Sleeping Beauty: Efficient Communication for Node Scheduling

Chayan Sarkar*, R. Venkatesha Prasad*, Raj Thilak Rajan*†, Koen Langendoen*
*Delft University of Technology, The Netherlands
†IMEC, Holst Center, Eindhoven, The Netherlands
Email: {c.sarkar, r.r.venkateshaprasad, r.t.rajan, k.g.langendoen}@tudelft.nl

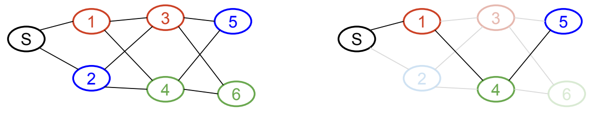
***Abstract*—Typical Wireless Sensor Networks (WSN) deployments use more nodes than needed to accurately sense the phenomena of interest. This redundancy can be leveraged by switching-on only a subset of nodes at any time instant (node-scheduling) and putting the remaining nodes sleep. This effectively extends the network lifetime. In addition to sensing coverage, node-scheduling schemes must also ensure that (i) the network stays connected, and (ii) the time needed to wake-up the complete protocol stack after sleeping is minimized. We present *Sleeping Beauty*, a highly-efficient data collection protocol that aids node-scheduling schemes in both aspects.**

Sleeping Beauty uses a slotted and tightly synchronized communication primitive, where a node keeps its radio off for most of the time, except in the slots when it needs to participate for successful communication. Further, an efficient neighbor-discovery mechanism is included that provides partial, but sufficient topology information (potential parents) to avoid network partitions. Furthermore, Sleeping Beauty employs a novel, yet simple clock-offset estimation technique that maintains highly-accurate time synchronization over long radio-off periods (i.e., less than 500 μs deviation even after 45 min of sleeping). This minimizes time wasted in resynchronizing the network in between data collection rounds. Through experiments on two different testbeds, we verified that Sleeping Beauty decreases the duty cycle up to a factor of 3 compared to state-of-the-art techniques, while achieving similar delivery ratios.

I. Introduction

Even after more than a decade of research, energy-efficient communication –and data collection in particular– is still a *holy grail* within the community. Part of the challenge is that typical WSN deployments include more nodes than needed to accurately sense the phenomena of interest. This redundancy safe guards, on the one hand, against failing nodes and links, but on the other hand leads to inefficient use of energy. *Node-scheduling* schemes address the latter aspect by keeping only a (minimal) subset of nodes active at any instant. They do so by selecting a representative from each so-called *common sensing group*. A common sensing group can be formed when (i) a common target or area is covered by a set of nodes [1], [2], or (ii) the data generated by the nodes show high degree of correlation such that sensed data for multiple nodes can be reconstructed using data from a single node [3], [4]. In either case, representative node(s) from each group needs to be active during the data collection round. Based on the application scenario, only one or multiple representative (such as k-coverage) nodes are selected active. By alternating duties across rounds, nodes save energy and extend the network lifetime.

As an example, Fig. 1a shows a network of six nodes, where only a subset of active nodes is sufficient under the common sensing group regime (indicated by color). Fig. 1b shows one such combination. However, if nodes {3, 4, 5} are selected instead of {1, 4, 5}, all groups are still represented, but the network would be disconnected from the sink making it impossible for the application to receive the sensed data.



(a) (b)

Fig. 1: (a) A deployment with multiple correlated sensor groups (denoted by colors), and (b) coverage of the entire monitoring region by a subset of connected active nodes.

Given such common sensing groups and network topology of a WSN, the selection of a minimal set of active nodes that guarantees both (a) the connectivity of active nodes to the sink, and (b) the representation of each group by active node(s), is shown to be NP-hard [5], [6]. Thus a number of heuristic algorithms have been proposed in the literature [7], [8] to select close to optimal sets of active nodes satisfying the above constraints. After selecting the active nodes for a round, usually by the centralized sink node, the sensed data from these nodes still need to be collected *efficiently*. That is the main challenge we address in this paper.

A. Motivation

Based on the application scenario, a different node scheduling strategy has to be devised, e.g., k-coverage [7], point-coverage [5], spatial correlation [3], etc. Node-scheduling algorithms, which select a subset of active nodes, follow two general strategies. Either they select the set of active nodes at the application layer and leave the routing to the underlying protocols [7], [5], or they jointly perform node selection and routing [1]. In the first case, an advanced routing protocol is required that should use a minimal number of relay nodes besides the chosen active nodes in order not to diminish the overall efficiency. In the second case, the cross-layer solution is tightly coupled to the particular scheduling scheme making it non-reusable for other scheduling strategies. Thus there is a need for a generic data-collection algorithm that ensures highly-efficient network operation irrespective of the node-scheduling policy.

B. Solution approach and challenges

In this work we present Sleeping Beauty, an energy-efficient communication protocol for node-scheduling scenarios. It uses a slotted and tightly synchronous operation among the nodes, and deploys a pull-based mechanism to collect data from every active node when the sink requires it (based on application requirements). This ensures a highly compact radio-on time of the nodes and increases radio-off time in-between two sensing rounds. The efficient operation of Sleeping Beauty is rooted in the fast-flooding mechanism provided by Glossy [9]. Like LWB [10], which is also based on Glossy, Sleeping Beauty contains a central part controlling what happens in every slot. In order to be flexible and efficient, the control mechanism runs at the sink and has been designed to address the following challenges. First, as the inefficiency of multi-hop communication in WSN caused by retransmissions and per-hop delays, repercussion of these factors need to be restrained. Moreover, in every communication slot, only those nodes that provide routing progress need to be involved while maintaining minimal retransmission and delay. Second, as the network topology varies over time (e.g., due to link quality fluctuations), the sink should account for this and ensure that a valid path exists from any active node to the sink at all times. The challenge is to provide neighbor discovery at minimal cost. Third, since the efficiency of Sleeping Beauty stems from time-triggered operations, a tight synchronization among the nodes needs to be maintained, even when nodes go to sleep for long duration. If not, what was gained in node-scheduling will be lost in trying to synchronize again. In particular, standard node-scheduling schemes assume that once nodes wake-up they can instantly join the network. In reality, however, waking up the complete protocol stack can be time consuming. Thus, zero/minimal overhead should be ensured when a node rejoins the network after long periods of radio inactivity.

C. *Contributions*

Sleeping Beauty efficiently tackles the challenges mentioned above. Specifically, we improve state-of-the-art with these contributions:

1) Efficient data collection for node scheduling: We provide the design and evaluation of a communication protocol that combines fast flooding and on-off scheduling to support scenarios where only a small, representative set of nodes need to report their data to the sink. We do so by separating the concerns of node selection and data collection. In that respect Sleeping Beauty is an efficient, slot-based communication substrate supporting different node selection schemes.

2) Efficient neighbor discovery: Neighborhood discovery is a required service for many WSN applications. We propose a synchronized strobing mechanism that helps every node to learn about its neighborhood efficiently, that is, with minimal radio activity. This is achieved by having nodes focus only on potential parents and update information instead of discovering it every time from scratch. This aids node-scheduling algorithms in coping with network connectivity and dynamics.

3) Accurate and long-lasting time synchronization: Much of Sleeping Beauty's efficiency stems from the fact that operations are tightly synchronized. To counter the drifts of the hardware clocks in nodes, Glossy (who pioneered this style of communication) utilizes frequent network-wide floods, taking up a significant amount of energy. To avoid this overhead we propose a simple clock estimation algorithm with which nodes – even after 45 min – are still synchronized within an error bound of 500 μs.

We substantiate our claims by evaluating the performance of Sleeping Beauty on two public testbeds –Indriya [11] and FlockLab [12]– and on a local setup. We compare against two Glossy-based communication protocols LWB [10] and Forwarder Selection [13], showing overall reductions in energy consumption.

II. Background

In this section, we review state-of-the-art data collection protocols with respect to their limitations for node scheduling. The Collection Tree Protocol (CTP) [14] is a robust data collection protocol that builds and maintains a spanning tree. It provides a good packet delivery ratio for many deployments, but breaks down under increased traffic and network dynamics. To address the latter limitation, Landsiedel *et al.* [15] proposed an opportunistic data collection protocol, called ORW, which routes packets to the first available node from a set of possible forwarders. Although ORW outperforms CTP in many aspects, it shares the overhead of maintaining routing metrics, which compromises efficiency, especially when sets of nodes are switched on and off.

Recently, a number of synchronous all-to-all communication protocols have emerged as highly efficient as that of the traditional tree-based protocols [9], [10], [16]. Most of these protocols utilize Glossy's fast flooding mechanism, which we will describe in quite some detail as Sleeping Beauty is also based on it. In Glossy [9], a special node, called the initiator (sink node), starts the flooding in a time-triggered fashion. The overall flooding time is very small compared to traditional flooding [17], as Glossy embraces a phenomenon called constructive interference (CI). When more than one node transmits the same content exactly at the same time (within a tolerable time difference, usually half of a symbol period), the signals constructively interfere at the receiver. As a result, the receiver can successfully decode the symbols. In Glossy, each node turns on its radio just before the start of a new flood, and after the initiator sends the packet, all its first-hop neighbors receive the packet. Then they forward the packet immediately (with the same processing and switching delay), which causes constructive interference at the second-hop nodes from the initiator. The second-hop nodes then forward the packet immediately too. The process continues as a ripple effect and the whole network is flooded within a couple of milliseconds. Nodes turn off their radio immediately after the flooding. Note that as a result of participating in the flood, the nodes implicitly become synchronized with the initiator the moment they receive the packet.

The Low-power Wireless Bus (LWB) protocol [10] has turned the Glossy floods into a generic all-to-all communication primitive by adding a centralized component that creates a global schedule specifying who may initiate a flood (i.e., send data) in a particular slot. By having all nodes participate in every flood, LWB is completely topology agnostic and voids the need for obtaining/maintaining routing information, allowing it to function as a robust and efficient communication substrate. For data collection, however, it is not necessary to relay the messages at all nodes, providing room for optimization. For example, Carlson *et al.* [13] describe a *forwarder* selection mechanism for LWB that is quite effective in only involving a small set of nodes between source and sink. For node scheduling, however, this solution is inadequate as (i) nodes cannot go to sleep for extended periods of time without losing synchronization, and (ii) the schedule reserves slots for the inactive nodes, causing all nodes to waste energy by idling. These considerations prompted us to design a new protocol.

III. A first look at Sleeping Beauty

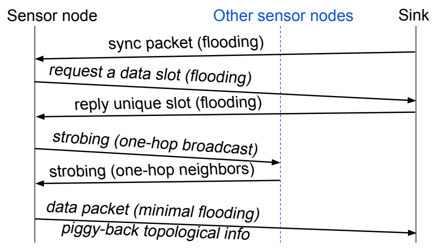
We begin our description of Sleeping Beauty with a conceptual overview depicted in Fig. 2. The sink coordinates all actions by periodically flooding the network with sync packets. Once a node has received such a sync packet it may join the network by requesting its own slot in the global schedule. It can then start reporting its sensed data to the sink every inter-packet interval (I). However, it may only do so if it is one of the active nodes determined by the sink every scheduling interval (S). The sink disseminates this set and an accompanying schedule along with the sync packets. The selection of the connected set of active nodes is based on basic one-hop neighbor information (possible parents) collected by the nodes through localized strobing. Unlike the sync slot, where all nodes participate in the flooding, only the set of active nodes participate in the flooding during the data slots. This minimal flooding ensures higher energy efficiency as compared to other Glossy-based protocols.

A. Design goals

The overarching objective is to enable inactive nodes to sleep for prolonged periods of time, while utilizing Glossy's fast and efficient flooding primitive. This amounts to the following two design goals. **First**, we want an efficient neighbor discovery mechanism to muster partial, but sufficient topological information. In particular, we are interested in providing lists of potential parents to the active node selection algorithm running at the sink so that any node-scheduling scheme can be applied on the network. **Second**, we desire a node synchronization scheme that allows nodes to be offline for extensive periods of time while maintaining the desired level of tight synchronization. That is to say that the requirement of frequently sending synchronization packets by a Glossy-based protocol is to be replaced by a method that runs at much larger (and even irregular) intervals.

B. Architecture

The operation of Sleeping Beauty is divided into two major phases. In the first phase, called *bootstrapping*, nodes join the network by synchronizing with the sink and requesting for data slots. During this phase nodes also estimate their own


Sensor node
Other sensor nodes
Sink
sync packet (flooding)
request a data slot (flooding)
reply unique slot (flooding)
strobing (one-hop broadcast)
strobing (one-hop neighbors)
data packet (minimal flooding)
piggy-back topological info
Fig. 2: Conceptual overview of Sleeping Beauty.

clock drift parameters, and survey the local neighborhood to identify high-quality links with nodes that can provide routing progress to the sink. This topological information is relayed to the sink by piggybacking with the data packet. In the second phase, called *steady-state*, the sink selects a new set of active nodes in every scheduling period. Once the selection is made and communicated to all nodes in the network, only the selected will stay active until the next scheduling round. Once the round finishes, all nodes wake-up again and topological information is refreshed to accommodate for any changes in network conditions, before the sink selects the next set of active nodes.

To achieve efficient data communication, Sleeping Beauty requires nodes to perform various secondary communication activities, e.g., neighbor discovery, offset estimation, etc., as outlined in Section V. These secondary activities can lead to significantly-higher energy consumption by the nodes if they are not done efficiently. Therefore, the sink defines a superframe that reduces the average duty cycle by carefully managing the radio on time of all nodes.

C. Superframe as a building block

To cater to various types of activities, a superframe in Sleeping Beauty may consists of four different types of communication slots, as shown in Fig. 3. A slot is of sufficient length to flood a packet from an initiator node (who starts the flooding within this slot) to all other nodes, including time for processing the contents at the receiver(s). The total duration of a superframe is equal to 1 s and I s during the bootstrapping and steady state, respectively.

1) ***Synchronization slot:*** This is the most important slot in a superframe, and it functions as the header of the superframe. This slot is used for (re)synchronizing the nodes with the sink. Thus only the sink can initiate transmission of a sync packet, and there is only one sync slot in a superframe. Though synchronization is performed based on the reception time of the sync packet, the content of this packet specifies the lengths of the other slots (avoiding unnecessarily fixing a large slot time). Additionally, a sync packet also disseminates the list of active nodes that are selected based on a node scheduling policy. How a sync packet is processed to decide the active set of nodes is discussed in Section V-A.

2) ***Request/Reply (RR) slots:*** Every node needs to acquire a data slot to send its data to the sink. RR slots are used to request a data slot (odd numbered RR slot) and subsequent

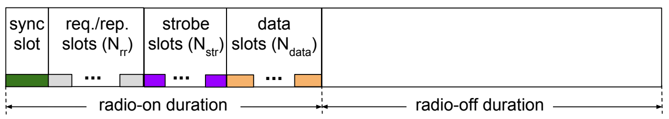


Fig. 3: Superframe (SF) structure that contains a radio-on and radio-off duration. Note that not all parts depicted need to be present (cf. Fig. 4).

granting (even numbered RR slot) If two nodes send requests in the same slot, most of the times the sink is able to receive a single request message successfully either due to the capture effect [18] or one of the contending nodes is at least one hop closer to the sink compared to the others.

*3) **Strobe slots**:* These slots are used to notify the presence of a node to its neighboring nodes as described in Section IV-B. The number of strobe slots is equal to the total number of nodes in the network to rule out collisions.

*4) **Data slots**:* Every node is assigned a unique data slot to deliver its data, which it obtained using the RR slot. Additionally, a list of potential parent nodes that were discovered using the strobe packets is also piggybacked in the data packet. This provides a partial, but sufficient topological view of the network to the sink.

IV. Bootstrapping

During bootstrapping nodes perform various learning activities, which are utilized for operational optimization and duty cycle reduction in the long run. Even during bootstrapping, nodes start reducing radio-on time as soon as they complete the required level of learning. Moreover, the duration of bootstrapping is very insignificant (only a couple of minutes) compared to that of the steady state, which can last for months if not years. The following three major tasks are performed during bootstrapping.

A. Node joining

As every communication is time triggered and synchronous, every node needs to synchronize itself with respect to the sink before starting any transmission. Thus, after a node is powered on, it keeps listening for sync packets. As soon as the node receives such a packet, it synchronizes with the network and learns about the superframe structure. As mentioned earlier, a node requests for a data slot and gets a reply from the sink in the odd and even RR slots, respectively. Once a node has obtained a slot, it stops sending any further requests, and completes its joining procedure. However, it keeps participating in all other RR slots to help with delivering the request/reply messages to the intended recipients.

It is clear that if the number of RR slots is low and the number of requesting nodes is large, then it will take a long time before every node gets a dedicated data slot. On the other hand, if there are more RR slots, more nodes can get their request granted within a short period of time. However, this implies that nodes will be wasting a significant amount of energy during the RR slots. Sleeping Beauty uses a dynamic approach to decide the number of RR slots.

***Deciding the number of Request/Reply slots**:* At the beginning, Sleeping Beauty starts with 48 RR slots (the maximum that fits within one second). Thus, within a second a maximum of 24 nodes can be allotted a data slot. When most of the nodes are allotted a slot, many unused RR slots cause unnecessary energy consumption by the nodes. Sleeping Beauty notes the number of used RR slots within the current superframe, and uses this information to adjust the number of RR slots in the next superframe. After some time, the number of RR slots is reduced to 2. By that time, most of the nodes, if not all, would have successfully acquired a data slot. The sink then decides to move to the steady state (see Section V).

B. Building a partial view of the network

To gather a partial, but sufficient topological view of the network at the sink, nodes piggyback a list of potential parents with the sensed data. The ETX metric is used to identify these potential parents from the beacons transmitted in the strobe slots. To avoid any collisions among nodes, a separate strobe slot is allocated to each node as follows. When a node is assigned to transmit in the t^{th} data slot, it shall use the $(t+1)^{th}$ strobe slot to send its strobes (the first strobe slot is reserved for the sink). In its slot, a node sends a fixed number of consecutive strobe packets, which contain the current ETX value of the transmitter. A neighboring node estimates the link quality between the sender and itself based on the number of received strobe packets and updates its own ETX value. Based on the ETX values of all neighbors, a receiver node compiles a list of potential parent nodes. Unlike CTP, where a node chooses only one parent exclusively based on the best (minimum) ETX value, Sleeping Beauty maintains a list of parents. If a long list of parents would be reported with the data packets, a significant amount of energy would be spent by the nodes for delivering larger payloads. Therefore, only a small list of parent nodes with better ETX values is reported to the sink. This way the sink can gather partial, but sufficient topological information about the network. From our empirical evaluation, see Section VII, we inferred that a list of 5 nodes is sufficient for 97% cases.

C. Clock-offset estimation

Due to unstable clocks of the small embedded devices involved, a node can experience a significantly higher clock offset with respect to the reference node (i.e. the sink) within a small period. To rectify such clock offsets, the common method is to send synchronization packets more often and individual nodes should put in extra efforts, such as keeping the radio on for a longer duration to receive the synchronization packets. This induces significantly higher energy consumption by the nodes. In Sleeping Beauty, during bootstrapping, frequent sync packets are sent so that sufficient data points can be collected to estimate the clock offset. Once in steady state the clock offset is rectified by the node itself without receiving sync packets as will be described in Section VI.

During bootstrapping, every node in the network remains active and senses data every inter-packet-interval (I s) as de-

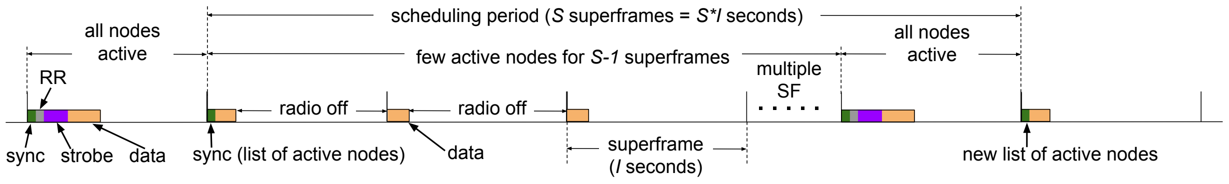


Fig. 4: Periodic strobe slots and new set of active node dissemination using various superframe structures in the steady state.

termined by the application. Therefore the superframe contains as many data slots as the number of nodes in the network. The bootstrapping process ends T s after the last node joins the network. The timeout T should be set such that the sink has sufficient time to acquire the topological view of the network and that the regular nodes can gather enough data points to estimate their clock offset.

V. Steady-state operation

The beginning of the steady state is marked by selecting a set of active nodes. The sink disseminates the list of active nodes using the sync packet. It selects a new set of active nodes every scheduling period, which amounts to $S{*}I$ s (Fig. 4). Note that the protocol is flexible enough to cope with any value of S and I as sought by the application.

A. Periodic active node selection

Since active node selection is outside the scope of this article, we used an existing node scheduling algorithm on top of Sleeping Beauty to demonstrate how it can be integrated with such methods. The common sensing groups are assumed to be known or can be formed at runtime based on the sensed data. Then the active nodes are selected by means of the algorithm described in [6]. The list of active nodes is disseminated using a bitmap in the sync packet. If, upon receiving a sync packet, a node finds a zero at the bit position of its data slot, it goes to sleep (dormant node). The active nodes continue to wake-up every I s to send their data without requiring any sync packet (Fig. 4). Note that active nodes keep their radio on in their assigned data slot, to transmit their own data, as well as in the data slots associated with the other active nodes, to forward the data from them.

After S superframes, the sink reconsiders the selection of the active nodes. Since the quality of the links may have changed during this long time period ($S * I$ s), the last superframe of the series includes strobing slots to reassess the links (see Fig. 4). To obtain a complete picture of the (changed) network topology all nodes participate in the link quality assessment procedure. Hence, non-active nodes can sleep for only S-1 superframes.

B. Updating the list of parents

During bootstrapping, every node listens to every strobe slot to find its neighbors, and compute its ETX to the sink. The same procedure could be followed in the steady state as well, but would waste a lot of energy in the case of large multi-hop networks that comprise many more nodes than neighbors. Therefore, a node records any neighbor seen during bootstrapping in a bitmap. This bitmap is then used to selectively listen to the strobing slots of potential parents during steady state. This 'smart strobing' optimization significantly reduces the energy consumed for link quality assessment, see Fig. 10a for details. To ensure a high Packet Reception Rate (PRR), nodes sort the list of parents based on ETX and picks the top 5 (lowest ETX) as the potential parents (cf. Fig. 10b).

C. Clock-offset correction

Active nodes wake up in every superframe, allowing them to stay synchronized easily. Dormant nodes, in contrast, run the risk of waking up out of phase due to drifts in their clocks when rejoining the network for the link quality assessment after $(S-1)I$ s. The drift could be countered by waking up early, but that would waste energy, so Sleeping Beauty adopts a correction procedure based on estimating the clock-offset parameters, as discussed in next.

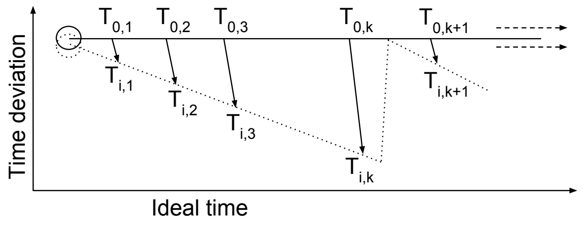


Fig. 5: Aperiodic one-way communication between sink (solid line) and sensor node (dotted line), and the effect of iterative clock correction at the sensor node at the k^{th} time instant [19].

VI. Low-cost clock-offset estimation

We consider the scenario shown in Fig. 5, where the sink (solid line) transmits packets to the sensor nodes. For the sake of simplicity, we illustrate a single-hop communication link. However, in the case of multi-hop scenarios, given the known additive transmission delay between the hops, a similar illustration holds. Note that time deviation may either be positive or negative with respect to the sink.

At the k^{th} transmission, the sink node transmits the reference time $T_{0,k}$, which is received by node i at its local time $T_{i,k}$. The sink node transmits a packet to the local node i at every δT s for a total time duration of ΔT, during which K transmissions occur. It is worth noting that the communication between the nodes is not necessarily at a fixed interval, which allows us to dynamically vary the duty cycle of communication δ between the nodes as per our requirement.

A. Least squares estimator

All the clocks are inherently non-linear, however given sufficiently low *Allan* deviation for a short period of time, the clock model could be linearized to fit a first-order model [20]. Under the assumption that the sink node is our reference node, the corresponding local time at node i is,

$$T_{i,k} = (1 + \dot{\phi}_i) T_{0,k} + \phi_i, \tag{1}$$

where $\{\dot{\phi}_i, \phi_i\}$ are the frequency offset and phase offset of node i [21]. Under ideal conditions, $\{\dot{\phi}_i, \phi_i\} = \{0, 0\}$ and subsequently $T_{i,k} = T_{0,k}$, however in practice these clock errors are prevalent and the challenge is to estimate and correct the local clock at node i appropriately. Furthermore, the time deviation at node i with respect to the reference clock is,

$$\epsilon_{i,k} \triangleq T_{i,k} - T_{0,k} = \dot{\phi}_i T_{0,k} + \phi_i. \tag{2}$$

Now, collecting all the K transmissions and using (2), the unknown clock coefficients $\boldsymbol{\theta}_i \triangleq [\dot{\phi}_i, \phi_i]$ can be estimated by solving for,

$$\begin{aligned} \hat{\boldsymbol{\theta}}_i &= \arg\min_{\boldsymbol{\theta}_i} \|\mathbf{A}_K \boldsymbol{\theta}_i - \boldsymbol{\epsilon}_i\|_2^2 & (3) \\ &= \left(\mathbf{A}_K^T \mathbf{A}_K\right)^{-1} \mathbf{A}_K^T \boldsymbol{\epsilon}_i = \mathbf{G}_K^{-1} \mathbf{b}_{i,K}, & (4) \end{aligned}$$

where $\mathbf{A}_K = [\mathbf{t},\ \mathbf{1}_K]$, $\mathbf{b}_{i,K} = \mathbf{A}_K^T \boldsymbol{\epsilon}_i$, $\mathbf{G}_K = \mathbf{A}_K^T \mathbf{A}_K$, $\mathbf{1}_K$ denotes a column vector of K ones and the measurement vectors are of the form,

$$\begin{aligned} \mathbf{t} &= \left[T_{0,1} \quad T_{0,2}, \ldots, T_{0,K}\right], & (5) \\ \boldsymbol{\epsilon}_i &= \left[\epsilon_{i,1} \quad \epsilon_{i,2}, \ldots, \epsilon_{i,K}\right]. & (6) \end{aligned}$$

Here $\hat{\boldsymbol{\theta}}_i$ is an estimate of the true clock parameters and (3) has a feasible solution provided $K \geq 2$ [19]. Observe that the matrix $\mathbf{G}_K$ is dependent only on the time stamps from the reference sink node $\mathbf{t}$. Hence, in case the (possibly varying) polling-interval δt is known advance, then the inversion $\mathbf{G}_K^{-1}$ can be estimated offline and stored locally, at the cost of more memory.

B. Iterative Least Square update

For a set of K time measurements, the number of multipliers to solve the least squares (LS) solution of (3) is $\mathcal{O}(2K)$, where the inversion of the Grammian matrix $\mathbf{G}_K$ is the most expensive operation. However, since the measurements arrive sequentially, the proposed least squares estimator can be solved row-iteratively [22]. Let $\mathbf{a}_k^T$ and $\boldsymbol{\epsilon}_{i,k}^T$ denote kth row input (during the k^{th} transmission) of $\mathbf{A}$ and $\boldsymbol{\epsilon}_i$, respectively. Then, solving (3) for $2 < k \leq K$ is equivalent to iteratively solving $\mathbf{G}_k^{-1} \mathbf{b}_{i,k}$, where the k^{th} update is given by,

$$\begin{aligned} \mathbf{G}_k^{-1} &\triangleq (\mathbf{G}_{k-1} + \mathbf{a}_k \mathbf{a}_k^T)^{-1} \\ &= \mathbf{G}_{k-1} - \frac{\mathbf{G}_{k-1}^{-1} (\mathbf{a}_k \mathbf{a}_k^T) \mathbf{G}_{k-1}^{-1}}{1 + (\mathbf{a}_k^T \mathbf{G}_{k-1}^{-1} \mathbf{a}_k)}, & (7) \\ \mathbf{b}_{i,k} &= \mathbf{b}_{i,k-1} + \mathbf{a}_k^T \epsilon_{i,k}. & (8) \end{aligned}$$

The initial estimate at $k = 2$ is obtained by solving the 2×2 linear system $\mathbf{G}_2^{-1} \mathbf{b}_{i,2}$, where

$$\begin{aligned} \mathbf{G}_2 &= \mathbf{A}_2^T \mathbf{A}_2 = \begin{bmatrix} G_{1,1}, & G_{1,1} \\ G_{2,1}, & G_{2,2} \end{bmatrix}, & (9) \\ \mathbf{G}_2^{-1} &= \frac{1}{\det(\mathbf{G}_2)} \begin{bmatrix} G_{2,2}, & G_{1,2} \\ -G_{1,2}, & G_{1,1} \end{bmatrix}, & (10) \\ \mathbf{b}_{i,2} &= \mathbf{A}_2 \boldsymbol{\epsilon}_{i,2}. & (11) \end{aligned}$$

As the iterative-LS update is very inexpensive in terms of memory and CPU cycles, it can be implemented on any embedded device. We integrated the above clock-offset estimation technique in our Sleeping Beauty implementation.

VII. Performance Evaluation of Sleeping Beauty

We evaluate the performance of Sleeping Beauty based on the overall energy consumption by the nodes in a WSN. In this regard, we study the reduction in duty cycle of the nodes over a longer period. Additionally, we monitor the performance in terms of packet reception ratio (PRR).

A. Implementation details

Sleeping Beauty is implemented on the Tmote Sky platform using the Contiki operating system. We use the core functionalities of Glossy to obtain the precise clock synchronization using fast flooding. We further adopted from our previous implementation of LWB and FS-LWB [23]. As Sleeping Beauty does not have any platform-specific component, it becomes directly usable on any other platform where Glossy will be ported to.

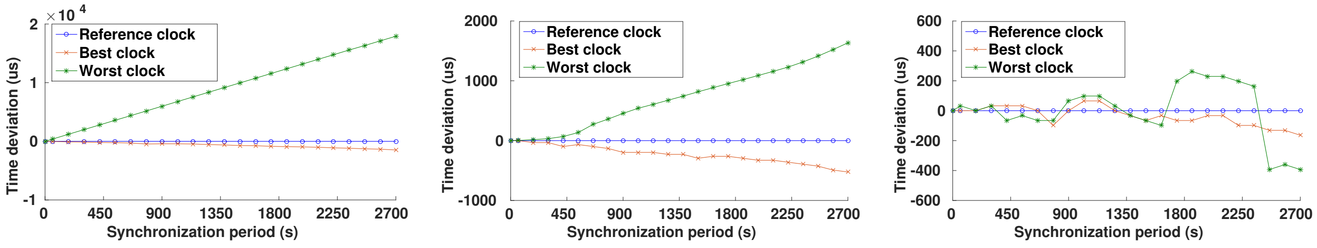
There are a few configuration parameters in Sleeping Beauty that can be tuned based on the application's requirements. Without loss of generality, we use the following values during our evaluation – the inter-packet interval I is set to 10 s, the scheduling interval S to $10I$ s (i.e., 100 s), and the time-out period T to end bootstrapping is set to 120 s.

B. Evaluation platforms

We present the results of our study carried out on the Indriya [11] and FlockLab [12] testbeds along with experiments using some local nodes in our lab. Our experiments were conducted on 80 and 32 TelosB nodes on Indriya and FlockLab, respectively. All communications are done using channel 26 and majority of our experiments are consucted at nights to avoid interference from WiFi. We performed two separate sets of experiments: one to evaluate the clock-offset estimation and another to evaluate the overall performance of Sleeping Beauty.

C. Accuracy of clock-offset estimation

First, we discuss the results related to the clock-offset estimation and correction. As the clocks on different nodes behave differently, we present the clock behavior of two nodes that showed the minimum (best) and maximum (worst) offset over time with respect to the reference node (sink) in Fig. 6. The clock of the first node is running slower than the reference node as the observed deviation is on the negative side. The



(a) clock behavior with respect to the reference clock (b) offset estimation error with fixed parameters (c) offset estimation with parameter updates

Fig. 6: Behavior of two clocks (best and worst) with respect to the reference clock.

clock of the second node instead is running faster than the reference node (Fig. 6a).

Using 120 training samples, the clock-offset parameters are estimated, where sync packets are sent every second. In the evaluation phase, sync packets are sent at various (longer) intervals. Before receiving the next sync packet, the node estimates the clock offset and adjusts its clock. Later, in steady state, when a sync packet is received the offset is calculated, if any. This offset denotes the estimation error. Note that during the evaluation, clocks are always readjusted with respect to the reference clock after receiving a sync packet.

Fig. 6b shows the estimation error for various synchronization periods. For each synchronization period, multiple measurements were performed and an average value is plotted in the figure. From this data, it is clear that the estimation error increases when the synchronization period increases. This behavior is due to the non-linear nature of the clock and the small set of the training data. The estimation error can be contained if the offset parameters are also updated during the evaluation phase after receiving each sync packet. Fig. 6c shows an improved clock correction and the estimation error is smaller compared to Fig. 6b. However, even in this case, the estimation error increases with respect to the synchronization period, though at a much lower rate.

Glossy maintains high synchronization accuracy among the nodes by transmitting sync packets frequently enough such that the clock offset of the nodes remain within a bound. Glossy defined this bound as the guard time and set it to 500 μs. Based on the experiments on Indriya, we discovered that some of the nodes become asynchronous (offset is higher than the guard time) even if a sync packet is sent as often as every 10 s. Thus, we assumed that to maintain synchronization for all the nodes sync packets need to be sent in 5 s intervals. However, if the application sends data at larger intervals, a lot of sync packets will be exchanged just to keep the network synchronized. Using our clock-offset estimation technique, a sync packet can be sent once in every 45 min, while the clock offset will be within the predefined threshold of 500 μs (Fig.6c).

D. Sleeping Beauty's performance

Next we study the performance of Sleeping Beauty. As the number of common sensing groups in a WSN can vary

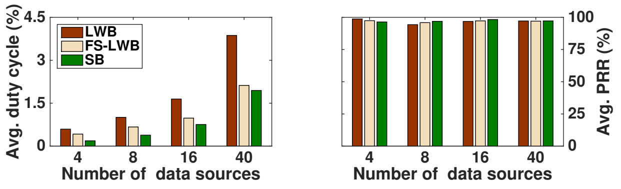


Fig. 7: Experiments on Indriya: average duty cycle per node over a long period and average packet reception ratio.

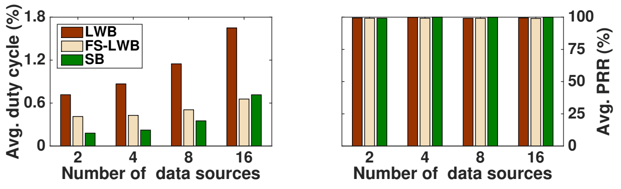
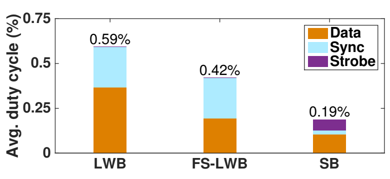


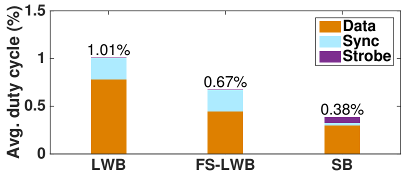
Fig. 8: Experiments on FlockLab: average duty cycle per node over a long period and average packet reception ratio.

significantly, we experimented with different sets of groups. We assumed that only one representative node from each group is sufficient to meet the sensing requirement where groups are formed based on correlation as described in [24]. Specifically, we experimented with four different group settings such that there are the following percentages of active nodes: 6.25%, 12.5%, 25%, and 50% while keeping the total number of nodes constant. That means, there are respectively 4, 8, 16, and 40 groups on Indriya. In case of FlockLab, there are 2, 4, 8, and 16 groups.

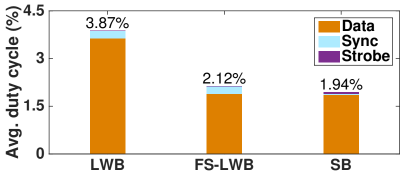
We compare Sleeping Beauty (SB) with two state-of-the-art protocols – LWB [10] and FS-LWB [13]. As the radio is the most significant energy consuming activity of a node, we compare the radio-on time (duty cycle) with both of these protocols. Figures 7 and 8 show the duty cycle and packet reception rate averaged across all nodes of the Indriya and FlockLab testbeds using: (i) LWB; (ii) FS-LWB, and (iii) Sleeping Beauty for the four different numbers of source nodes. For each experiment, we used 100 superframes, and the total number of data packets varied from 200 to 4000 (based on the number of source nodes). In all but one case Sleeping Beauty outperforms (FS-)LWB in terms of duty cycle, up to a factor of *three*, while achieving comparable packet reception rates. The savings in energy consumption are



(a) 4 source nodes



(b) 8 source nodes



(c) 40 source nodes

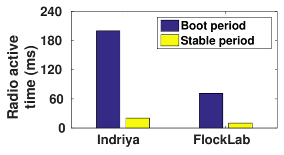
Fig. 9: Experiments on Indriya with 80 nodes: average radio-on time for nodes for various operations of Sleeping Beauty (SB).

most pronounced when the least number of source nodes are active. This matches intuition as only then Sleeping Beauty can put nodes to sleep for prolonged periods. The observed PRRs show that most of the data is delivered at the sink; only on the Indriya testbed some packets are lost. There is, however, no clear winner amongst the three protocols. Depending on the number of source nodes, Sleeping Beauty looses more or fewer packets than (FS-)LWB. We conjecture that the underlying cause for the drop in PRR is the constructive interference (CI) mechanism, whose effectiveness depends on the number of simultaneous transmitters. With too few transmitters nothing is gained; with too many, a slight misalignment at the symbol granularity leads to packet loss. Depending on the number of active sources, a different protocol may select the ideal set of simultaneous transmitter for CI.

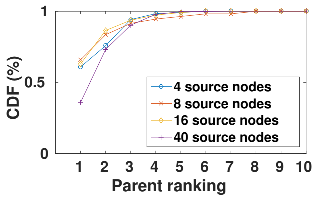
To gain a deeper understanding of how Sleeping Beauty can provide the same service (PRR) while consuming less energy, Fig. 9a shows the average radio-on time broken down into the main activities of the protocols (handling data, sync, and strobe packets). In LWB, even if there are a few data sources and few data slots associated with them, all nodes participate in all data slots. FS-LWB reduces the overall duty cycle of the nodes, by using only a subset of the nodes in a data slot associated with a particular data source. Because of the more refined topology information collected by Sleeping Beauty, it is capable of putting even more nodes to sleep, reducing the average time spent on handling (forwarding) data even further. For example, when 8 source nodes are active, LWB spends 0.78 percentage points on handling data packets, while FS-LWB reduces that to 0.44 and Sleeping Beauty only requires 0.30 percentage point.

Besides gaining efficiency by flooding less data, Sleeping Beauty also benefits greatly from the improved synchronization method. LWB and FS-LWB spend about 0.23 percentage point of their duty cycle on handling synchronization packets to maintain Glossy's $\leq 500\mu$s timing requirements. Sleeping Beauty on the other hand, spends only 0.02 percentage points on handling sync packets. This coarsely matches the ratio of injection rates, with (FS-)LWB sending out sync packets once every 5 s, and Sleeping Beauty synchronizing once every 100 s.

Sleeping Beauty has some extra overhead (0.06 percentage points) in the form of strobe packets, which are are used to collect topology information. This amounts to a total synchronization cost of 0.08 percentage points, which still compares favorably to that of (FS-)LWB (0.23). The importance of



(a)



(b)

Fig. 10: (a) Strobe overhead during bootstrapping and steady state. (b) Active parent distribution across the ranked list.

the 'smart strobing' policy, which only updates ETX values for potential parent nodes (cf. Section V-B), becomes clear when comparing the overhead of Sleeping Beauty during bootstrapping and steady state. Fig. 10a shows that listening to all nodes in the network (bootstrapping) is 7 to 10 times more expensive than just monitoring the potential parent nodes (steady state) on FlockLab and Indriya, respectively. This factor matches with the ratio of the total number of nodes in the testbed over the average number of parent nodes: $\frac{32}{4.5} = 7$ for FlockLab and $\frac{80}{8} = 10$ for Indriya.

The collected neighborhood information is reported back to the sink, but only partially to limit the overhead. Sleeping Beauty orders the neighbors based on their ETX values and sends out the top-X entries with the shortest routes to the sink. It is important to report the right number of potential parents as, on the one hand, the overhead is directly proportional to it, and on the other hand, the sink can make better decisions on which nodes to activate if it has more complete knowledge of the topology. To study this trade-off we conducted an experiment in which we configured nodes to report the top-10 list with potential parents. We then monitored which parents were then selected by the sink for the next scheduling round. Fig. 10b shows the cumulative distribution of the selected, ranked parents for different numbers of active sources on the Indriya testbed. In 97% of the cases the sink selected 5 or fewer parents to become active, prompting us to conclude that lists of 5 parents provide sufficient topology information for the sink to build an efficient data-collection overlay.

Knowing that the overall efficiency of Sleeping Beauty is mainly dictated by the number of active nodes, it is tempting to construct overlays where each (source) node has only 1 active parent. However, such sparse overlays are likely to be susceptible to packet loss induced by errors on the wireless

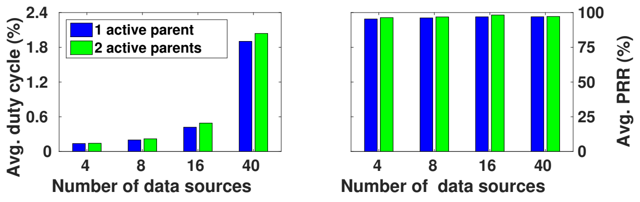


Fig. 11: Comparison of average radio-on time and PRR when one and two parent nodes are selected active.

channel. As the underlying fast-flooding mechanism relies on exercising all available links, it could be advantageous to activate a few redundant (parent) nodes. To study the PRR-redundancy trade-off, we experimented with selecting one or two parent nodes. Fig. 11 shows the difference in PRR and duty cycle for both cases on Indriya with different numbers of active source nodes. To our surprise the PRR is hardly affected. Studying the single-parent overlays in detail, we noted that in many cases selecting the best parent of one node, as a side effect, also implied activating the 2nd or 3rd best parent of other nodes. The effect of selecting 1 or 2 parents is a bit more pronounced for the duty cycle, but still limited, showing that either option is fine.

VIII. Conclusions

To provision against adverse network conditions and node failures, WSN deployments typically contain redundant nodes. Node-scheduling exploits this feature by limiting the number of active nodes to achieve energy-efficient network operation without violating the coverage requirements of the applications. However, it is not sufficient to just limit the number of active nodes; active nodes should form a *connected* (sub)network. There exists a large body of work on selecting a connected subset of nodes covering the complete deployment. These optimized node-scheduling techniques, however, are generally not applicable to real-world deployments as they require complete topological information, which is difficult and expensive to obtain accurately.

We presented Sleeping Beauty, an energy-efficient communication protocol that operates with partial topological information, yet outperforms state-of-the-art flooding-based protocols (LWB and FS-LWB) in node-scheduling scenarios. Sleeping Beauty accomplishes this by including (i) an efficient neighbor-discovery mechanism that enables the selection of a minimal, but connected set of active nodes, and (ii) a simple, but elegant clock-offset estimation technique that allows nodes to sleep for a longer duration without the need for explicit resynchronization. The latter is important for allowing the use of efficient time-triggered flooding *a la* Glossy. Our time synchronization technique can be used in any application. We compared the performance of Sleeping Beauty with state-of-the-art protocols on two public testbeds (Indriya and FlockLab), and showed that the same performance (PRR) can be achieved at a fraction of the energy consumption. We recorded a factor of *three* reduction for the best case scenario with 5 % active nodes.

References

[1] A. Chamam and S. Pierre, “On the planning of wireless sensor networks: Energy-efficient clustering under the joint routing and coverage constraint,” *IEEE Trans. on Mobile Computing*, vol. 8, no. 8, 2009.

[2] S. Mini, S. K. Udgata, and S. L. Sabat, “Sensor deployment and scheduling for target coverage problem in wireless sensor networks,” *IEEE Sensors Journal*, vol. 14, no. 3, pp. 636–644, 2014.

[3] H. Gupta, V. Navda, S. Das, and V. Chowdhary, “Efficient gathering of correlated data in sensor networks,” *ACM Transactions on Sensor Networks*, vol. 4, no. 1, p. 4, 2008.

[4] L. A. Villas, A. Boukerche, H. De Oliveira, R. De Araujo, and A. A. Loureiro, “A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks,” *Ad Hoc Networks*, 2014.

[5] M. Cardei and J. Wu, “Energy-efficient coverage problems in wireless ad-hoc sensor networks,” *Computer communications*, vol. 29, no. 4, pp. 413–420, 2006.

[6] C. Sarkar, V. S. Rao, R. Venkatesha Prasad, and K. Langendoen, “Sleep-route: Assured sensing with aggressively sleeping nodes,” in *IEEE 11th Conf. on Mobile Ad Hoc and Sensor Systems*, 2014, pp. 237–241.

[7] H. M. Ammari and S. K. Das, “Centralized and clustered k-coverage protocols for wireless sensor networks,” *IEEE Transactions on Computers*, vol. 61, no. 1, pp. 118–133, 2012.

[8] Y. Yao, Q. Cao, and A. V. Vasilakos, “Edal: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for wireless sensor networks,” in *IEEE MASS*, 2013, pp. 182–190.

[9] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, “Efficient network flooding and time synchronization with glossy,” in *10th ACM/IEEE Conf. on Information Processing in Sensor Networks*, 2011, pp. 73–84.

[10] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele, “Low-power wireless bus,” in *10th ACM Conf. on Embedded Network Sensor Systems*, 2012, pp. 1–14.

[11] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, “Indriya: A low-cost, 3D wireless sensor network testbed,” in *Testbeds and Research Infrastructure. Development of Networks and Communities*. Springer, 2012, pp. 302–316.

[12] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel, “Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems,” in *ACM/IEEE IPSN*, 2013.

[13] D. Carlson, M. Chang, A. Terzis, Y. Chen, and O. Gnawali, “ Forwarder Selection in Multi-Transmitter Networks ,” in *9th Conference on Distributed Computing in Sensor Systems*. IEEE, May 2013.

[14] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, “Collection tree protocol,” in *7th ACM Conf. on Embedded Networked Sensor Systems*, 2009, pp. 1–14.

[15] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson, “Low power, low delay: opportunistic routing meets duty cycling,” in *11th ACM/IEEE Conf. on Information Processing in Sensor Networks*, 2012.

[16] O. Landsiedel, F. Ferrari, and M. Zimmerling, “Chaos: Versatile and efficient all-to-all data sharing and in-network processing at scale,” in *11th ACM Conf. on Embedded Networked Sensor Systems*, 2013.

[17] P. Levis, N. Patel, D. Culler, and S. Shenker, “Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks,” in *NSDI’04*. USENIX Association.

[18] K. Leentvaar and J. H. Flint, “The capture effect in fm receivers,” *IEEE Transactions on Communications*, vol. 24, no. 5, pp. 531–539, 1976.

[19] R. T. Rajan and A.-J. van der Veen, “Joint ranging and synchronization for an anchorless network of mobile nodes,” *IEEE Transactions on Signal Processing,*, vol. 63, no. 8, pp. 1925–1940, 4 2015.

[20] R. T. Rajan, M. Bentum, and A.-J. Boonstra, “Synchronization for space based ultra low frequency interferometry,” in *IEEE Aerospace Conference*, 3 2013, pp. 1–8.

[21] E. Serpedin and Q. M. Chaudhari, *Synchronization in Wireless Sensor Networks: Parameter Estimation, Peformance Benchmarks, and Protocols*, 1st ed. NY, USA: Cambridge University Press, 2009.

[22] L. L. Scharf, *Statistical signal processing*. Addison-Wesley Reading, MA, 1991, vol. 98.

[23] C. Sarkar, “Lwb and fs-lwb implementation for sky nodes using contiki,” arXiv preprint - https://arxiv.org/abs/1607.06622, 2016.

[24] C. Sarkar, V. S. Rao, R. V. Prasad, S. N. Das, S. Misra, and A. Vasilakos, “Vsf: An energy-efficient sensing framework using virtual sensors,” *IEEE Sensors Journal*, vol. 16, no. 12, pp. 5046–5059, 2016.