Wakeup Scheduling in Wireless Sensor Networks

Abtin Keshavarzian
Robert Bosch Corporation
Research & Tech. Center
4009 Miranda Ave
Palo Alto, CA 94304
abtin.keshavarzian
@rtc.bosch.com

Huang Lee
Stanford University
Electrical Engineering Dept.
Packard, 305 Serra St.
Stanford, CA 94309
huanglee@stanford.edu

Lakshmi Venkatraman [*]
Robert Bosch Corporation
Research & Tech. Center
4009 Miranda Ave
Palo Alto, CA 94304
lakshmi.venkatraman
@rtc.bosch.com

ABSTRACT

A large number of practical sensing and actuating applications require immediate notification of rare but urgent events and also fast delivery of time sensitive actuation commands. In this paper, we consider the design of efficient wakeup scheduling schemes for energy constrained sensor nodes that adhere to the bidirectional end-to-end delay constraints posed by such applications. We evaluate several existing scheduling schemes and propose novel scheduling methods that outperform existing ones. We also present a new family of wakeup methods, called *multi-parent* schemes, which take a cross-layer approach where multiple routes for transfer of messages and wakeup schedules for various nodes are crafted in synergy to increase longevity while reducing message delivery latencies. We analyze the power-delay and lifetime-latency tradeoffs for several wakeup methods and show that our proposed techniques significantly improve the performance and allow for much longer network lifetime while satisfying the latency constraints.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Algorithm, Performance, Design, Theory

Keywords

Wireless Sensor Network, Wakeup Scheduling, Energy-efficient Algorithms, Cross-layer Protocols , Power-delay Tarde-off, Graph Coloring Algorithms.

[*]Additional authors: Krishna Chintalapudi, Dhananjay Lal, Bhaskar Srinivasan. All with Robert Bosch Corporation, Research and Technology Center.
Please see the additional author section at the end of the paper for authors' affiliations.

1. INTRODUCTION

A large class of critical monitoring and sensing-actuation systems (*e.g.,* fire alarm sprinkler systems or wireless sensor based control systems) are deployed specifically to (a) detect events that occur rarely but require immediate notification and (b) transfer delay sensitive actuation commands to a particular node or a set of nodes in the network. Such systems necessitate a design that can provide bidirectional delay guarantees. On the other hand, the design of sensor network based systems that comprise energy constrained nodes is typically dictated by longevity concerns. Therefore, the design of such systems must not only strive to reduce average power consumption but also provide packet delivery guarantees over potentially multiple hops.

A popular approach towards increasing longevity of sensor networks is by employing sleep scheduling where nodes stay in low-power or sleep modes for most of the time, periodically waking up to check for activity [1–4]. This increased longevity, however, comes at the cost of increased message delivery latency since a forwarding node has to wait until its next-hop neighbor awakens and is ready to receive the message. Researchers in ad hoc and sensor networks continue to search for new wakeup techniques to save power without suffering the large latency penalties associated with the wakeup process. Current methods can be divided into two main categories:

1) Scheduled wakeups: In this class, the nodes follow deterministic (or possibly random) wakeup patterns [1–11]. Time synchronization among the nodes in the network is generally assumed. However, asynchronous wakeup mechanisms [9–11] which do not require synchronization among the different nodes are also categorized in this class. Although asynchronous methods are simpler to implement, they are not as efficient as synchronous schemes, and in the worst case their guaranteed delay can be very long.

2) Wakeup on-demand (out-of-band wakeup): It is assumed that the nodes can be signaled and awakened at any point of time and then a message is sent to the node. This is usually implemented by employing two wireless interfaces. The first radio is used for data communication and is triggered by the second ultra low-power (or possibly passive) radio which is used only for paging and signaling. STEM [12] and its variation [13], and passive radio-triggered solutions [14] are examples of this class of wakeup methods. Although these methods can be optimal in terms of both delay and energy, they are not yet practical. The cost issues, currently limited available hardware options which results in limited range and poor reliability, and stringent system requirements prohibit the widespread use and design of such wakeup techniques. Consequently, there is a need for efficient scheduled wakeup schemes which are reliable and cost-effective and can also guarantee the delay and lifetime constraints.

In this paper, we focus on the synchronous scheduled wakeup methods which provide bidirectional delay guarantees. We analyze and compare the existing methods and introduce new efficient wakeup methods that outperform the existing ones. We present a novel class of wakeup methods called *multi-parent* schemes which assign multiple parents (forwarding nodes) with different wakeup schedules to each node in the network. This method takes a cross-layer approach and exploits the existence of multiple paths between the nodes in the network to *significantly* improve the energy-efficiency of wakeup process and therefore increase the lifetime of the network while meeting the message delay constraints.

We derive the best-case, worst-case, and generally the distribution of delay for many existing and our new wakeup schemes, and also characterize the trade-off between power consumption (or lifetime) and guaranteed delay for many different wakeup mechanisms. In a practical example, we show that by using our proposed wakeup schemes, the lifetime increases from 40 months for the best existing method to a notable 65 months for our proposed multi-parent scheme which achieves two additional years of lifetime while providing the same delay guarantees.

Furthermore, we formulate the process of parent assignment for multi-parent methods as a graph coloring problem, and prove that it is NP-complete, but we present an efficient heuristic algorithm to solve this problem and evaluate its performance through simulation.

The rest of this paper is organized as follows: In Section 2, we review the existing synchronous methods and describe the differences between previous related studies and our approach to the wakeup scheduling problem. Section 3 presents the general framework and assumptions underlying our approach. In Section 4, the delay distributions of different wakeup schemes are derived. The multi-parent technique is described in Section 5. Different wakeup schemes are compared in Section 6. The parent assignment problem is studied in Section 7, and finally, Section 8 concludes this paper.

2. EXISTING METHODS

A good survey of wakeup-based power management techniques can be found in [8]. In [1, 2] a MAC protocol for sensor networks called S-MAC, was introduced where the idea of duty-cycling and scheduled sleeping of the nodes is incorporated in the MAC layer. Each node follows a periodic active/sleep cycle, and the nodes that are close to one another synchronize their active cycles together. T-MAC [3] is an extension of the previous protocol which adaptively adjusts the sleep and awake periods based on estimated traffic flow to increase the power savings and reduce delay.

DMAC [4] is an efficient data gathering protocol for sensor networks where the communication pattern is restricted to an unidirectional tree. It uses staggered wakeup schedules to create a pipeline for data propagation to reduce the latency of data collection process significantly. Similar wakeup schemes are used in [5, 6]. As we will see in Section 4, this scheme provides good delay in one direction but it is not efficient when bidirectional delay guarantees are required. In [5] a protocol is proposed for scheduling the wakeup time of different nodes such that *detection delay* is minimized, *i.e.,* each point in the environment is sensed within some finite interval of time. This scheme is mainly useful when there are many redundant sensor nodes in the network such that the same point is covered by multiple sensors.

In a recent paper [7] by Lu et. al, the authors formulate the wakeup scheduling as a graph-theoretical problem. They consider low traffic network with arbitrary communication flows and show that minimizing the end-to-end *communication delay* is in general NP-hard. However, they present efficient heuristic methods to find the best schedules and prove the optimality of different wakeup patterns under specific conditions for special tree and ring topologies.

In our model the goal is to minimize the worst-case end-to-end *overall* delay which includes both transmission delay and detection delay. Unlike the model in [7] where general traffic flows are assumed, we consider a specific yet very common and practical traffic pattern where the base station (a central node) is either the source or originator of the messages (forward direction) or it is the sink or the final receiver of the messages (backward direction) (see Figure 1). By focusing on this traffic pattern we are able to design very energy-efficient methods and guarantee a significantly better delay performance than existing methods. For example, in a four-hop network when the nodes wake up on average *once every two seconds*, our proposed wakeup methods guarantee a *worst-case* overall delay of less than $3s$ over four hops in both forward and backward directions while DMAC achieves a worst-case delay of $6s$ (in both directions), and the best existing scheme gives a delay of $4.1s$. Even for data collection or monitoring applications where the backward delay is important, our proposed methods perform better than the existing schemes. For the same system, DMAC guarantees a delay of $2.1s$ for backward direction, while our proposed multi-parent technique achieves a delay of less than $1.15s$, which is almost half of the wakeup period of the nodes!

3. MODEL AND ASSUMPTIONS

For the rest of the discussion in the paper, our sensor network comprises of several tens of energy constrained sensor nodes that either notify an event to a base station or receive commands/queries from the base station, possibly over multiple hops. The base station station is assumed to be less energy constrained, however it does not necessarily provide greater radio bandwidth or range than the regular nodes[1].

Traffic Model: Figure 1 depicts the two kinds of communication paths in the network, namely,
1) *Forward direction* (downlink): The base station sends a message to one of nodes in the network.
2) *Backward direction* (uplink): A regular node sends a message to the base station.

Several sensor nodes today, are often equipped with passive event detection capabilities that allow a node to detect an event even while it is in sleep mode. Still others provide ultra low-power, low-rate periodic sampling mechanisms for rare event detection. Upon the detection of an event, the sensor node is immediately woken up (within sev-

[1]This work can be easily generalized to the case where there are multiple base stations.

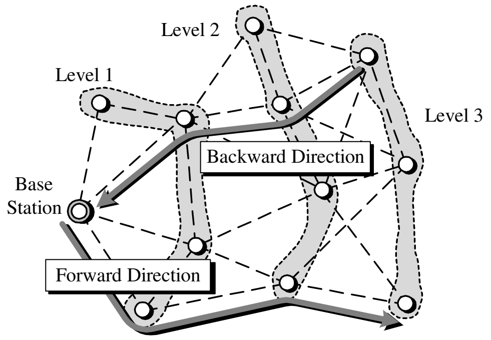


Figure 1: Network and Traffic Model.

eral μsec) and is ready to transmit a notification message to the base-station. Similarly, the base-station is often required to transmit imperative commands or queries to sensor nodes that may originate asynchronously. Messages in either directions, thus, originate at random times (asynchronously) and this implies that messages may potentially originate at an inopportune time when all other nodes in the network are in sleep mode and not ready to receive the message. While these messages occur infrequently, they reflect urgency, as such their delivery demands non-negotiable worst case delay bounds. For the rest of the discussion in the paper, *delay* is defined as the time duration between generation of a message at a node (base-station or a regular node) until its eventual delivery at the destination node.

Channel Sniffing and Wakeup: Nodes in the network wake up from time to time and scout the channel for activity. This is performed by listening to the channel for a very short period of time and measuring the received signal strength. If the signal strength exceeds a pre-determined threshold, the node remains awake in an attempt to receive a possible transmission, otherwise it powers itself down. This entire process is called *sniffing* the channel.

Wakeup for sniffing constitutes the most frequent operation in the network and consequently is typically the most energy consuming activity. To illustrate the importance of the wakeup power consumption, we consider the following example: According to the data sheet of Chipcon CC1100 radio [19], if the nodes wake up once every second the average current consumption over the one second is $15\mu A$ which gives a charge draw of $15\mu C$ per wakeup. This current draw may seem negligible in comparison with average current draw of 15 mA for reception or transmission of packets at data rate 250Kbps. However, in a day of operation of the network, the energy consumed by a node due to wakeup will add up to $15\mu A \times 3V \times 86400s = 3.9J$. This much energy can be used to transmit or receive almost 21 Mbits of data. In many applications, the overall traffic that passes through a node in one day is much less than this!

The length of the sniffing period and the energy consumed while performing a wakeup, critically determine the longevity of the network. In practice, the sniffing length is determined by several hardware limitations such as the warm up time of the radio, and the minimum time required to reliably detect a signal in the channel. Sniffing period is typically in the order of hundreds of μsec to few $m$sec.

Time Synchronization: We assume that a network-wide time synchronization protocol maintains a consistent notion of time between various sensor nodes in the network. Time synchronization in wireless sensor networks is well-researched and several implementations (*e.g.,* [15, 16]) can achieve synchronization within few of μsec. As such, synchronizing nodes within an accuracy of few $m$sec, which is required by the wakeup schedules, is a relatively easy task.

Although the time synchronization protocol may create additional energy burden for the system, in most delay-sensitive applications this extra energy cost is either negligible in comparison with the energy consumed by the wakeup process[2] and/or will be compensated by the energy saving that can be achieved by employing an efficient synchronous wakeup method. Therefore, for delay-sensitive applications, synchronous wakeup methods are preferred due to their overall energy-efficiency.

Network Topology: Each node in the network is represented by a node in a graph and a link between two nodes signifies their ability to communicate with each other. An initial connectivity graph is formed by the base station during network initialization followed by occasional updates to account for temporal changes in the wireless channel (*e.g.,* see [17, 18]). While wireless link qualities are subject to changes temporally, two static nodes that are connected via a reliable link (high signal to noise ratio) rarely experience a complete change in their connectivity over short periods of time. In this work we assume that the sensor network deployment is dense enough such that every node has few neighbors with highly reliable links. In such a network, if only reliable links are used for communication, the connectivity graph itself is not subject to frequent changes. As such, we assume that the connectivity graph formed by using the reliable links of the network is stationary.

Variables and Notation: There are N nodes in the network. Levels are assigned to various nodes in the network in a breadth-first order based on the connectivity graph. The base station is assigned a level 0. Essentially, level of a node signifies the minimum number of hops from the node to the base-station. This is illustrated in Figure 1. L_k denotes the set of nodes in level k. The maximum number of hops (or maximum number of levels) in the network is denoted by h.

Our goal in this paper is to analyze the worst case delay observed by any node in the network. Since nodes that are the farthest from the base-station (nodes in the highest level) experience the longest delays, we consider the delay for only such nodes in our analysis. We use $\mathbf{D}_{\triangleright}$ and $\mathbf{D}_{\triangleleft}$ to represent the random variables for the delay seen by a node in level h in forward and backward directions respectively. In other words, $\mathbf{D}_{\triangleright}$ shows the delay seen by a message sent from the base station to a node in L_h, and $\mathbf{D}_{\triangleleft}$ shows the delay of a message from a node in L_h to the base station. The delay is random due to the uncertainty in the arrival time of the messages. We assume that if a message arrives within a period of Δ, the arrival time is uniformly distributed over

[2]Consider the previous example with CC1100 radio: Assume that each node need to re-synch every 100 seconds, and each time the radio should be in active mode for at most $5ms$ to compensate for any clock drift during the $100s$ and exchange synchronization packets. Note that the clock drift between two nodes in 100 seconds caused by a poor quality crystal with 40ppm inaccuracy is at most $4ms$. Thus, each synch period consumes $5ms \times 15mA \times 3V = 225\mu J$. This adds up to $864 \times 225\mu J = 0.19J$ over a day which is much smaller than the $3.9J$ for the wakeup process.

this period. Many arrival point processes including Poisson process satisfy this condition (*e.g.,* see [21]). The notation $X \sim \mathbf{U}[\alpha, \beta]$ is used to show that X is a continuous random variable with uniform distribution over the range of $[\alpha, \beta]$.

We denote the energy consumed for each wakeup by E_o. The value of E_o depends only on the hardware and the duration that a node stays awake in each wakeup. For example, for CC1100 we have $E_o = 3V \times 15\mu C = 45\mu J$.

In a scheduled wakeup scheme each node must be able to decide upon the times for sniffing the channel for possible receptions. The simplest scheme is to schedule a node to wake up periodically after a fixed time interval. In a more sophisticated scheme, each node may follow a periodic *wakeup pattern i.e.,* a sequence of pre-determined wakeup times that exhibit periodicity. We denote the period of a wakeup pattern by T. Note that during one period T the node may wake up multiple times, therefore to compare consistently across various schemes we define the *effective wakeup period* T_{eff} as,

$$T_{\text{eff}} = \lim_{\tau \to \infty} \frac{\tau}{N_\tau}. \tag{1}$$

Here, N_τ represents the number of wakeups in a time duration τ. So on average, the nodes wake up once every T_{eff} seconds. We also define the *effective wakeup rate* as,

$$R_{\text{eff}} = \frac{1}{T_{\text{eff}}}. \tag{2}$$

The power consumption due to wakeups is then given by,

$$P_{\text{wakeup}} = \frac{E_o}{T_{\text{eff}}} = R_{\text{eff}} E_o. \tag{3}$$

4. WAKEUP PATTERNS

In this section, we present different wakeup patterns and derive their corresponding delay distributions. In our analysis the tradeoff between power (or equivalently lifetime) and latency is characterized by obtaining the relationship between delay distribution (in particular the worst-case delay) and the effective wakeup period for each wakeup pattern. Note that from (3) the power consumed by wakeup process is mainly related to T_{eff}.

Throughout this section, to compare different wakeup patterns we provide numerical values for delay and lifetime for two example scenarios:

(a) ***Fixed-power case***: In this case, different patterns are compared based on their worst-case guaranteed delay when they consume the same amount of power and therefore provide the same lifetime. We assume that the amount of power allocated for the wakeup process is fixed such that

$$P_{\text{wakeup}} = 0.5E_o \Rightarrow T_{\text{eff}} = 2s.$$

So the nodes wake up on average once every two seconds.

(b) ***Fixed-delay case***: In this case, we assume that the worst-case delay required by the application is fixed such that maximum delay should be less than one second:

$$\max(\mathbf{D}_\triangleright, \mathbf{D}_\triangleleft) \leqslant 1.$$

Then, the amount of power each pattern consumes to satisfy this delay requirement is calculated. To convert the power consumption to network lifetime, we assume a fixed battery capacity[3] equal to $(2.4 \times 10^8)E_o$. This value is obtained assuming 2/3 of the capacity of two AA batteries ($1000mAh$)

[3]In general, the total energy of the battery should be divided among different processes executed on a node where the wakeup process is one of them. However, here to simplify the calculation we assume that the portion of the battery capacity devoted to the wakeup process is fixed.

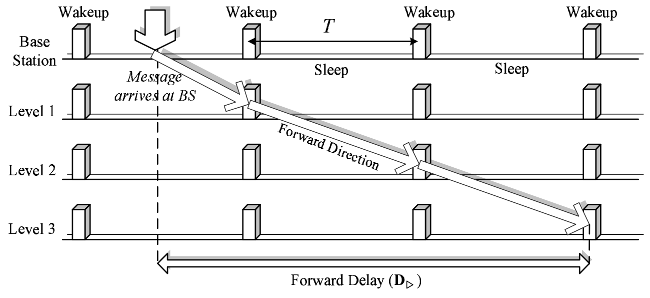


Figure 2: Fully Synchronized Wakeup Pattern.

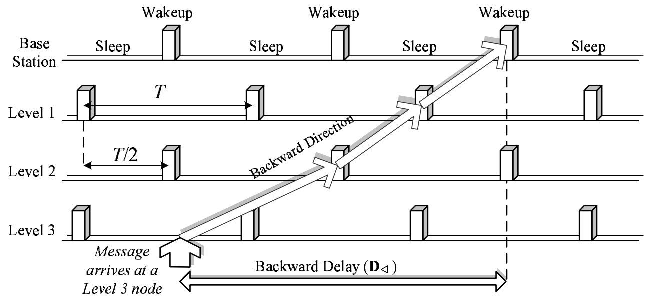


Figure 3: Shifted Even and Odd Pattern.

used on a Chipcon CC1100 radio with $E_o = 45\mu J$ ($15\mu C$) per wakeup. In both cases, we further assume that the network has $h = 4$ hops.

4.1 Fully Synchronized Pattern

In this pattern which is shown in Figure 2, all the nodes in the network wake up at the same time according to a simple periodic pattern with a fixed period $T_{\text{eff}} = T$. This pattern is very similar to the S-MAC protocol [1, 2]. In the figure, the delay of a message that arrives at the base station and is forwarded to a node in level 3 is shown. The worst case delay in the network is simply hT and due to the symmetry of the pattern, the distribution of delay in both forward and backward directions is the same:

$$\begin{gathered} \mathbf{D}_\triangleright, \mathbf{D}_\triangleleft \sim \mathbf{U}\,[\,(h-1)T_{\text{eff}}\ ,\ hT_{\text{eff}}\,]\,, \\ \mathbb{E}(\mathbf{D}_\triangleright) = (h - \tfrac{1}{2})T_{\text{eff}}. \end{gathered} \tag{4}$$

In our two example scenarios, for the fixed-power case with $h = 4$ and $T_{\text{eff}} = 2s$:

$$\mathbf{D}_\triangleright, \mathbf{D}_\triangleleft \sim \mathbf{U}\,[6, 8]\ \Rightarrow \max(\mathbf{D}_\triangleright, \mathbf{D}_\triangleleft) = 8s,$$

and for the fixed-delay case, the nodes should wake up every $T_{\text{eff}} = 250ms$ which gives $P_{\text{wakeup}} = 4E_o$ and a network lifetime of 23.1 months.

4.2 Shifted Even and Odd Pattern

This pattern is derived from the previous one by shifting the wakeup pattern of the nodes in even levels by $T/2$. It is shown in Figure 3. The figure also shows the worst-case delay scenario: A message arrived to a level 3 node immediately after the wakeup time of the parent of the node. In this case, the first hop requires T seconds and the following $(h-1)$ hops each takes $T/2$ seconds. The worst-case delay is therefore $(h+1)T/2$ and the distribution of delay is:

$$\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft} \sim \mathbf{U}[\ (\tfrac{h-1}{2})T_{\text{eff}}\ ,\ (\tfrac{h+1}{2})T_{\text{eff}}\ ], \qquad (5)$$
$$\mathbb{E}(\mathbf{D}_{\triangleright}) = \tfrac{h}{2}T_{\text{eff}}.$$

In our examples, for the fixed-power case

$$\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft} \sim \mathbf{U}[3,5]\ \Rightarrow \max(\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft}) = 5s,$$

and for the fixed-delay case to achieve one second delay

$$T_{\text{eff}} = 400ms \Rightarrow P_{\text{wakeup}} = 2.5E_o$$

which gives a lifetime of 37 months, which is much better than the 23.1 months of the first pattern.

Note that by this simple modification, the delay for this pattern is almost half of the delay for the synchronized pattern, and the lifetime is significantly increased. In fact, in [7] it is proved that in a network with tree topology this pattern provides the best overall *average* delay among all simple (one-wakeup-per-period) patterns with different shifts (see Theorem 2 and its conclusion in [7] for more conditions).

4.3 Ladder Pattern

In this pattern, the nodes still follow the simple periodic pattern but the wakeup patterns of different levels are staggered. Figure 4 shows this pattern where the wakeup are staggered in the forward direction. As explained in [4]: "This idea is very similar to the common practice of synchronizing the traffic lights to turn green (wake up) just in time for the arrival of vehicles (packets) from the previous intersections (hops)".

This pattern has been suggested by many authors [4–7] and has been given different names such as staggered wakeup (DMAC) [4], streamlined wakeup [5], fast path algorithm (FPA) in [6]. We refer to this pattern as *ladder wakeup*.

The time difference between the wakeup times of two nodes in adjacent levels is denoted by τ. By decreasing this value, the forwarding time of the message can be minimized. However, an intermediate node should fully receive the message before it can forward it to the next level, so the value of τ is limited by the size of the message and the time required to transmit it. Typically τ should be in the order of tens of $m$sec.

This wakeup pattern is no longer symmetric, so the forward and backward delay distributions are different. In the forward direction the first hop requires between zero to T seconds and then the next $(h-1)$ hops each require only a short period of length τ, so (note that $T_{\text{eff}} = T$):

$$\mathbf{D}_{\triangleright} \sim \mathbf{U}[\ (h-1)\tau\ ,\ T_{\text{eff}} + (h-1)\tau\ ], \qquad (6)$$
$$\mathbb{E}(\mathbf{D}_{\triangleright}) = \tfrac{T_{\text{eff}}}{2} + (h-1)\tau.$$

For backward direction, the first hop again requires at most T seconds, and the next hops each takes $(T-\tau)$ seconds. Note that the wakeup time of the base station does not impact the forward delay[4]. So in order to reduce the backward delay the base station wakes up after (instead of before) the wakeup time of the L_1 nodes as shown in Figure 4. The distribution of the backward delay is given by:

$$\mathbf{D}_{\triangleleft} \sim \mathbf{U}\Big[\ (h-2)(T_{\text{eff}}-\tau)+\tau\ ,\ (h-1)T_{\text{eff}} - (h-3)\tau\ \Big], \qquad (7)$$
$$\mathbb{E}(\mathbf{D}_{\triangleleft}) = (h-\tfrac{3}{2})T_{\text{eff}} - (h-3)\tau.$$

In the two numerical example cases, we assume $\tau = 50ms$. For the fixed-power case with $T_{\text{eff}} = 2s$:

$$\mathbf{D}_{\triangleright} \sim \mathbf{U}[0.15, 2.15]\,,\ \ \mathbf{D}_{\triangleleft} \sim \mathbf{U}[3.95, 5.95]\,,$$

so the maximum delay is $5.95s$; and for the fixed-delay case to achieve one second delay in both directions we need

$$T_{\text{eff}} = 350ms \Rightarrow P_{\text{wakeup}} = 2.86E_o,$$

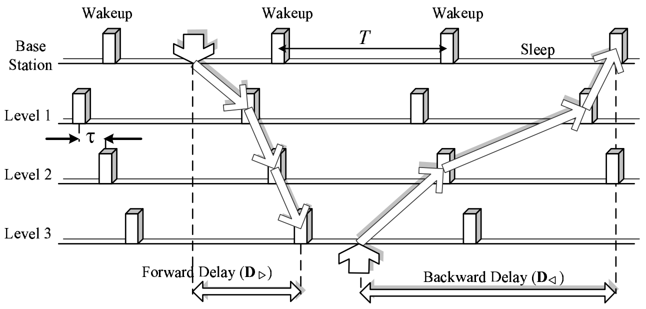
which gives 32.4 months as the lifetime of the network.

Note that the delay in the forward direction is significantly reduced but the backward delay is almost the same as the first pattern. So when the delays in both directions are considered, there is no major improvement in the worst-case delay or the lifetime of the network.

The pattern shown in Figure 4 is the forward ladder pattern. This pattern can be reversed to create the backward ladder pattern which improves the backward direction and is essentially the same as the wakeup method used in DMAC or FPA [4–7].

4.4 Two-Ladders Pattern

To improve the delay in both directions we can combine a forward ladder with a backward ladder. This pattern is shown in Figure 5. A similar idea is proposed in [7]. Note that the nodes in the middle levels $(L_1, \ldots, L_{h-1})$ wake up twice in every period T, so the effective wakeup period is $T_{\text{eff}} = T/2$. Since the pattern is symmetric the distribution



Wakeup
Wakeup
Wakeup
Base Station
T
Sleep
Level 1
τ
Level 2
Level 3
Forward Delay ($\mathbf{D}_{\triangleright}$)
Backward Delay ($\mathbf{D}_{\triangleleft}$)

Figure 4: Ladder Wakeup Pattern (Forward).

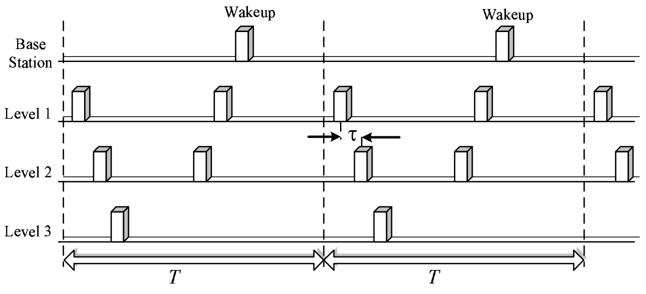
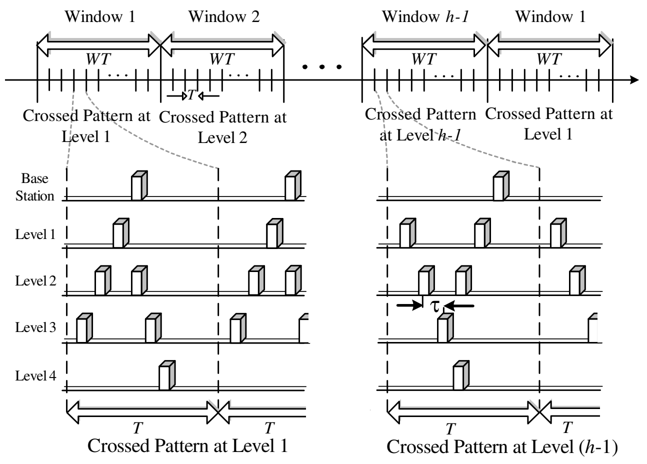


Figure 5: Two-Ladders Pattern.

[4]The same statement is true for the backward delay and the pattern of the nodes in the last level.



Window 1
Window 2
Window h-1
Window 1
WT
WT
WT
WT
τ
Crossed Pattern at Level 1
Crossed Pattern at Level 2
Crossed Pattern at Level h-1
Crossed Pattern at Level 1
Base Station
Level 1
Level 2
Level 3
Level 4
τ
T
T
T
T
Crossed Pattern at Level 1
Crossed Pattern at Level (h-1)

Figure 6: Crossed-ladders Pattern.

of delay in both directions is the same:

$$\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft} \sim \mathbf{U}\left[(h-1)\tau,\ \ 2T_{\text{eff}} + (h-1)\tau\right], \qquad (8)$$
$$\mathbb{E}(\mathbf{D}_{\triangleright}) = T_{\text{eff}} + (h-1)\tau.$$

In our examples, for the fixed-power case with $T_{\text{eff}} = 2$,

$$\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft} \sim \mathbf{U}\left[0.15, 4.15\right] \ \Rightarrow \max(\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft}) = 4.15s,$$

which is better than the $5s$ for the even-odd pattern. For the fixed-delay case with the worst-case delay of one second

$$T_{\text{eff}} = 425ms \Rightarrow P_{\text{wakeup}} = 2.35E_o,$$

and the network lifetime is 39.3 months, which is 2.3 months longer than the even-odd pattern.

This pattern is more efficient than the preceding ones but it is also more complex. The nodes no longer follow a simple "one wakeup per period T" pattern. In addition, note that the base station and the nodes in the last level (leaf nodes) wake up only once, unlike other nodes which wake up twice in every period T. Consequently, they save energy and consume less power for wakeup in comparison with other nodes.

4.5 Crossed-Ladders Pattern

To enhance the previous wakeup pattern, we can cross the two ladders at one of the wakeup points so that the same wakeup is used in both forward and backward directions. The cross point can be in any of the middle levels $(L_1, L_2, \ldots, L_{h-1})$. We refer to such pattern as a *crossed ladder* pattern. However, this technique saves energy for the nodes on the level on which the wakeups are crossed. So to distribute the energy saving over all levels, we propose to change the wakeup pattern over time as shown in Figure 6. For a window of WT seconds ($W >> 1$) wakeup pattern with two ladders crossed at first level is used. Then the network switches to a different wakeup pattern with crossed ladders at level 2, and in the same way it proceeds. After the crossed pattern at level $(h-1)$ the network goes back to the first pattern and the whole cycle repeats. Over a full cycle of $(h-1)WT$ seconds, the nodes in the intermediate levels wake up twice every T seconds in $(h-2)$ windows, and once in every T seconds in one window. Therefore, the effective wakeup period is

$$T_{\text{eff}} = \frac{(h-1)WT}{2W(h-2)+W} = (\frac{h-1}{2h-3})T. \qquad (9)$$

For this scheme the forward and backward delays are the same as in (6) but the effective wakeup period should be scaled by the term $((2h-3)/(h-1))$:

$$\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft} \sim \mathbf{U}\left[(h-1)\tau, \left(\tfrac{2h-3}{h-1}\right)T_{\text{eff}} + (h-1)\tau\right], \quad (10)$$
$$\mathbb{E}(\mathbf{D}_{\triangleright}) = \left(\tfrac{2h-3}{2h-2}\right)T_{\text{eff}} + (h-1)\tau.$$

Let us consider the two example scenarios. For the fixed-power case with $T_{\text{eff}} = 2s$ and $h = 4$

$$\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft} \sim \mathbf{U}\left[0.15, 3.48\right] \ \Rightarrow \max(\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft}) = 3.48s.$$

The worst-case delay in both directions is 3.48s which is much better than the 4.15 guaranteed by the previous pattern. For the fixed-delay case to provide a worst-case delay of one second we need

$$T_{\text{eff}} = 510ms \Rightarrow P_{\text{wakeup}} = 1.96E_o,$$

which gives 47.2 months as the lifetime of the network. This is 20% longer (8 additional months) than 39.2 months of the two-ladders pattern. This pattern is the most complex one, but also it is the most energy-efficient among the patterns considered in this section.

5. MULTI-PARENT METHOD

In this section, we describe a new method, called *multi-parent* technique, which improves the performance of the wakeup process significantly. This method can be independently applied to any of the wakeup patterns from the previous section. We show that the multi-parent idea along with the forward ladder pattern yields a very efficient and yet simple wakeup scheme which is more efficient than all the previous methods.

5.1 Motivation and Assumptions

In many application scenarios and network deployments, the network is dense and therefore most of the nodes at higher levels have many neighbors and they can communicate with many lower level nodes. We take advantage of this fact in the multi-parent idea and exploit the full connectivity of the network. Instead of using a tree network topology where a single parent is assigned to each node in the network and the messages are always forwarded through the same fixed path, multiple paths and multiple parents with different wakeup schedules are associated with each node in the network. Basically, in the multi-parent idea when a message arrives to a node in the network, depending on its arrival time it chooses the fastest path in the network to get to its destination. For example, if the node has two parents it forwards the message to the the parent which will wake up earlier. Another message that comes at a later time may find the other parent/path to be optimal at that moment.

The main assumption for the multi-parent method is that we can divide the nodes in the network into multiple disjoint groups such that at least one parent from each group can be assigned to *any* node in the network. For example, Figure 7 shows a graph in which all the nodes are divided into two groups, namely red group and blue group. Note that each node in the network has one red parent (mother) and one blue parent (father). The base station is a special node which belongs to both groups and can act as both parents. We call base station a *purple* node. We defer further discussions on network partitioning and parent assignment to

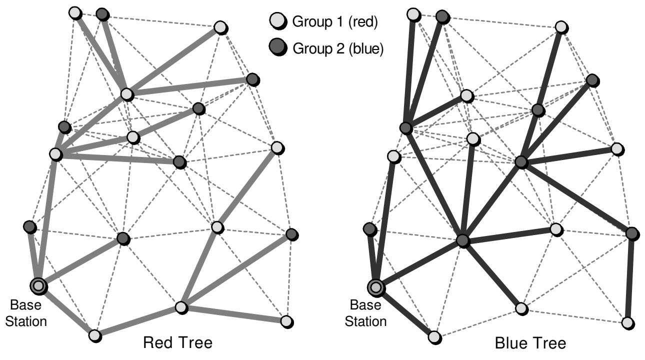


Figure 7: Example of partitioning the network.

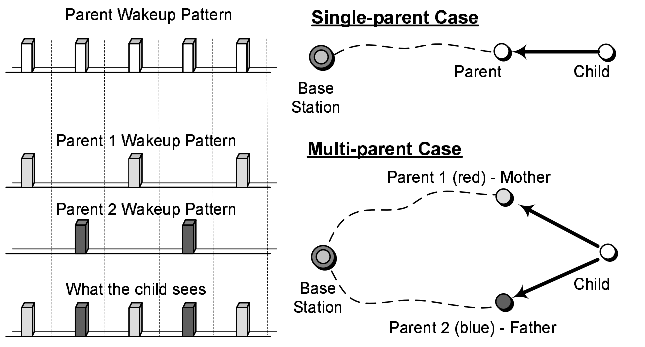


Figure 8: Multi-parent Method.

Section 7, and in the rest of this section we analyze the effect of the multi-parent idea and the improvement that can be achieved by applying this method to different wakeup patterns.

5.2 Description and Analysis

We denote the number of groups by g, so the nodes are divided into g groups and every node has g parents, each from a different group. In the previous wakeup patterns, all the nodes in the same level wake up at the same time according to a periodic wakeup pattern. We define one period of the wakeup pattern as a *frame*. We consider g consecutive frames and associate each frame with a different group. The nodes in each group follow the same wakeup pattern only in their corresponding frame and sleep in the other $(g-1)$ frames. This is illustrated in Figure 8 for $g=2$ and a simple periodic wakeup pattern: When two parents (mother and father) are assigned to each node, if the mother is awake, the father can sleep and vice versa and the child node does not see any difference from the single-parent case. The base station belongs to all groups so it should wake up in all frames.

Now computing the delay distribution of different wakeup patterns with the multi-parent idea is in order. We show that the multi-parent idea can reduce the backward delay significantly by almost a factor of g, but the forward delay is not impacted by this idea.

5.2.1 Distribution of Backward Delay

As it can be seen from Figure 8, with multi-parent method, the child node still gets the same opportunities to send a message and sees the same pattern as in the single-parent case. So the delay in backward direction remains the same while the nodes in the network wake up g times less frequently as the single-parent case. Therefore, the expression for distribution of delay has the same form as in the single-parent case but the effective wakeup period is scaled down by a factor of g, *i.e.,* T_{eff} is replaced by (T_{eff}/g). For example, for the even-odd pattern:

$$\mathbf{D}_{\triangleleft} \sim \mathbf{U}[\ (\tfrac{h-1}{2g})T_{\text{eff}}\ ,\ (\tfrac{h+1}{2g})T_{\text{eff}}\ ], \qquad (11)$$
$$\mathbb{E}(\mathbf{D}_{\triangleleft}) = (\tfrac{h}{2g})T_{\text{eff}}.$$

So the same backward delay can be guaranteed while all nodes wake up much less frequently, or equivalently if the nodes wake up at the same rate as before, the backward delay is reduced by a factor of g. All the expressions for the distribution of delay of different wakeup patterns are summarized in Table 1.

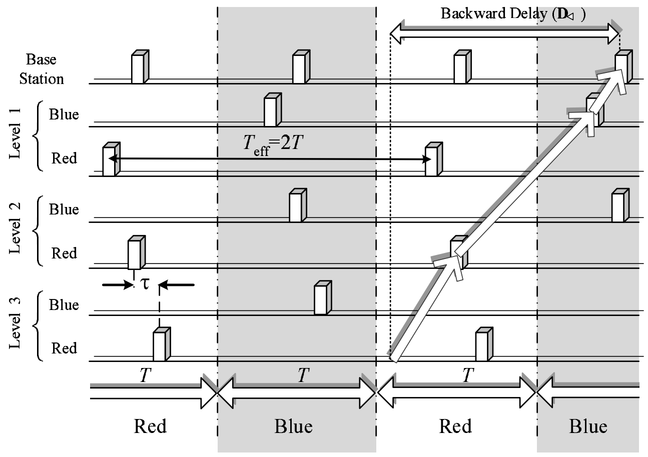


Figure 9: Multi-parent Forward Ladder Pattern.

5.2.2 Distribution of Forward Delay

The forward delay can be divided into two segments: First segment is the time from the arrival of the message to the base station till it reaches one of the parents of the destination node, and the second segment is the time to send from the parent to the node. With the multi-parent idea, the first segment of the delay is reduced as we can use different paths to send the message from the base station to one of the parents very fast. However, this idea increases the second segment of the delay. The node itself wakes up less frequently, so the message has to wait in the parent node for the node to wake up to receive it. For the more efficient wakeup patterns (ladder and two-ladders and crossed-ladders patterns) these two effects cancel out each other and the distribution of the delay remains exactly the same as in the single-parent case. For synchronized and even-odd patterns the delay is slightly improved[5]. See Table 1 for all the distributions.

We do not see this effect in the backward direction because in that case the messages go to the base station which is in fact waking up more frequently that the rest of the nodes in the network.

[5]Due to number of pages' limit we do not present the detail derivation of the expressions for the forward delay.

Table 1: Delay Distribution of Different Wakeup Patterns.

Wakeup Pattern		Minimum Delay	Maximum Delay	Average Value
Synchronized	$\mathbf{D}_{\triangleright}$	$(\frac{h-1}{g})T_{\text{eff}}$	$(\frac{g+h-1}{g})T_{\text{eff}}$	$(\frac{g+2h-2}{2g})T_{\text{eff}}$
	$\mathbf{D}_{\triangleleft}$	$(\frac{h-1}{g})T_{\text{eff}}$	$(\frac{h}{g})T_{\text{eff}}$	$(\frac{2h-1}{2g})T_{\text{eff}}$
Even-Odd	$\mathbf{D}_{\triangleright}$	$(\frac{h-1}{2g})T_{\text{eff}}$	$(\frac{2g+h-1}{2g})T_{\text{eff}}$	$(\frac{g+h-1}{2g})T_{\text{eff}}$
	$\mathbf{D}_{\triangleleft}$	$(\frac{h-1}{2g})T_{\text{eff}}$	$(\frac{h+1}{2g})T_{\text{eff}}$	$(\frac{h}{2g})T_{\text{eff}}$
Ladder Forward	$\mathbf{D}_{\triangleright}$	$(h-1)\tau$	$(h-1)\tau + T_{\text{eff}}$	$(h-1)\tau + (\frac{1}{2})T_{\text{eff}}$
	$\mathbf{D}_{\triangleleft}$	$(\frac{h-2}{g})T_{\text{eff}} - (h-3)\tau$	$(\frac{h-1}{g})T_{\text{eff}} - (h-3)\tau$	$(\frac{2h-3}{2g})T_{\text{eff}} - (h-3)\tau$
Ladder Backward	$\mathbf{D}_{\triangleright}$	$(h-2)T_{\text{eff}} - (h-3)\tau$	$(h-1)T_{\text{eff}} - (h-3)\tau$	$(h-\frac{3}{2})T_{\text{eff}} - (h-3)\tau$
	$\mathbf{D}_{\triangleleft}$	$(h-1)\tau$	$(h-1)\tau + (\frac{1}{g})T_{\text{eff}}$	$(h-1)\tau + (\frac{1}{2g})T_{\text{eff}}$
Two-Ladders	$\mathbf{D}_{\triangleright}$	$(h-1)\tau$	$2T_{\text{eff}} + (h-1)\tau$	$T_{\text{eff}} + (h-1)\tau$
	$\mathbf{D}_{\triangleleft}$	$(h-1)\tau$	$(\frac{2}{g})T_{\text{eff}} + (h-1)\tau$	$(\frac{1}{g})T_{\text{eff}} + (h-1)\tau$
Crossed-Ladders	$\mathbf{D}_{\triangleright}$	$(h-1)\tau$	$(\frac{2h-3}{h-1})T_{\text{eff}} + (h-1)\tau$	$(\frac{2h-3}{2h-2})T_{\text{eff}} + (h-1)\tau$
	$\mathbf{D}_{\triangleleft}$	$(h-1)\tau$	$(\frac{2h-3}{g(h-1)})T_{\text{eff}} + (h-1)\tau$	$(\frac{2h-3}{2g(h-1)})T_{\text{eff}} + (h-1)\tau$

5.3 Best Combination

The multi-parent idea can significantly reduce the backward delay but it has almost no effect on the forward delay, and it can be used with any of the wakeup patterns from the previous section. The question that arises is that which combination provides the best performance. If we start from the forward ladder pattern (as defined in Section 4.3) which can guarantee a good forward delay but is poor in terms of backward delay and then apply the multi-parent method to it, the resulting scheme can provide very short delays in both directions. We see in the following section that this pattern is the most efficient for a wide range of wakeup rates and system parameters. Figure 9 shows this scheme with two groups ($g = 2$) of red and blue nodes.

In the two example cases defined at the beginning of previous section, for the fixed-power case with $T_{\text{eff}} = 2s$, $h = 4$, and $g = 2$, we obtain the following distributions for forward and backward delays:

$$\left.\begin{array}{l}\mathbf{D}_{\triangleright} \sim \mathbf{U}\,[0.15, 2.15]\\ \mathbf{D}_{\triangleleft} \sim \mathbf{U}\,[1.95, 2.95]\end{array}\right\} \Rightarrow \;\max(\mathbf{D}_{\triangleright}, \mathbf{D}_{\triangleleft}) = 2.95s.$$

So the maximum delay is $2.95s$ which is significantly smaller than all the previous values including the best single-parent case delay of $3.48s$ for the crossed-ladders pattern.

For the fixed-delay case to achieve one second delay in both directions we need

$$T_{\text{eff}} = 700ms \Rightarrow P_{\text{wakeup}} = 1.43E_o,$$

which gives 64.8 months as the lifetime of the network. This is considerably longer than 47.2 months (about 1.5 years longer) for the previous best solution, the crossed-ladders pattern.

An additional advantage of the multi-parent technique which is actually a beneficial side-effect, is the increased robustness to node failure, *i.e.,* if one of the parents fails (*e.g.,* the node is momentarily blocked by an obstacle or there is interference in the channel or it is in a deep fading state), the message can still be sent through the other parents (but at the cost of additional delay), and the network remains connected. So the overall reliability of the network is also increased.

6. EVALUATION AND COMPARISON

In this section, we assemble all the numerical examples from previous sections and also present examples of delay-power tradeoff curves. These help us assess and compare different wakeup methods. We see that at low wakeup rates or when the application requires small delay bounds, the selection of an efficient wakeup scheme significantly impacts the performance. Furthermore, we consider the effect of number of groups on the overall performance of the wakeup methods.

Figure 10 and Figure 11 show the distribution of delay for different wakeup patterns for the first example scenario, the fixed-power case from Section 4 with $T_{\text{eff}} = 2s$, $h = 4$, and $\tau = 50ms$. Among the single-parent wakeup patterns ($g = 1$), crossed-ladders pattern achieves the smallest worst-case delay of $3.48s$. The multi-parent ($g = 2$) forward ladder pattern with worst-case delay of $2.95s$ achieves the best overall delay in both directions.

If the application requires a good backward delay, then clearly the backward ladder pattern is the optimal solution. For single-parent case ($g = 1$) backward ladder guarantees a delay of less than $2.15s$. By applying the multi-parent method the delay can be further reduced to $1.15s$, which is even smaller than the average wakeup period of $2s$.

Figure 12 shows the trade-off curves between the delay and power consumption for different wakeup schemes with $h = 4$ and $\tau = 50ms$. The horizontal axis shows the effective wakeup rate, $R_{\text{eff}} = 1/T_{\text{eff}}$ (number of wakeups per second) which from equation (3) is directly proportional to the wakeup process's power consumption. The vertical axis represents the worst-case delay in both directions. We get the following observations from this figure:

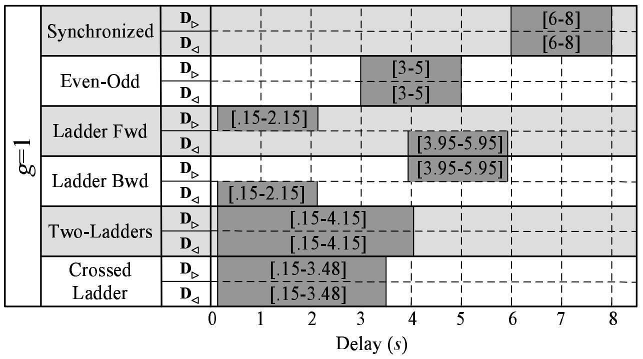


Figure 10: Delay Distribution (Single-parent Case): $T_{\text{eff}} = 2s$, $h = 4$, $\tau = 50ms$.

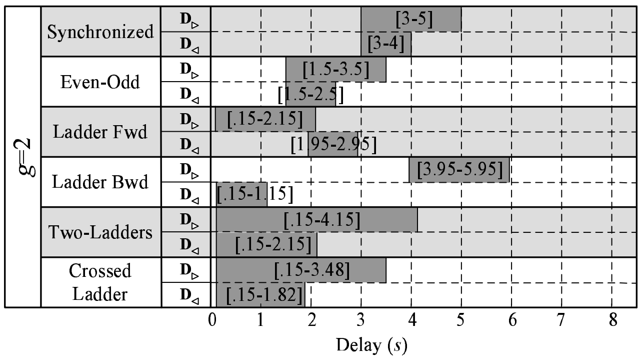


Figure 11: Delay Distribution (Multi-parent Case): $T_{\text{eff}} = 2s$, $h = 4$, $\tau = 50ms$.

(a) As expected, the tradeoff curves are decreasing, *i.e.,* at higher wakeup rates (more power consumption) better (shorter) delays can be guaranteed.

(b) A lower trade-off curve for a wakeup method indicates that it is more efficient as it provides a better delay at the same wakeup rate. We see that the multi-parent forward ladder pattern's trade-odd curve is below all other schemes which shows that this scheme is the most efficient over a wide range of wakeup rates[6].

(c) At low (slow) wakeup rates, the difference between the delays achieved by different wakeup schemes is large but as the wakeup rate increases (faster wakeups) the difference becomes smaller and all of the wakeup schemes provide almost the same delay. This shows that if the network can afford the energy cost of a fast wakeup rate, then the selection of the wakeup scheme is not crucial, but for energy-limited systems which operate at low wakeup rates, the selection of a good wakeup scheme can significantly improve the performance.

(d) The trade-off curves are closer at higher delay values and as the delay decreases the difference in the amount of power different schemes consume to guarantee the same delay becomes larger. For example for $3s$ delay, the difference between the power required by the best scheme (two-parent forward ladder with power $0.5E_o$) and the worst scheme (synchronized scheme with $1.4E_o$) is about $0.9E_o$. But to get $1s$ delay, the difference between the best and worst increases to $4E_o - 1.96E_o = 2.4E_o$; and for $0.5s$ delay the difference is much larger $8E_o - 3E_o = 5E_o$. This shows that as the delay requirement by the application becomes tighter, it is more beneficial and therefore more important to choose an efficient wakeup scheme.

[6]This statement is true and can be validated over a wider range than what is shown in this figure and with different number hops.

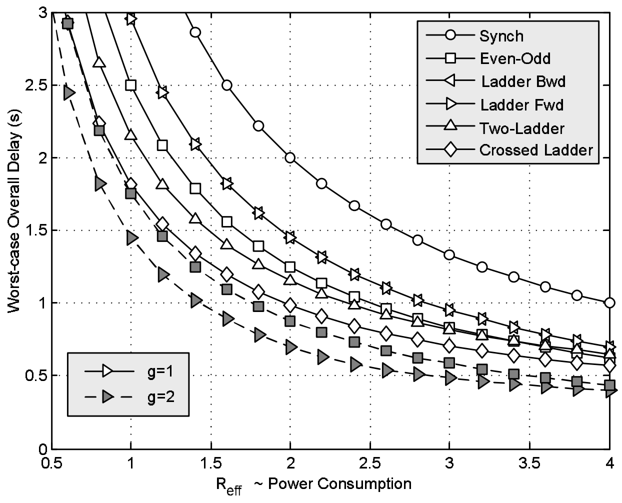


Figure 12: Delay-Power Tradeoff Curves ($h = 4$, $\tau = 50ms$).

Figure 13 shows the effect of number of groups, parameter g, on the delay for different wakeup patterns in a network for typical parameters: $R_{\text{eff}} = 2$, $h = 4$ and $\tau = 50ms$. Increasing the number of groups reduces the backward delay but the forward delay remains unchanged, so for large enough values of g, when backward delay becomes smaller than the forward delay, the overall worst case delay is bounded by the backward delay. This is the saturation point for the value of g beyond which there is almost no improvement. We observe from this figure that for many of the patterns such as two-ladders or crossed ladder, the saturation starts at even $g = 1$, but for forward ladder and even-odd and synchronized pattern we see a gradually declining improvement in the performance. Also note that, the main improvement is achieved from $g = 1$ to $g = 2$, and beyond that the reduction in delay is small.

7. PARENT ASSIGNMENT

In this section, we consider the problem of parent assignment. Our goal is to find a method to divide the nodes in the network into g groups (or color them with g different colors) such that every node has at least one parent from each group (or each color). Equivalently, the problem can be formulated as finding g edge-disjoint spanning trees with a fixed root in a given graph. An example where a graph is divided into two groups with two disjoint trees spanning all nodes in the network is shown in Figure 7.

In the appendix, we prove that this problem even for the case of $g = 2$ is NP-complete. This is shown by reducing 3SAT problem to an instance of the graph coloring problem. For larger values of g, unless the network is very dense, parti-

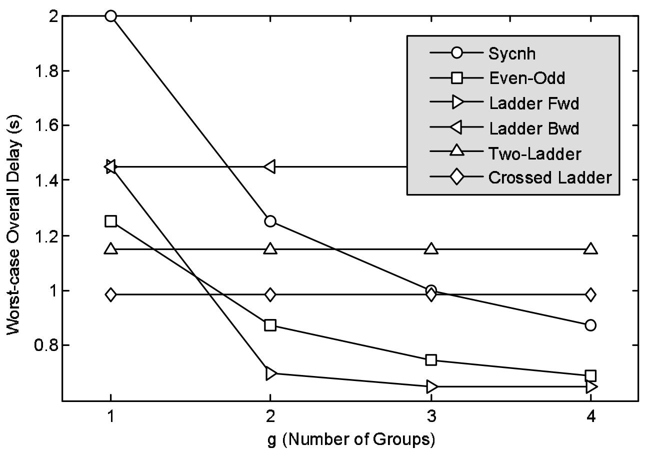


Figure 13: Effect of Number of Groups ($h = 4$, $\tau = 50ms$, $R_{\text{eff}} = 2$).

tioning the nodes into groups and assigning parents becomes either impossible or computationally very hard. Also, as discussed in previous section most of the benefit of using the multiple-parent technique is acquired by going from $g = 1$ to $g = 2$ and beyond that there is no significant improvement. Therefore, $g = 2$ is the most practical value which balances the initial complexity in the parent assignment process with the benefits we gain from using the multi-parent idea. In the rest of this section, we describe an efficient heuristic algorithm to find a valid coloring for a given network with $g = 2$. A centralized approach is used where the algorithm is executed on the base station or some other computationally powerful device which has information about the connectivity graph of the network.

7.1 Graph Coloring Algorithm

The algorithm colors the nodes in a given graph red or blue (or possibly purple) such that every node in the network gets at least one red parent and one blue parent, or instead a purple parent. Purple nodes have the functionality of both blue and red nodes, so they wake up in all frames and therefore they consume twice as much energy as other nodes. Ideally, the base station should be the only purple node in the network.

If the number of nodes in the network is small ($N < 20$), it may be possible to search over the entire space of possible colorings (which has 2^N points) to find a feasible solution. However, since the number of possible points grows exponentially with N, for larger networks this will be very time-consuming. So for large networks, we transform a relaxed version of the coloring problem into semi-definite programming (SDP) form and therefore find approximate solution very fast. Based on this, we propose a heuristic algorithm for graph coloring which consists of different phases. In each phase, a better approximate solution is found based on the results from the previous phases.

Layer Assignment: The initial step of the algorithm is to assign a *layer* to each node. Layer of a node is a tree-related parameter which shows the distance (in terms of number of hops) between the node and the base station on a tree. When there are multiple trees, the layer is defined as the largest distance over all trees[7]. Initially, the algorithm assigns the layer of each node the same as its level. Later in the fourth phase of the algorithm some of nodes may be moved to a higher layer.

1) First Approximation: After the initial layer assignment, each node finds the set of nodes in the lower layers to which it is connected. This set is called the *potential parents set* and is denoted by $\mathcal{P}_n$ for node n. We define a variable x_n associated with each node in the network which takes $+1$ value to indicate that the node is blue or -1 value to show that the node is red.

Instead of finding a coloring for the nodes that provides two differently colored parents for each node, we attempt to *balance* the number of red and blue parents for all nodes. Variable z_n is defined for each node as a measure of this balance:

$$z_n = \sum_{k \in \mathcal{P}_n} x_k. \tag{12}$$

If node n has almost the same number of blue and red parents, then z_n will be close to zero. Therefore, a relaxation of the graph coloring problem can be formulated as the following optimization problem where the weighted squared sum of variables z_n is minimized:

$$\begin{cases} \text{Minimize } \sum_n w_n {z_n}^2 = x^T P^T W P x, \\ \\ \text{subject to } \ x_n^2 = 1. \end{cases} \tag{13}$$

Here, w_n is the weight associated with node n, vector $x = (x_1, \ldots, x_N)^T$ is the optimization variable, the 'parent' matrix $P = [p_{ij}]$ is defined such that $p_{ij} = 1$ if $j \in \mathcal{P}_i$, otherwise $p_{ij} = 0$, and the matrix W has elements w_n on its main diagonal and zero elsewhere.

The problem is now reduced to a *two-way partitioning problem* [20]. A simple approximate solution to this problem through SDP relaxation can be found as follows[8]: Find the eigenvector corresponding to the smallest eigenvalue of matrix $P^T W P$, then apply the sign function to the eigenvector to get an approximate solution for x vector. Sign function assigns $+1$ to positive values and -1 to negative values. For zero we randomly choose either $+1$ or -1.

2) Soft Sign Function: Sign function quantizes the value to either $+1$ or -1. However, a larger absolute value in the eigenvector shows higher confidence in that the quantized value is correct. So we refine the solution from the previous phase according to the following procedure: First, sort the elements of the eigenvector according to their *absolute* value. Then, swap each bit in the solution ($+1$ to -1 and vice versa) from the smallest value to the largest and check for any improvement. If the solution with the swapped value is better, we keep it and check the next element. Otherwise, we keep the original value. Note that the number of swaps is linear with respect to the number of nodes and not exponential in N.

3) Adaptive Weighting: We minimize the *weighted* sum of z_ns and assign different weights to the nodes in the network. The rational behind this is to initially normalize the

[7]Note that 'layer' and 'level' of a node are two different parameters. Level of a node is related to the connectivity graph, but layer is assigned by our algorithm. Clearly layer of a node should be larger or equal to its level. For instance, a node in level 2 can be in layer 2 or 3 or more.

[8]For details please refer to [20] section 5.1.5, remark 5.1, and exercise 5.39.

effect of the number of parents and also have the ability to adaptively increase the cost induced by the nodes that do not get two differently colored parents. If a node has fewer parents, it is essential that it gets a balanced number of red and blue parents. For instance, the only acceptable value for z_n for a node with only two parents is zero, so a large weight is assigned to it. Initially we set the weight values as $w_n = \frac{1}{|\mathcal{P}_n|^2}$, where $|\mathcal{P}_n|$ shows the number of potential parents of node n. After finding an approximate solution, the algorithm checks the solution. If it satisfies all the conditions and all the nodes have one red and one blue parents, the algorithm stops and returns the solution. If the solution is not valid, the algorithm increases the weight of the nodes whose parents are all of the same color, and recomputes the solution by finding the eigenvector of the new matrix. In this way, the algorithm adaptively assigns larger weights to the nodes that are causing problem and have more strict requirements.

4) Layer Re-assignment: If after some pre-specified number of iterations no solution is found, the layer of the nodes causing the problem can be increased. The weight of such nodes is generally large, so it is easy to identify them. By assigning a higher layer to a node, the set of its potential parents grows. For example, a node in layer 3 can have layer 2 nodes as its parents, but if this node moves to layer 4, it can have its layer 2 and also its layer 3 neighbors as potential parents. After layer-reassignment the whole algorithm is repeated again with the new layer values. Increasing the layer of a node can possibly affect the layer of its children and descendant nodes in the graph, and therefore may also increase the maximum number of layers (parameter h) in the network. To avoid this problem we limit the maximum number of layers to a pre-specified value. In our implementation, the layer promotions were allowed only if after layer re-assignment the maximum number of layers remained the same as before.

7.2 Simulation Results

To evaluate this heuristic algorithm, we implemented it in both Matlab and C++. We ran our algorithm on many instances of randomly generated graphs. Random geometric graphs [23] were used to model the connectivity graph, and in the generated graphs all nodes had at least two neighbors. Table 2 shows the performance of the different phases of algorithm. The first row shows the average number of neighbors of the nodes in the network. The second row shows the average number of parents per node. The first level nodes are excluded in computing the average number of parents. The values in the third to sixth rows of the table show the percentage of graphs for which a valid coloring could be found. Even if one node did not get two differently colored parents, the coloring was considered invalid. The algorithm ran for 50 iterations of adaptive weight change between layer reassignments. Clearly for a denser graph, finding a solution becomes easier. As it can be seen from the simulation results, with adaptive weighting and layer re-assignment almost all graphs can be colored.

Although it is unlikely, it is still possible that no valid coloring can be found for a particular graph. For instance, if the graph has an *orphan node* which has only one potential parent, no coloring is valid. In such case, two solutions are possible: (a) either color the parent of the orphan node purple which therefore increases the power consumed by the parent node, or (b) accept more delay for the messages sent by or received by the orphan node.

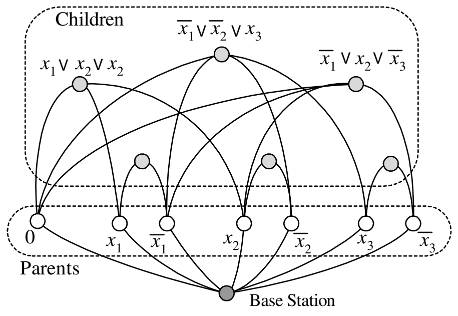


Figure 14: Reduction of 3SAT problem to the graph coloring problem.

8. CONCLUSION

In this paper, we analyzed different wakeup schemes and obtained their delay distribution and delay-power tradeoff curves. All existing wakeup patterns such as synchronized (SMAC) and staggered patterns (DMAC) are considered and we also introduced new efficient wakeup patterns such as crossed-ladders pattern which outperforms other methods. We also presented the new cross-layer idea, called multi-parent technique, where by assigning multiple parents with different wakeup schedules to each node in the network, significant performance improvement is achieved. A heuristic algorithm for parent assignment problem for multi-parent method was presented and evaluated through simulation.

APPENDIX

In this section, we prove that the coloring problem defined in Section 7 is NP-complete by presenting a polynomial time reduction from 3SAT to this problem. 3SAT is a special case of satisfiability problem where the Boolean formulas are in a special form consisting of k clauses connected by $\wedge$, each having three literals: $\phi = (a_1 \vee b_1 \vee c_1) \wedge \ldots \wedge (a_k \vee b_k \vee c_k)$. Each literal is a Boolean variable x_i or a negated Boolean variable $\overline{x_i}$ $(i = 1, \ldots, n)$. We would like to know if some assignment of true and false to the variables makes the formula evaluate to true.

We construct an instance of the graph coloring problem for a Boolean formula $\phi(x_1, \ldots, x_n)$ as follows: There are two set of nodes in the graph, a *parent set* which has $2n+1$ nodes and a *children set* which has $n+k$ nodes. For each variable x_i, we create two nodes in the parent set, one for the variable and one for its negated version, and one node in the children set which is connected to these two parents. Also there is a common parent node representing the 0 (false) value. For each clause $(a_j \vee b_j \vee c_j)$ we have a node in the children set which is connected to the parents corresponding to a_j, b_j, c_j, and also to the common parent. Figure 14 shows an example, where $\phi = (x_1 \vee x_2 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3})$. To demonstrate that this construction works, we show that a valid coloring for the constructed graph gives a satisfiable assignment for ϕ and vice versa.

Suppose ϕ has a satisfiable assignment. We use color 'red' for false and 'blue' for true and color the parent nodes according to the satisfiable assignment for ϕ. We show that

Table 2: Simulation Result for Graph Coloring Algorithm ($N = 100$).

Average Number of Neighbors		6.7	8.2	9.8	12.3	15.1	18.2	21.6
Average Number of Parents		2.50	2.99	3.40	4.13	4.89	5.79	6.83
Success Percentage	First Approximation	9.19%	16.92%	27.12%	39.17%	41.57%	57.83%	67.96%
	Soft Sign Function	30.09%	58.84%	75.28%	79.95%	83.75%	85.57%	92.18%
	Adaptive Weighting	81.13%	98.22%	99.38%	99.71%	99.78%	99.91%	99.93%
	Layer Re-assignment	91.44%	99.24%	99.65%	99.89%	99.96%	99.98%	100.00%

this results in a valid coloring and all children are connected to both red and blue nodes. The children corresponding to the variables are connected to both x_i and $\overline{x_i}$, so they get red and blue parents. The other children are related to the clauses in ϕ. Since ϕ is true, at least one of the literals in each clause is true and therefore the child node related to each clause has at least one blue parent. Since each child is also connected to the common 0 parent node which is red, it gets both red and blue parents.

For reverse, if we find a coloring for the nodes, we can convert it into an assignment for variables. We show that this assignment makes ϕ evaluate to true. First note that x_i and $\overline{x_i}$ are forced to select different colors as they have one common child node. Assume the common 0 parent is colored red. Then we use red for false and blue for true and form an assignment for x_i variables. Each clause in ϕ has a child node in the graph and the child node has at least one blue parent so the clause has at least one true literal and therefore it is true. So under this assignment, all the clauses are true and ϕ evaluates to 1.

A. ADDITIONAL AUTHORS

Krishna Chitalapudi, Robert Bosch Corporation, Research and Technology Center (RTC),
email: `krishna.chintalapudi@rtc.bosch.com`,
Dhananjay Lal, Robert Bosch Corporation, RTC,
email: `dhananjay.lal@rtc.bosch.com`,
and Bhaskar Srinivasan, Robert Bosch Corporation, RTC,
email: `bhaskar.srinivasan@rtc.bosch.com`.

B. REFERENCES

[1] W. Ye, J. Heidmann, and D. Estrin, “An Energy Effecient MAC protocol for Wireless Sensor Networks,” in *IEEE Infocom*, 2002.

[2] W.Ye, J. Heidmann, and D. Estrin, “Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks,” *ACM/IEEE Transation on Networking*, Vol. 12, No. 3, pp. 493-506, June 2004.

[3] T. van Dam and K. Langendoen, “An Adaptive Energy-efficient MAC Protocol for Wireless Sensor Networks,” *in Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys ’03)*, 2003, pp. 171-180.

[4] G. Lu, B. Krishnamachari and C. Raghavendra, “ An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks,”, *IEEE Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04)*.

[5] Q. Cao, T. F. Abdelzaher, T. He, and J. A. Stankovic,“Towards Optimal Sleep Scheduling in Sensor Networks for Rare Event Detection,” *In the fourth international conference on Information Processing in Sensor Networks (IPSN)*, April 2005.

[6] Y. Li, W. Ye, and J. Heidemann, “Energy and Latency Control in Low Duty Cycle MAC Protocols,” *In Proceedings of the IEEE Wireless Communications and Networking Conference*, New Orleans, LA, USA, March, 2005.

[7] G. Lu, N. Sadagopan, B. Krishnamachari, and A. Goel, “Delay Efficient Sleep Scheduling in Wireless Sensor Networks,” *IEEE Infocom*, March 2005.

[8] T. Armstrong, “Wake-up Based Power Management in Multi-hop Wireless Networks,”, *Term Survey Paper for ECE1717 Quality of Service Provisioning in Mobile Networks course*, available at url: *http://www.eecg.toronto.edu/ trevor/Wakeup/index.html*

[9] R. Zheng, J. C. Hou, and L. Sha, “Asynchronous Wakeup for Ad hoc Networks,” *in Proceedings of the 4th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing*, 2003, pp. 35-45.

[10] Y. Tseng, Ch. Hsu, and T. Hsieh, “Power-saving Protocols for IEEE 802.11-based Multi-hop Ad hoc Networks,” *in Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Com- munications Societies (INFOCOM 2002)*, 2002.

[11] J. Polastre, J. Hill, and D. Culler, “Versatile low power media access for wireless sensor networks,” *in 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, Baltimore, MD, USA, November 2004.

[12] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. B. Srivastava, “Topology management for sensor networks: exploiting latency and density,” *in Proceedings of the 3rd ACM Interational Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, 2002, pp. 135-145.

[13] M. J. Miller and N. H. Vaidya, “Power save mechanisms for multi-hop wireless networks,” *in Proceedings of 11 the 1st International Conference on Broadband Networks*, 2004, pp. 518-526.

[14] L. Gu and J. A. Stankovic, “Radio-Triggered Wake-Up Capability for Sensor Networks,” *10th IEEE Real-Time and Embeded Tehcnology and Application Symposium (RTAS’04), Toronto, May 2004.*

[15] J. Elson, L. Girod, and D. Estrin, “Fine-Grained Network Time Synchronization using Reference Broadcasts,” *In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA., December 2002.

[16] S. Ganeriwal and R. Kumar and M. Srivastava, “Timing-sync Protocol for Sensor Networks,” *ACM SenSys*, Los Angeles, CA, November 2003.

[17] A. Keshavarzian, E. Uysal, A. Menjashwar, and F. Hermann, “Energy-Efficient Link Assessment in Wireless Sensor Networks,” *In Proceedings of IEEE Infocom*, 2004.

[18] S. Vasudevan, J. Kurose, and D. Towsley, “On Neighbor Discovery in Wireless Networks with Directional Antennas,” *In Proceedings of IEEE INFOCOM*, Miami, FL, March 2005.

[19] Chipcon/TI, CC1100 Datasheet, *http://www.chipcon.com*

[20] S. Boyd and L. Vandenberghe, Convex Optimization, *Cambridge University Press*, 2004.

[21] A. Papoulis, Probability and Statistics, *Prentice Hall*, 1990.

[22] M. Sipser, Introduction to the Theory of Computation, *PWS Publishing Company*, 1997

[23] M. Penrose, Random Geometric Graphs, *Oxford Studies in Probability*, May 2003.