obtain a slot such that no two nodes within two hops are assigned the same slot. The current implementation uses DRAND [53], a distributed version of RAND [54] for slot assignment. The slot assigned by DRAND is then used to derive the time frame of every node v. The time frame of v specifies the time slot at which v is allowed to transmit. When v has data to transmit, it first determines whether it is the owner of the slot and whether it is in *low contention level* (LCL) mode or *high contention level* (HCl) mode. A node v is in the latter state if it receives an explicit congestion notification from one of its two-hop neighbors. The transmission rule is as follows:

- If v is the owner of the slot, it performs random backoff (within fixed time period T_o) followed by clear channel assessment. If the channel is idle, it proceeds to transmit its data; otherwise it repeats the process.
- If v is not the owner of the slot and it is in LCL mode, or it is in HCl mode and none of its two-hop neighbors own the current slot, v waits for T_o and then performs random backoff within a contention window $[T_o, T_{no}]$. Node v then performs clear channel assessment and if the channel is idle, it proceeds to transmit its data; otherwise, it repeats the process.
- If v is not the owner of the slot and it is in HCl mode, v postpones its transmission until it finds a time slot that either it owns or is not owned by any of its two-hop neighbors.

Table 3
Comparison of wakeup scheduling schemes.

Scheme	Dynamics adaptation	Latency-aware	Duty cycle range	Processing complexity	Overhead
B-MAC [24]	✓		M–H	N	L
X-MAC [25]	✓		M–H	N	L
BoX-MAC [26]	✓		M–H	N	L
RI-MAC [27]	✓		M	N	L
RC-MAC [28]	✓		M	N	L
Nguyen et al. [29]	✓		M	M	L
Symmetric WSF [17]			L–H (C)	M	L
Quorum-based [31]	✓		L–H (C)	H	H
Lee and Eun [32]	✓		L–H	L	L
S-MAC [38]			L–H	L	M
T-MAC [39]			L–H	L	M
PMAC [40]			L–H	M	M
DSMAC [41]			L–M (C)	L	M
ESC [7]	✓	✓	L	M	M
FSP [19]			L–H	M	M
Wakeup Patterns [19]		✓	L–H	M	H
ASLEEP [42]		✓	L–H	M	H
Zhou and Medidi [43]		✓	L–H	M	H
RMAC [44]		✓	L–H	M	H
GSA [45]			L–H	M	H
Distributed DESS [18]	✓	✓	L–H	M	H
Centralized DESS [18]	✓	✓	L–H	H	H
SS-Trees [46]			M (T)	H	H
Byun and Yu [47]	✓		L–H	H	L
BRPS [48]	✓		L–H	L	L
Dai and Basu [49]			U	M	M
Ghidini and Das [50]	✓		L–H	M	L
Z-MAC [52]	✓		L–H	M	H
[i]Queue-MAC [55]	✓		L–H	M	H

N–*None*; L–*Low*; M–*Medium*; H–*High*; C–*Coarse*; T–*Tree-dependent*; U–*Uncontrollable*.

Queue-MAC and iQueue-MAC. Queue-MAC [55] is a hybrid CSMA/TDMA protocol that uses queue length information to switch between the two MAC protocols. A fixed period of the superframe is allocated for CSMA while a variable period of the superframe is allocated for TDMA. A parent node (cluster head or router) monitors the queue lengths of its child nodes. The protocol operates according to the following simple rule: a parent node allocates additional TDMA slots to its heavily loaded children nodes. The authors also proposed iQueue-MAC [51] as an enhancement of Queue-MAC [55] to provide better support for multi-hop packet forwarding and multi-channel operation.

Discussion. By design, hybrid schemes are well suited for environments where data traffic is dynamic. To a certain extent, they can also be adapted for dynamic duty-cycled operation. The ease or difficulty of adaptation essentially depends on the underlying schemes used in the two modes of operation. The major disadvantage of hybrid schemes is that they suffer from higher communication and computational overhead.

7. Conclusion and open issues

Environmentally-powered wireless sensor networks (EPWSNs) are adaptively duty-cycled to attain energy-neutral operation in the face of dynamic energy supply. Duty-cycled operation necessitates the use of *wakeup schedules* which indicate the time intervals at which a node activates its radio transceivers to perform either packet transmission or reception. In this paper, we presented a survey of the state-of-the-art in wakeup scheduling.

7.1. Wakeup scheduling schemes

Numerous wakeup scheduling schemes have been proposed for wireless sensor networks to address the unique challenges of duty-cycled node operation. We have summarized recent results on wakeup scheduling and classified the approaches into three main categories, namely asynchronous, synchronous and hybrid. Table 3 enumerates the various wakeup scheduling schemes that were presented in this survey, consisting of MAC-layer and non-MAC-layer approaches. The table also qualitatively assesses the schemes against the considerations discussed in Section 2.4 to establish suitability for EPWSNs. In summary, wakeup scheduling schemes used in EPWSNs must consider (i) environment dynamics (topology, energy supply, load), (ii) sleep latency; (iii) duty cycle range; (iv) processing complexity; and (v) communication and storage overheads.

7.1.1. Asynchronous schemes

Asynchronous schemes were the earliest and simplest protocols proposed for wakeup scheduling. A node with data to transmit simply wakes up to transmit without regard on whether other nodes are awake to receive. One of the major advantages of asynchronous operation is that they do not require any form of time synchronization which can be difficult and may entail considerable energy consumption. Another major advantage of asynchronous approaches is that they do not require any computational and storage overhead since no schedules are stored. The only overhead is the transmission of special frames (*e.g.*, preamble or beacon) prior to data transmission. There are four possible approaches to enable data transmission across nodes which are not aware of each other's wakeup schedule: (i) transmitter-initiated – a sending node transmits a preamble prior to the data frame; (ii) receiver-initiated – a node transmits a beacon every time it wakes up to indicate that it's ready to receive data; and (iii) combinatorial and random – combinatorial design is used to ensure that two or more nodes are simultaneously awake within bounded time while in random, wakeup rates are randomized and used in tandem with receiver-initiated approaches. Except for WSF, it is straightforward to adapt asynchronous schemes to be adaptive to the dynamics of its operating environment. However, both transmitter-initiated and receiver-initiated protocols cannot be used in the entire duty cycle range.

In both schemes, transmitters will suffer excessive waiting time in low duty cycle regimes while the use of beaconing in receiver-initiated protocols might cause excessive contention in high duty cycle regimes.

7.1.2. Synchronous schemes

In contrast to asynchronous schemes, synchronous schemes require nodes to be time synchronized. The majority of the proposed schemes presented fall under this category and we divide them further into two major sub-categories depending on the periodicity of the wakeup schedule. Most synchronous schemes operate periodically, *i.e.*, a wakeup schedule repeats every period, cycle or epoch until a change is made by the scheduling algorithm. There are however schemes wherein the decision to sleep or wakeup in every slot is random resulting in aperiodic wakeup schedules.

In periodic wakeup scheduling, the main problem is to determine a subset of time intervals within a cycle to wakeup so that a node can perform packet transmission or reception. The selection of appropriate time intervals is usually driven by an objective to optimize a certain performance metric such as throughput or latency. Nodes must therefore collaborate with each other in the process of computing their respective wakeup schedules. There are three levels of collaboration: (i) neighbor-coordinated – a node establishes its own wakeup schedule by considering the wakeup schedules of its neighbor nodes; (ii) path-coordinated – nodes along a path coordinate their wakeup schedules such that packets traversing along the path will have little delay; and (iii) network-coordinated – all nodes in the network collaborate to arrive at either a global wakeup schedule or a per-node schedule that satisfies certain performance objectives. In addition, uncoordinated schemes are also possible where nodes independently compute their respective wakeup schedules, employing control theory or other techniques that only require local information (*i.e.*, information within the node such as queue length or duty cycle.).

Except for ESC which was specifically designed for low duty cycle networks, all periodic schemes can support low–high duty cycles. In terms of suitability for dynamic environments, path-coordinated schemes, together with SS-Trees and GSA, are not amenable for adaptation to such environments. This is because considerable coordination effort is needed to support dynamic wakeup schedules. Meanwhile, network-coordinated and path-coordinated schemes require high computational complexity. Notably, BRPS together with neighbor-coordinated schemes (except PMAC) entail low computational complexity. In terms of overhead, uncoordinated schemes entail the lowest overhead followed by neighbor-coordinated schemes. Path-coordinated as well as network-coordinated schemes require higher overhead because every node needs to coordinate their respective schedules with a larger number of nodes.

7.1.3. Hybrid schemes

Finally, several schemes that combine synchronous and asynchronous scheduling have been proposed. The switching between the two modes is motivated by the fact that asynchronous schemes perform well under light load conditions while synchronous approaches are usually better in heavy load conditions. These schemes are suitable in dynamic environments and they can support low–high duty cycle networks. They also have moderate computational complexity but their major drawback is that they usually require higher storage and communication overhead. This is because additional messages are exchanged and stored to enable nodes to ascertain the proper mode of operation to use.

7.2. Open research issues

As mentioned in Section 2.4, sleep latency is a major challenge in duty-cycled networks. Path-coordinated schemes, including ESC and DESS appear to be heading in the right direction as they address this particular problem. In particular, wakeup slots or intervals are positioned such that the waiting time from node to node (in the direction of the expected traffic flow) is minimized. However, the design of these schemes suffer from one major flaw: they assume that packet transmissions are always successful. In practical sensor networks where wireless link qualities have high variation, packets retransmissions are more the norm than the exception. As such, the low latency advantage of path-coordinated schemes will vanish in real-world deployments. Indeed, it might be difficult to have deterministic guarantees in stochastic environments. Currently, there is very little research that looks at minimizing sleep latency in the presence of unreliable wireless links.

Schedule representation is another important area that needs to be studied further. Note that in most synchronized schemes, every node needs to store the wakeup schedules of all its neighbors. Because sensor nodes have limited memory, schedules must be represented in a compact manner. The most straight-forward approach to represent a schedule is to use an S-bit array, where S is the number of slots per cycle. A '0' bit means that the corresponding bit position is a sleep slot while a '1' bit means that the corresponding bit position is a wakeup slot. Note however that this approach is not scalable. If a scheme uses high value for S and the network is dense, then considerable amount of memory is needed for schedule storage. We note that our previous work on BRPS represents a first attempt to tackle the problem of (i) minimizing sleep latency in presence of unreliable wireless links; and (ii) minimal overhead schedule representation.

Another possible area of research is the consideration of node mobility in wakeup schedule design. Most of the existing schemes assume that the sensor nodes and sink are stationary. Asynchronous and non-collaborative synchronous schemes are good candidates for these scenarios because their lack of coordination requirement makes them robust to network topology changes. In the presence of node mobility, schemes that require coordination may not converge to an optimal schedule or may generate excessive overhead.

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 Comput. 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 Commun. Surv. Tutor. 13 (3) (2011) 443–461.

[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 Trans. Emb. Comput. Sys. 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. Di Francesco, A. Passarella, Energy conservation in wireless sensor networks: a survey, Ad Hoc Netw. 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] S. Kim, K.-S. No, P. Chou, Design and performance analysis of supercapacitor charging circuits for wireless sensor nodes, IEEE J. Emerg. Sel. Top. Circ. Syst. 1 (3) (2011) 391–402.

[12] A. Harb, Energy harvesting: state-of-the-art, Renew. Energy 36 (2011) 2641–2654.

[13] National Renewable Energy Laboratory. <http://http://www.nrel.gov> (accessed 31.05.12).

[14] F. Simjee, P.H. Chou, Efficient charging of supercapacitors for extended lifetime of wireless sensor nodes, IEEE Trans. Power Electron. 23 (3) (2008) 1526–1536.
[15] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Networks 38 (4) (2002) 393–422.
[16] E. Lattanzi, E. Regini, A. Acquaviva, A. Bogliolo, Energetic sustainability of routing algorithms for energy-harvesting wireless sensor networks, Comput. Commun. 30 (14–15) (2007) 2976–2986.
[17] R. Zheng, J.C. Hou, L. Sha, Asynchronous wakeup for ad hoc networks, in: Proc. ACM MobiHoc, 2003.
[18] G. Lu, N. Sadagopan, B. Krishnamachari, A. Goel, Delay efficient sleep scheduling in wireless sensor networks, in: Proc. IEEE INFOCOM, 2005.
[19] A. Keshavarzian, H. Lee, L. Venkatraman, Wakeup scheduling in wireless sensor networks, in: Proc. ACM MobiHoc, 2006.
[20] Y. Gu, T. He, Data forwarding in extremely low duty-cycle sensor networks with unreliable communication links, in: Proc. ACM SenSys, 2007.
[21] S. Guha, C.-K. Chau, P. Basu, Green wave: latency and capacity-efficient sleep scheduling for wireless networks, in: Proc. IEEE INFOCOM, 2010.
[22] W. Ye, J. Heidemann, D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, in: Proc. IEEE INFOCOM, 2002.
[23] Y. Wu, S. Fahmy, N. Shroff, Optimal sleep/wake scheduling for time-synchronized sensor networks with qos guarantees, IEEE/ACM Trans. Network. 17 (5) (2009) 1508–1521.
[24] J. Polastre, J. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in: Proc. ACM SenSys, ACM, New York, NY, USA, 2004, pp. 95–107.
[25] M. Buettner, G.V. Yee, E. Anderson, R. Han, X-mac: a short preamble mac protocol for duty-cycled wireless sensor networks, in: Proc ACM SenSys, 2006, pp. 307–320.
[26] D. Moss, P. Levis, Box-mac: exploiting physical and link layer boundaries in low-power networking, Tech. rep., technical, Report SING-08-00, 2008.
[27] Y. Sun, O. Gurewitz, D.B. Johnson, RI-MAC: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks, in: Pro. ACM SenSys, 2008, pp. 1–14.
[28] P. Huang, C. Wang, L. Xiao, H. Chen, RC-MAC: a receiver-centric medium access control protocol for wireless sensor networks, in: Proc. IWQoS Workshop, 2010, pp. 1–9.
[29] K. Nguyen, V.-H. Nguyen, D.-D. Le, Y. Ji, D. Duong, S. Yamada, A receiver-initiated mac protocol for energy harvesting sensor networks, in: Y.-S. Jeong, Y.-H. Park, C.-H.R. Hsu, J.J.J.H. Park (Eds.), Ubiquitous Information Technologies and Applications, Lecture Notes in Electrical Engineering, vol. 280, Springer, Berlin Heidelberg, 2014, pp. 603–610.
[30] I. Anderson, Combinatorial Designs and Tournaments, Oxford University Press, 1998.
[31] S. Lai, B. Ravindran, H. Cho, Heterogenous quorum-based wake-up scheduling in wireless sensor networks, IEEE Trans. Comput. 59 (11) (2010) 1562–1575.
[32] C.-H. Lee, D.Y. Eun, A distributed wake-up scheduling for opportunistic forwarding in wireless sensor networks, in: Proc. IEEE GLOBECOM, 2010, pp. 1–5.
[33] S. Ikeda, I. Kubo, M. Yamashita, The hitting and cover times of random walks on finite graphs using local degree information, Theor. Comput. Sci. 410 (1) (2009) 94–100.
[34] J. Elson, K. Römer, Wireless sensor networks: a new regime for time synchronization, SIGCOMM Comput. Commun. Rev. 33 (1) (2003) 149–154.
[35] M. Maróti, B. Kusy, G. Simon, A. Lédeczi, The flooding time synchronization protocol, in: Proc. ACM SenSys, 2004.
[36] S. Ganeriwal, D. Ganesan, H. Shim, V. Tsiatsis, M.B. Srivastava, Estimating clock uncertainty for efficient duty-cycling in sensor networks, in: Proc. ACM SenSys, 2005, pp. 130–141.
[37] S. PalChaudhuri, A.K. Saha, D.B. Johnson, Adaptive clock synchronization in sensor networks, in: Proc. ACM IPSN, 2004, pp. 340–348.
[38] W. Ye, J. Heidemann, D. Estrin, An energy-efficient mac protocol for wireless sensor networks, in: Proc. IEEE INFOCOM, vol. 3, 2002, pp. 1567–1576.
[39] T. van Dam, K. Langendoen, An adaptive energy-efficient mac protocol for wireless sensor networks, in: Proc. ACM SenSys, 2003, pp. 171–180.
[40] T. Zheng, S. Radhakrishnan, V. Sarangan, PMAC: an adaptive energy-efficient mac protocol for wireless sensor networks, in: Proc. IEEE Parallel and Distributed Processing Symposium, 2005.
[41] P. Lin, C. Qiao, X. Wang, Medium access control with a dynamic duty cycle for sensor networks, in: Proc. IEEE WCNC, vol. 3, 2004, pp. 1534–1539.
[42] G. Anastasi, M. Conti, M. Di Francesco, Extending the lifetime of wireless sensor networks through adaptive sleep, IEEE Trans. Ind. Inform. 5 (3) (2009) 351–365.
[43] Y. Zhou, M. Medidi, Sleep-based topology control for wakeup scheduling in wireless sensor networks, in: Proc. IEEE SECON, 2007, pp. 304–313.
[44] S. Du, A. Saha, D. Johnson, RMAC: a routing-enhanced duty-cycle mac protocol for wireless sensor networks, in: Proc. IEEE INFOCOM, 2007, pp. 1478–1486.
[45] Y. Li, W. Ye, J. Heidemann, Energy and latency control in low duty cycle mac protocols, in: Proc. IEEE WCNC, vol. 2, 2005, pp. 676–682.
[46] R.W. Ha, P.-H. Ho, X.S. Shen, J. Zhang, Sleep scheduling for wireless sensor networks via network flow model, Comput. Commun. 29 (13–14) (2006) 2469–2481.
[47] H. Byun, J. Yu, Adaptive duty cycle control with queue management in wireless sensor networks, IEEE Trans. Mobile Comput. 12 (6) (2013) 1214–1224.
[48] A.C. Valera, W.-S. Soh, H.-P. Tan, Energy-neutral scheduling and forwarding in environmentally-powered wireless sensor networks, Ad Hoc Netw. 11 (3) (2013) 1202–1220.
[49] L. Dai, P. Basu, Energy and delivery capacity of wireless sensor networks with random duty-cycles, in: Proc. IEEE ICC, 2006.
[50] 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.
[51] S. Zhuo, Z. Wang, Y.-Q. Song, Z. Wang, L. Almeida, iQueue-MAC: a traffic adaptive duty-cycled mac protocol with dynamic slot allocation, in: Proc. IEEE SECON, 2013, pp. 95–103.
[52] I. Rhee, A. Warrier, M. Aia, J. Min, M.L. Sichitiu, Z-mac: a hybrid mac for wireless sensor networks, IEEE/ACM Trans. Netw. 16 (3) (2008) 511–524.
[53] I. Rhee, A. Warrier, L. Xu, Randomized dining philosophers to tdma scheduling in wireless sensor networks, Tech. rep., Computer Science Department, North Carolina State University, 2004.
[54] S. Ramanathan, A unified framework and algorithms for (t/f/c)dma channel assignment in wireless networks, in: Proc. IEEE INFOCOM, 1997, pp. 900–907.
[55] S. Zhuo, Y.-Q. Song, Z. Wang, Z. Wang, Queue-mac: a queue-length aware hybrid csma/tdma mac protocol for providing dynamic adaptation to traffic and duty-cycle variation in wireless sensor networks, in: Proc. IEEE Workshop on Factory Communication Systems (WFCS), 2012, pp. 105–114.