Noname manuscript No.
(will be inserted by the editor)

BailighPulse: A Low Duty Cycle Data Gathering Protocol For Mostly-Off Wireless Sensor Networks

Wojciech Bober and Chris J. Bleakley

UCD Complex & Adaptive Systems Laboratory,

UCD School of Computer Science and Informatics,

University College Dublin, Ireland

Abstract Mostly-off sensor network applications alternate between long periods of inactivity (ranging from minutes to hours) and short periods of activity (normally a few seconds). From an energy consumption point of view, it is desirable that the network switch off completely during application inactive periods and wake-up efficiently at the start of application active periods. The fundamental problem preventing this is the inter-node clock skew arising from the network being off for a long period. Existing solutions maintain synchronization during the inactive period or use the radio excessively to enable asynchronous wake-up. Herein, we propose BailighPulse, a low duty cycle data gathering protocol for mostly-off WSN applications. BailighPulse incorporates a novel multi-hop wake-up scheme that allows for energy efficient recovery of network synchronization after long off periods. The scheme uses a staggered wake-up schedule and optimized channel polling during wake-up based on knowledge of the pre-defined application-level schedule. Herein, we provide an extensive assessment of the protocol's performance including an analytic model, simulations, and testbed results. We show that, for a homogeneous schedule with collection period greater than 2 minutes, BailighPulse reduces radio duty cycles by at least 30% and 90% compared to Dozer and B-MAC, respectively. We also show that BailighPulse is able to reduce radio duty cycle by to 68% for a heterogeneous schedule under similar conditions.

1 Introduction

In data gathering applications Wireless Sensor Network nodes periodically sample their sensors and transmit the measurements back to a single node (the sink). In *mostly-off* WSN applications [1], the application alternates between long periods of inactivity (ranging from minutes to hours) and short periods of activity (normally a few seconds). Examples of these applications include predictive maintenance [2], utility metering networks [3], and environmental research [4].

Address(es) of author(s) should be given

From an energy consumption point of view, it is desirable that a Wireless Sensor Network follows application characteristics and switches off completely during application inactive periods and switches on efficiently at the start of application active periods. This mode of operation where a network is completely switched off between short on periodsis referred to as *disconnected mode* [5]. Clearly, to preserve energy, the time for which the sensor network is on should be as short as possible. Dutta *et al.* [5] showed that, in principle, a network using this mode of operation can operate at a duty cycle of 0.01%. In practice, however, achieving this goal has proven difficult due the fundamental problem of inter-node clock skew arising from the network being off for a long period [6]. Clock skew results from crystal oscillator instability which is mainly caused by temperature variation. In outdoor environments this can lead to clock drifts of 125-400 ms/h (35-110 ppm) [7]. Due to the resulting clock skew, nodes wake-up at different times and need to wait until the whole sensor network is ready before communicating. This increases the network on time, increasing the nodes' duty cycle.

The problem can be addressed at the hardware or software level. At the hardware level, a temperature compensated crystal oscillator (TXCO), which has minimal clock drift can be used, so that nodes wake-up at almost the same time [6]. Another hardware approach uses an out-of-band wake-up radio that allows efficient asynchronous network wake-up [8]. Hardware approaches, however, are rarely used since they require additional electronics which increase sensor node cost and size. Hence most research focuses on the software level in the form of energy efficient data gathering protocols.

Data gathering protocols can be categorized according to whether they use synchronous or asynchronous wake-up methods. In the case of synchronous wake-up, either explicit or implicit synchronization is used during the application inactive period. In the case of asynchronous wake-up, the sensor network is kept on so that the drift does not matter. Both approaches require energy which is wasted from the application point of view. As a result, no previously proposed protocol achieves the theoretical duty cycle target.

Typically, in mostly-off sensor network applications the application schedule (the times of sensor sampling and data collection) is determined by a domain expert and fixed *a priori*. An application schedule might, for example, set the the network to sample sensors less often during the day and more often during the night. In the protocol presented herein, we take advantage of this property to propose a different approach to dealing with the clock skew problem. In BailighPulse the network is switched off completely when the application is inactive. Fine grained network synchronization is then recovered at the beginning of the active phase using a novel multi-hop wake-up scheme. This wake-up scheme uses a staggered wake-up schedule and optimizes channel polling based on knowledge of the pre-defined application-level schedule. Using this approach energy consumption during wake-up is minimized.

To the best of our knowledge, BailighPulse is the first data gathering protocol for mostly-off sensor networks utilizing coarsely synchronized multi-hop wake-up.. BailighPulse is suitable for use with complex scheduling algorithms such as those described in [9, 10, 11, 12]. These algorithms derive sampling schedules based on criteria such as sensing converge and data correlation. For completeness we propose a compact notation for specifying the application level schedules. The notation allows for expression of application activity, i.e., sampling and collection, in a simple and compact way.

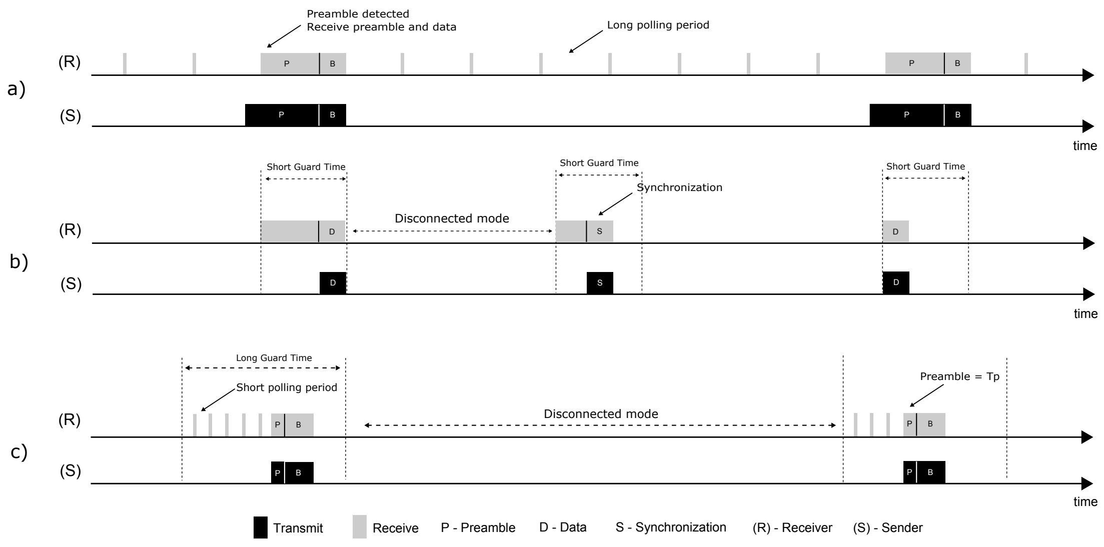


Fig. 1: Comparison of different wake-up strategies in a one-hop scenario: a) asynchronous wake-up b) synchronous wake-up c) BailighPulse wake-up

The results of analytic and simulation based evaluations show that, for low data collection rates, the proposed protocol significantly reduces the average radio duty cycle of the network. We show that in mostly-off applications BailighPulse outperforms Dozer, which is a state-of-art data gathering protocol.

The reminder of the paper is organized as follows: In Section 2 we discuss related data gathering protocols. Section 3 defines assumptions and formulates the problem statement. Section 5 gives details of the protocol design. An analytic model of BailighPulse is given in Section 6. Performance evaluation of the protocol is presented in Section 7. A case study and testbed results are discussed in Section 8. We conclude the paper with Section 9.

2 Related Work

A large number of techniques aiming to reduce energy consumption in WSNs have been proposed [13, 14]. The solutions most closely related to the proposal are data gathering protocols, i.e., protocols specifically designed for collecting data from the entire network. Typically, data gathering protocols are cross-layer, integrating MAC layer, routing layer, and energy management mechanisms. From the point of view of mostly-off sensor network applications, these protocols can be divided according to whether they use synchronous or asynchronous wake-up.

A simplified overview of both approaches is shown in Figure 1. In the first case (Figure 1(a)), nodes are not synchronized and periodically poll (check) the wireless channel for a wake-up beacon. The polling period is typically in the range of 100-1000 ms. However, energy is wasted when the application does not require transmission of data. In the second case, nodes are synchronized (Figure 1(b)) and time synchronization is maintained in the network. This means that clock drift is kept low. This allows the nodes to wake-up at scheduled times and send data immediately. Although less energy is wasted than in the case of asynchronous wake-up, energy is still wasted on maintaining time synchronization.

Most proposals based on asynchronous wake-up are focused on the MAC layer. Protocols such as B-MAC [15], Wise-MAC [16], X-MAC [17] use periodic channel polling to reduce energy consumption. For data gathering

Table 1: Comparison of data gathering protocols

Approach	Koala [21]	CTP + BoX [18]	DISSense [20]	Dozer [22]	AppSleep [2]	BailighPulse
Typical Average Duty Cycle	0.1% (N:25 PP:20s)	1%(N:131 PP:1s)	0.1% (N:14 CP:60min)	0.2% (N:40 CP:2min)	1%	0.01% (N:40 CP:2min)
Wake-Up method	Asynchronous	Asynchronous	Synchronous	Synchronous	Synchronous	Synchronous
Typical Collection Period	Hours	10s of seconds	10s of minutes	10s of minutes	Hours	10s of minutes - hours
Disconnected mode	Yes	No	No	No	No	Yes
Integrated MAC	Yes	No	No	Yes	No	Yes
Adaptability to wake-up scheme	No	No	Yes	No	No	Yes
N: Number of nodes; PP: polling period; CP: collection period						

applications, these protocols must be used with a collection protocol which provides packet routing. One of the most recognized collection protocols is the Collection Tree Protocol (CTP) [18], which is a standard data gathering protocol distributed with TinyOS [19]. The main drawback of CTP, when used in mostly-off sensor network applications, is that it requires the network to be active during the application inactive period. This is required in order to maintain the routing state of the data gathering tree. The duty cycle of CTP is thus directly related to the duty cycle of the underlying MAC protocol, with typical values of 1%. In DISSense [20], the authors added an adaptation layer to CTP which allows the network to switch off between active periods. Thanks to this, DISSense is able to achieve a duty cycle of 0.1% for a collection period of 60 minutes.

Koala [21] is a data gathering protocol strictly designed for mostly-off sensor network applications. Koala uses an asynchronous wake-up technique called Low Power Probing (LPP). In LPP nodes actively probe the channel in order to detect a wake-up message sent by the sink. This allows for data collection to be initiated by the sink at any time. Unlike CTP, Koala does not maintain the routing state between application inactive periods. Instead, the data gathering tree is recreated at the beginning of the active period. The approach used by Koala, is beneficial in applications with collection periods greater than tens of hours. In this way, the considerable overhead of establishing the data gathering tree is amortized.

Protocols using synchronous wake-up have been proposed both as MAC protocols and cross-layer data gathering protocols. Synchronous MAC protocols include well known protocols such as S-MAC [23], Z-MAC [24], T-MAC[25]. As in the case of MAC protocols which use asynchronous wake-up, an appropriate collection protocol must be used for data gathering applications. Although these protocols achieve low duty cycles they are outperformed by cross-layer approaches such as Dozer [22].

Dozer [22] is a state of the art protocol which incorporates synchronous wake-up, MAC layer, topology control and routing. The key idea in Dozer is local scheduling. Instead of creating a global schedule, each node maintains a separate schedule for its role as a parent and as a child. As a parent, the node is responsible for assigning a transmission slot to its children. As a child, the node is obliged to transmit only at a given time. Each schedule cycle is initialized and synchronized by a beacon. Since there is no global time synchronization, messages from two adjacent nodes, which are not part of the same schedule may collide. Instead of the common approach of preventing collisions at the MAC layer, Dozer explicitly permits and handles them by retransmission. Consecutive collisions may be caused by unintentional alignment of two schedules. To solve this problem, each frame is randomly extended, thus after some time two aligned schedules drift apart. Dozer achieves a very low duty cycle of 0.1% verified by a testbed evaluation.

AppSleep [2] is a synchronous wake-up counterpart of Koala. Similarly it does not maintain routing state between application active periods, but nodes are scheduled to wake-up after a certain time. Time synchronization is maintained during the inactive period to control clock skew. At the beginning of the active period AppSleep uses the DSDV [26] protocol to establish routing and collect data from the network.

In Bailigh [27] we proposed a technique for optimizing channel polling with respect to data collection period. The proposed optimization scheme relied on knowledge of the predefined application schedule. In Bailigh optimized channel polling was used during data collection. This concept is used herein but is applied as part of a novel multi-hop wake-up pulse. This allows for reduction in energy wastage due to the inter-node clock drift present in mostly-off sensor networks.

BailighPulse is unique in three aspects (Figure 1(c)). Firstly, it is a data gathering protocol which uses a synchronous wake-up but does not require maintenance of time synchronization during the application inactive period. Instead nodes' clocks are allowed to drift and fine grained synchronization is recovered by means of a multi-hop wake-up scheme. Secondly, energy consumption during wake-up is minimized by means of optimized channel polling. Thirdly, the protocol adapts the wake-up scheme to the current collection schedule using information from the application layer.

Table 1 summarizes the data gathering protocols discussed herein and compares them to BailighPulse.

3 Problem statement

In this paper we address the problem of energy efficient data gathering in *mostly-off* sensor networks applications [1]. It is assumed that the data from the sensor network is collected periodically at very low rates, i.e., greater than tens of minutes. It is assumed that the data is collected from the network according to a predefined schedule. It is also assumed that the application is tolerant to some latency and instantaneous access to sensor readings is not required. Hence the sensor network can operate in disconnected mode between consecutive active periods, i.e., network communication can be deactivated.

In general, for energy efficient data gathering, the sensor network has to switch on at a predefined time, transmit the data to the sink node, and switch off as soon as possible. In a multi hop network, data collection cannot start until all nodes are on. Hence, the question is: how to minimize the energy consumption of the sensor network operating in disconnected mode, taking into consideration unpredictable clock drift between nodes?

Formally the problem can be defined as follows: for a multi-hop network of N nodes, minimize the mean radio Duty Cycle (DC) DC_{avg} required for collection of sensor data according to a sampling and collection schedule S given that nodes have a clock accuracy of δ ppm. Collection of sensor data is defined as the transfer of sensor data from all nodes to the sink. The mean radio DC is defined as:

$$DC_{avg} = \sum_{n=1}^{N} \frac{T_{on}(n)}{T_{on}(n) + T_{off}(n)} \tag{1}$$

where $T_{on}(n)$ is the period of time for which the radio of node n is on and $T_{off}(n)$ is the period of time for which the radio of node n is off, as calculated over a representative data collection interval.

The node data sampling and collection schedule S_n is determined by the needs of the application and is known *a priori*. The base data collection period is T_{bp} seconds. In the simple homogeneous case, all nodes send data nearly simultaneously and have the same collection period, which must be an integer multiple of the base period, i.e., $T_{cp} = kT_{bp}$, where $k \in N$. In the general heterogeneous case, nodes are allowed to have different collection periods but all collection periods must be integer multiples of the base period, i.e., $T^n_{cp} = k_n T_{bp}$, where $k_n \in N$. For the applications of interest herein $T_{cp} > 120$s in both the homeneous and heterogenous cases. It is assumed that the worst clock accuracy of nodes r_{skew} is known a priori. The actual clock accuracy can vary between nodes over time due to dependency on temperature and physical properties of the crystal. Typical values of drift are assumed to be in the range 20 - 500 ppm.

4 Application Schedule

The approach proposed herein requires an abstract representation of application sensing and collection schedules. In the simplest case these can be represented as two variables which define the sensing and collection period. However, a number of recently proposed scheduling algorithms such as [9, 10, 11, 12] require more powerful schedule abstraction. The abstraction should be sufficiently flexible so as to allow description of both homogeneous and heterogeneous schedules. Homogeneous schedules require that all nodes sample at the same time and that data collection is performed for all nodes at the same time. Heterogeneous schedules allow for subsets of nodes to have unique sampling and collection schedules. This allows for running sampling and collection schemes where parts of the network work interchangeably in a round robin fashion, thus saving energy. Since the *application schedule* must be disseminated and stored in node memory, it is crucial that the abstraction is compact. We propose a notation which uses a fixed time period as a base unit of time. Clearly there is a trade-off between the notation's expressiveness and compactness. We argue that the proposed based period notation is sufficient for most applications using *mostly-off sensor networks.*

Formally, we define a periodic schedule for node n as $S_n = \{t_1, t_2, \dots, t_k\}$. Each element $t_k \in S_n$ is a schedule task. The schedule task is represented as a triple $t_k = (s_k, f_k, p_k)$, where s_k represents the start time, f_k represents the finish time, and p_k represents the task periodicity. Time in the schedule is represented in terms of a base period T_{bp} i.e., $\forall t_k \in S_n(s_k = c_s T_{bp},\ f_k = c_f T_{bp},\ p_k = c_p T_{bp})$, where $c_s,\ c_f,\ c_p \in N$. The base period T_{bp} is the smallest unit of time that the schedule can represent. The duration of the base period depends on application requirements. Typically the base period duration is in the range of seconds to minutes. The task start and finish times, as well as periodicity, are relative to the schedule global period T_{gp}. The global period is also defined in terms of base period, that is $T_{gp} = c_g T_{bp}$where $c_g \in N$. The base and global periods are the same for all nodes in the network. After the global period of a schedule is over, the schedule is repeated.

Figure 2 shows two schedules. The nodes are capable of measuring two quantities: temperature T and humidity H. Let's assume that the collection schedule S^c is the same as sampling schedule S^s and that the base period T_{bp} equals 2 minutes. The global period is defined as $T_{gp} = 8T_{bp}$, hence its duration is 16 minutes. The first schedule, defined as $S_{1,2} = \{t_1 = \{0, 7, 2\}, t_2 = \{1, 3, 2\}\}$, is a homogeneous schedule. This means that all nodes in

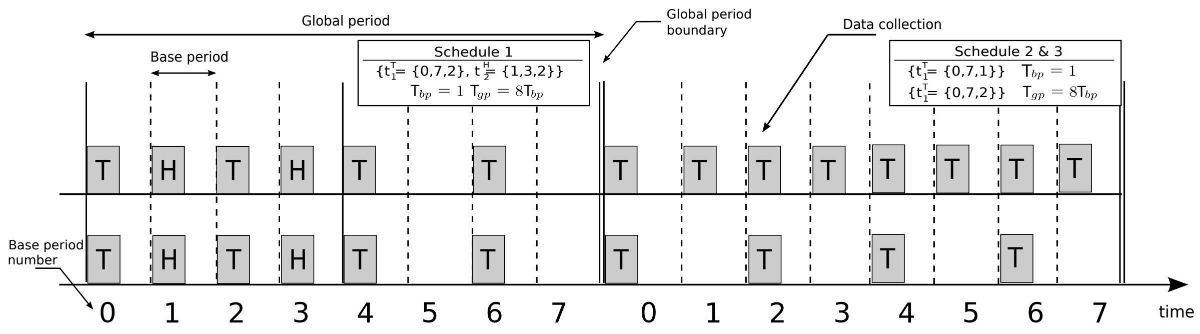


Fig. 2: Schedule representation example

the network execute the same schedule. The schedule defines two tasks. The first task t_1 defines temperature sampling, which is done every four minutes. The second task t_2 defines a humidity sampling task which is also sampled every four minutes, however, only for the half of the global period.

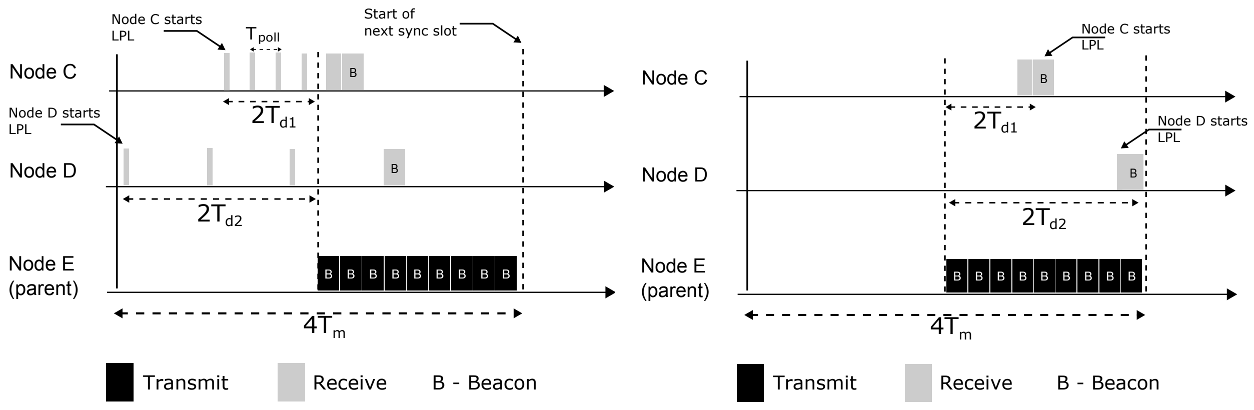
After the first global period expires, nodes execute a heterogeneous schedule. In this case, the network is partitioned into two subsets, each executing a different schedule. The first schedule is defined as $S_1 = \{t_1 = \{0, 7, 1\}\}$ and the second as $S_2 = \{t_2 = \{0, 7, 2\}\}$. These two schedules represent a sub-sampling scenario, where part of the network samples less often.

The proposed notation can be compactly represented in node memory. With the base period $T_{bp} = 2$ minutes we can define a task with a global period of 8.53 hours using 8 bit time representation. Therefore, a collection task can be defined for 8 hours in advance with a granularity of 2 minutes using only 3 bytes (task triple is defined using tree time variables). Depending on the application requirements and node capabilities various tasks such as, sampling, collection, computation, and maintenance, can be defined.

5 BailighPulse

BailighPulse alternates between four states: *Initialization, Wake-up, Collection, and Inactive.* The *Initialization* state is used by BailighPulse to create a routing tree and disseminate application schedules. In addition, the *Initialization* state is executed periodically in order to refresh network topology. For brevity we do not discuss herein operation during this state. The reader is refered to [28] which describes the process in detail.

Nodes enter the *Wake-Up* state according to the collection tasks defined in the application schedule. In this state, nodes switch on their radio and recover fine grained synchronization. During the state the radio duty cycle is optimized taking into consideration the time elapsed since the last data collection. After the *Wake-Up* state is complete, the network enters the Collection state and nodes upload data according to a staggered communication schedule. After the upload is complete, nodes switch to the *Inactive* state. Nodes remain in the *Inactive* until the next *Wake-Up* state is scheduled.



(a) Nodes C and D are faster than parent node E. (b) Nodes C and D are slower than parent node E.

Fig. 3: BailighPulse wake-up scheme in one-hop scenario for nodes with different drift and schedule.

5.1 Wake-Up

Whenever the application schedule defines that data collection should start, nodes enter the *Wake-Up* state. Protocol operation in this state is key to BailighPulse's ultra low duty cycle.

As explained earlier, waking all nodes at exactly the same time is not possible due to the drift of the crystal oscillators. The clock skew depends on the time elapsed since the last synchronization T_{sync} and the accuracy of the node's crystal oscillator r_{skew}. The maximal drift is $T_d = T_{sync}\, r_{skew}$. If T_I is the ideal clock then the fastest node wakes up at $T_I - T_d$ and the slowest at $T_I + T_d$. Hence, the maximum drift between two nodes is $2T_d$.

In BailighPulse nodes wake-up coarsely synchronized. In order to handle drift, each node wakes up $2T_d$ before its estimate of the wake-up time, and at, most waits for $4T_d$. The time $4T_d$ is called the guard time. Keeping the radio constantly on during the guard time would lead to excessive energy consumption because the guard time is typically long, e.g., hundreds of ms. Instead nodes periodically poll the channel for a wake-up pulse. In BailighPulse, channel polling is performed only during the guard time. This is unlike previously proposed asynchronous techniques (B-MAC, X-MAC) which poll the channel continuously. Also, the polling period T_{poll} is much shorter, typically a few ms.

The wake-up pulse consists of a trail of beacons, i.e., a number of small consecutive beacon packets. A node will wake-up on reception of a single beacon. All other beacons are rejected by the node as duplicated packets. Consecutive beacons of the wake-up pulse are transmitted without gaps. As the receiver performs only a few channel checks during the guard time, gaps between consecutive beacons would increase the chance of missing the wake-up pulse. For this reason the wake-up pulse transmission cannot be terminatted by the receiver acknowledgment as in the case of X-MAC.

The length of the wake-up pulse is equal to the polling period T_{poll} and is automatically optimized with respect to the maximum drift T_d. In general, the duration of the polling period reflects the drift - as the drift grows so does the the polling period. Details of how to select the optimum values for the wake-up pulse duration and polling period are given in Section 6.

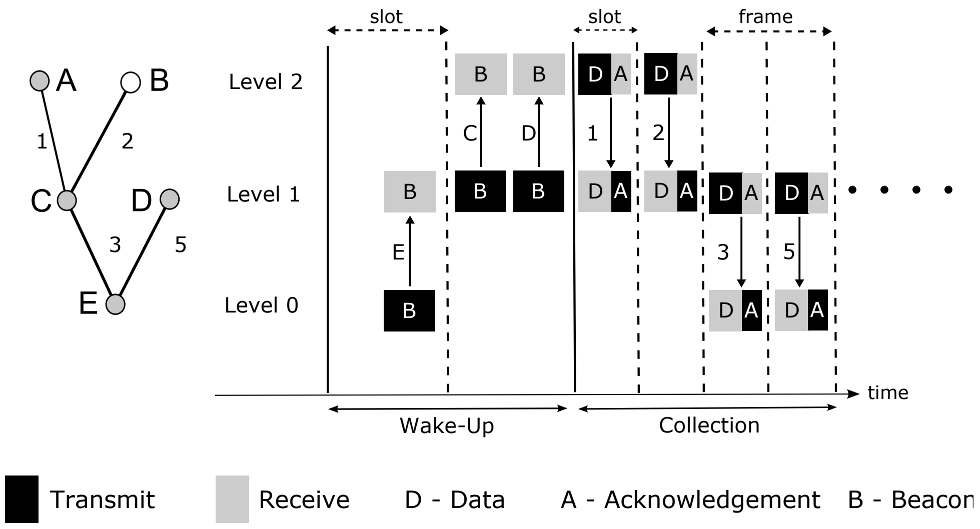


Fig. 4: Overview of the multi-hop wake-up scheme.

This wake-up primitive can be also used when nodes follow heterogeneous collection schedules. An example of such a case is shown in Figure 3. In the example, collection from node C is done more often than from node D. Due to this, the guard time of node D is longer than that of node C. As each node uses a polling period optimized to its own drift, node D's polling period is greater than the polling period of node C. In order to wake-up both nodes the length of wake-up pulse matches the worst case drift, i.e., of node D.

In order to wake-up and resynchronize the entire network, BailighPulse uses a staggered wake-up scheme. The scheme uses the proposed wake-up primitive and the staggered schedule established during *Initialization*. An example of the scheme is shown in Figure 4. The scheme is organized according to frames and slots. A frame is the period of time when adjacent levels communicate (e.g., Level 0 and 1). A slot is an offset within the frame. Its index $slot_i$ is assigned during *Initialization*. The duration of a slot during wake-up depends on the drift of a node. To ensure proper organization of the wake-up scheme when heterogeneous schedules are used, the synchronization slot duration is calculated assuming the worst case drift in the current collection period. This information is part of the schedule, and is known to all nodes in the network.

The scheme starts with the sink which broadcasts a wake-up pulse. Upon receiving the wake-up pulse, nodes connected to the sink as children store the offset between their clock and their parent's clock. This offset is used to synchronize the child's clock by aligning its offset. The relative skew between the nodes is not calculated, as the time between the *Wake-Up* and *Collection* state is very short, hence the clock drift can be neglected during the short active period. Next, a node rebroadcasts the wake-up pulse using its slot $slot_i$. In order to improve reliability, slot assignment should be collision free.

If requested by the user, the wake-up pulse can trigger execution of the *Initialization* state, by setting an appropriate flag in the wake-up pulse beacon. This is needed if a new schedule must be uploaded or the network topology changed (due to adding or removing nodes). If that happens, the network enters the *Initialization* state for a period of time defined in the wake-up pulse. After the defined period of time elapses, the network enters the *Wake-Up* state.

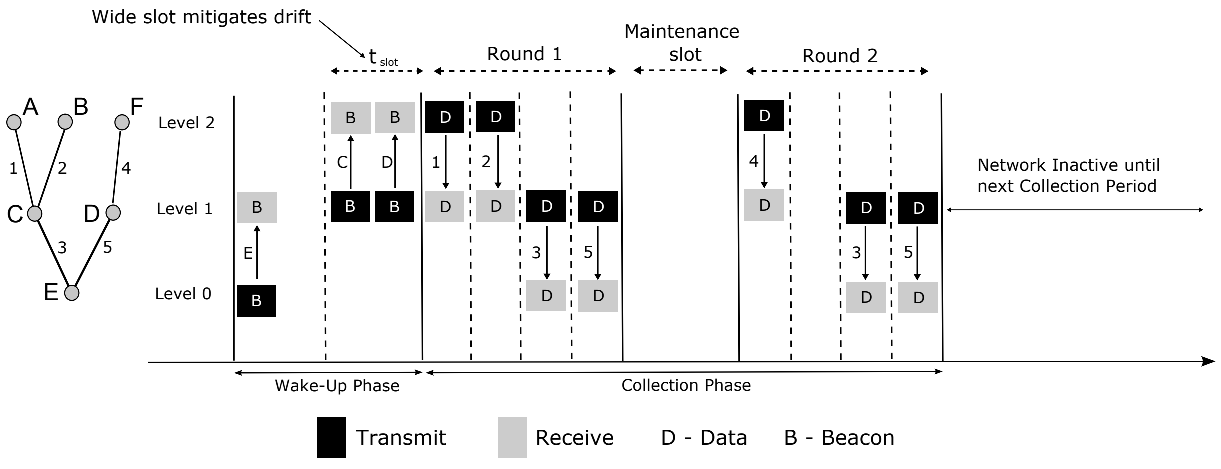


Fig. 5: Collection process in detail

5.2 Collection

After wake-up is completed, the nodes are synchronized and ready for data collection. The *Collection* state, as in case of the *Wake-Up* state, uses a staggered communication schedule. Data collection is organized into consecutive rounds. A round is the time required to send a packet from the furthermost leaf node to the sink. A round consists of frames. A frame is the period of time when adjacent levels communicate. In Figure 5, nodes A and B are in the same frame and send their data to parent C. A frame consists of slots. Each child communicating with the same parent has an individual slot. For the best performance, the slot assignment should be collision free. However, the protocol is able to work with non-collision free assignment by using CCA in each slot. The slot is long enough to allow for sending P_c packets. Each transmitted packet must be acknowledged by a parent. In the example shown in Figure 5, one packet transmission is made during a slot. All transmission is stopped before the end of the slot. Each parent keeps track of the slots assigned to its children. If a slot is not assigned, the node will switch off its radio.

Since data collection in BailighPulse takes place in a short period of time, it is important to ensure that data is reliably collected from the whole network. As in the case of the *Wake-Up* state, a node uses information about the expected packet delivery. Whenever an expected packet from one of the children is not received, additional rounds are scheduled. If an expected packet is not received for a number of rounds, the parent assumes that the child node is dead and removes it from its children list.

In order to decide if an additional round is required, a node maintains a Remaining Round Count (RRC) field for each associated child. The field indicates how many additional rounds are required by a given child. At the beginning of the first collection round the RRC field for each associated child is set to RRC_0. This ensures that a parent node will wait for reception of the packet for at least RRC_0 rounds. At the end of the collection round, the value of the RRC field of each child node is decreased by 1. If the value of the field for at least one child node is greater then 0, an additional round is scheduled. Each data packet, also contains the RRC field. A node which has more data to send in its queue, sets its value to RRC_0. Otherwise, the value of the RCC field is set to 0. Upon receiving the packet, the parent saves the value of the RCC field.

A parent informs its child if he is able to accept another packet by setting an appropriate flag in the acknowledgment. Hence, an additional round is scheduled when a parent's message queue becomes full. Note, that nodes which do not participate in additional rounds are excluded from communication (e.g. node D in 5), and enter the *Inactive* state.

5.3 Maintenance

Low power wireless links are often susceptible to quality fluctuations. Brief degradation of link quality can be caused by interference from other wireless devices operating in the same frequency band such as WiFi or Bluetooth [29]. Permanent faults can arise due to node failure or obstacles which block wireless transmission.

To improve robustness to link variation, BailighPulse uses maintenance slots. A maintenance slot is a short contention slot between consecutive collection rounds (see Figure 5). To reduce energy consumption, LPL is used during the slot. The maintenance slot allows for wake-up recovery and parent switches.

A brief degradation of link quality might prevent a child node from receiving a wake-up pulse from its parent. The parent node can infer that the child node did not receive the wake-up pulse if a packet expected from the child is not received during the *Collection* phase. This is possible since each parent maintains a record of the application schedules of its children. If this happens, the parent assumes that the child missed the previous wake-up pulse and sends an additional one during the maintenance slot. A node which is not synchronized estimates the start and end of the maintenance slot based on the previous collection time. If a node does not receive a wake-up pulse from its parent for $RRC_0 - 1$ maintenance slots then it is assumed that the link is down.

In this case a node changes its parent node. To reduce the cost of rescheduling the tree by invoking *Initialization*, each node maintains a list of potential parents, which are at the same level in the tree. The list is created by overhearing wake-up pluses during the *Wake-Up* phase. The parent switch depends on the tree building algorithm. Typically it involves exchange of handshake packets which establish the parent - child association.

6 Duty Cycle Analysis

In this section we provide an energy analysis of BailighPulse and show how the information from the schedule is used to optimize energy consumption. We analyze the energy consumption of the protocol in terms of Duty Cycle (DC). DC is determined by the radio usage of a protocol. In general, lower DC means a more energy efficient protocol. We use a multi-hop traffic model which assumes a spanning tree based on shortest-path routing to the sink (Figure 6). The model used herein provides lower bounds on the energy performance of BailighPulse. It does not take into consideration a number of factors, suchas interference, collisions, and protocol organization.

We place N nodes in an area A. Each node has a radio range of radius r. The area A is divided into k annuli, which represent the distance (hop count) from the sink. The sink is placed in the center of area A. Each node belongs to one and only one annuli. It is assumed that nodes are positioned within the annuli according to a uniform random distribution. The distance, between the inner and outer radii of an annulus is equal to the node

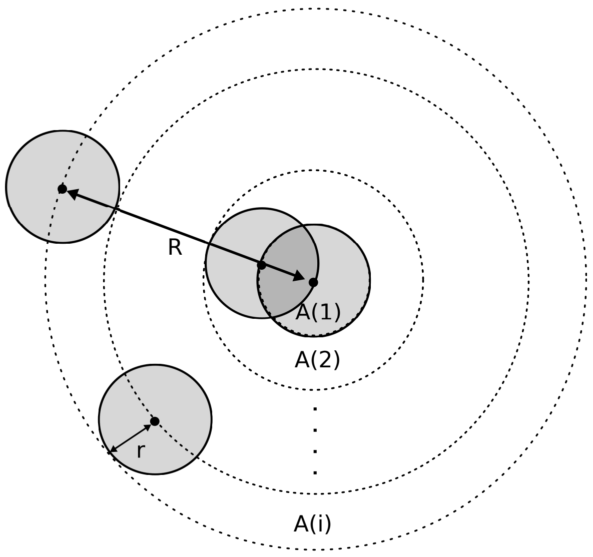


Fig. 6: Network model. We use Unit Disk with radius r to model transmission range.

transmission range r. As stated in Bulusu et al [30], network density μ, can be defined as $\mu = \frac{N\pi r^2}{A}$, where r is radius of node's radio range and A is the field area. The radius R of area A can be derived as $R = r\sqrt{\frac{N}{\mu}}$.

The network diameter $D(N)$ can be calculated as the average number of nodes which fit in the radius R (see Figure 6), i.e., $D(N) = \left\lceil \frac{R}{r} \right\rceil = \left\lceil \sqrt{\frac{N}{\mu}} \right\rceil$. We take the ceiling value of the result to ensure an integer value for the network diameter: The number of nodes at each hop can be calculated by dividing the area of each annuli by the radio range of a node. Let us define i^{th} annuli radius as $R(i) = ir$ and its area as $A(i) = A_r(i) - A_r(i-1) = \pi r^2(2i-1)$. The number of nodes at annuli i can be calculated as follows:

$$C(i) = \begin{cases} 1 & \text{if} \, i = 0 \\ \frac{\mu \cdot A(i)}{\pi r^2} = \mu(2i-1) & \text{if } i \in (1,\, D(N)) \\ N - \sum_{j=1}^{i-1} C(j) & \text{if} \, i = D(N) \end{cases} \tag{2}$$

In order to calculate a node's DC we need to know the number of messages forwarded through that node. A node at a given hop forwards messages from nodes placed at further hops. In addition, it transmits its own messages. Nodes without children do not forward messages. Let us denote the average number of messages forwarded by a node at level i as $M(i)$ and the number of messages produced by a leaf node as M_0. The value of $M(i)$ can be represented as follows:

$$M(i) = \begin{cases} 0 & if\, i = 0 \vee i = D(N) \\ \frac{\sum_{j=i}^{D(N)} C(j)}{C(i)} & \text{otherwise} \end{cases} \tag{3}$$

Let us define an average DC for a node which uses a protocol p as $DC_p(M_{in}, M_{out})$. The function DC_p depends on number of messages than node receives M_{in} and transmits M_{out}. In order to the calculate the average

DC for a network of N nodes using protocol p, we sum DC at each level (4) and calculate the average (5):

$$\begin{aligned} DC_{sum}(p,N) &= \sum_{i=1}^{D(N)-1} (DC_p(M(i),\, M(i)+M_0)\cdot C(i)) + \\ &+DC_p(0,\, M_0)\cdot C(k) \qquad (4) \\ DC_{avg}(p,N) &= \frac{100}{N} DC_{sum} \qquad (5) \end{aligned}$$

In BailighPulse collection takes place every T_{cp}. The protocol does not transmit any data between consecutive collections, therefore the guard time takes into consideration the drift developed since last data collection $t_{guard} = 4T_{cp}r_{skew}$. After waking up for data collection a node enters the wake-up phase and performs LPL only during the guard time.

We assume that drift between nodes has normal distribution therefore on average each node must poll for half of the guard time, hence

$$DC_{poll} = \frac{t_{guard}\cdot t_{poll}}{2T_{cp}T_{poll}} \qquad (6)$$

The collection in BailighPulse is performed in rounds. The number of rounds , $R = \frac{M_{out}}{P_c}$, depends on number of packets M_{out} which need to be forwarded andthe number of packets P_c which can be sent in a round.

A node receives data during wake-up and collection state. During the wake-up state, a node turns its radio on, which takes t_{wkp} seconds. After that it receives, on average, half of the preamble and a wake-up beacon t_{bcn}. While in the Collection state, a node turns on its radio in each collection round. In total, it receives M_{in} data packets from its children.

$$DC_{rx} = \frac{t_{wkp} + 0.5T_{poll} + t_{bcn}}{T_{cp}} + \frac{t_{wkp}\cdot R + t_{pkt}\cdot M_{in}}{T_{cp}} \qquad (7)$$

Transmission Duty Cycle DC_{tx} is similar to reception duty cycle DC_{rx} with the difference that a whole preamble must be transmitted:

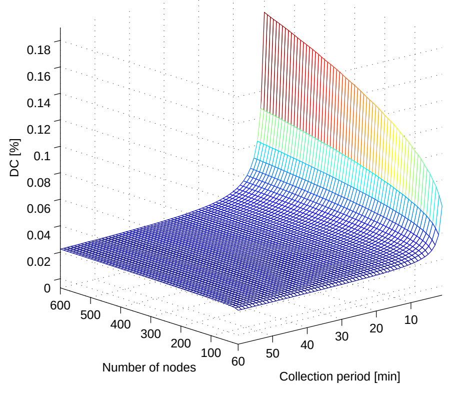
$$DC_{tx} = \frac{t_{wkp} + t_{bcn} + T_{poll}}{T_{cp}} + \frac{t_{wkp}\cdot R + t_{pkt}\cdot M_{out}}{T_{cp}} \qquad (8)$$

Overall the duty cycle of BailighPulse DC_{BP} is equal to:

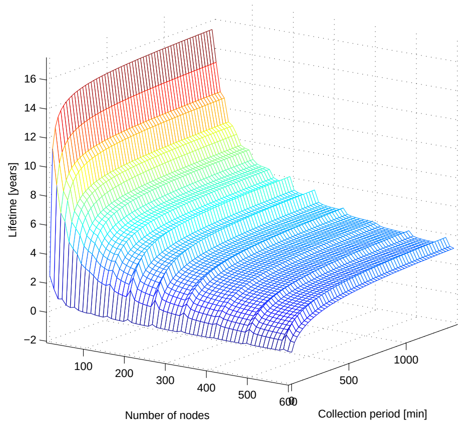
$$DC_{BP} = DC_{poll} + DC_{rx} + DC_{tx} \qquad (9)$$

We apply a polling period optimization scheme similar to the one proposed in [1]. Polling during the synchronization phase of BailighPulse is optimized with respect to the protocol duty cycle:

$$T^*_{poll} = \frac{\delta DC_{BP}}{\delta T_{poll}} = \sqrt{\frac{4}{3}T_{cp}r_{skew}t_{poll}} \qquad (10)$$



(a)



(b)

Fig. 7: Node's average duty cycle (a) and lifetime (b)

The optimal polling period T^*_{poll} cannot be less than the physical time required by a radio to poll the channel t_{poll}. Hence $T_{cp} > \frac{3}{4}\frac{t_{poll}}{r_{skew}}$ must hold. Assuming $t_{poll} = 2.5ms$ and $r_{skew} = 100ppm$, the communication period must exceed $T_{cp} > 18s$.

Figure 7 shows the average duty cycle of a node and and its estimated lifetime as a function of network size and collection period. In can be seen that increasing collection period above 8 hours does not bring significant benefit. This is because the power consumption is dominated by the sleep mode of the node.

7 Performance evaluation

A simplified analytical model cannot capture the complex interaction between nodes. Therefore, we implemented a detailed model of BailighPulse in OMNeT++ [31] . This section describes the simulation model, performance metrics, simulation scenarios, and parameters used in the simulation.

7.1 Simulation setup

We simulated BailighPulse using 10, 25, and 50 randomly deployed nodes. In the first two scenarios the simulation area was 35×35 m^2, whereas in the last one the area was 65×65 m^2. In each case, the sink was placed in the center of the simulation area. We simulated BailighPulse running both homogeneous and heterogeneous schedules.

Simulations were performed using the Castalia [32] framework for OMENeT++. The framework provides a wireless channel model which uses the Log-Normal Shadowing Model [33] for wireless signal propagation. The model takes into consideration the effects of wireless signal fading and shadowing. These effects, common in wireless transmissions, are modeled by adding a perturbation factor to the reception power. This factor follows a normal distribution, with standard deviation σ which can be defined for each simulation run. In addition, the asymmetry of links is modeled. To capture the effects of wireless interference, the simulation uses an physical (additive) interference model [34]. In this model, reception probability is determined by signal-to-noise ratio. The sum of power from multiple concurrent transmissions may cause interference at a given node, even though

Table 2: Simulation parameters.

Parameter	Value
P_{tx}^*	58.5 mW
P_{rx}^*	65.4 mW
P_{sleep}^*	0.015 mW
P_{poll}^*	14.1 mW
t_{poll}	2.5 ms
t_{cca}	2 ms
t_{wkp}	2 ms
Data Rate	250 kbps

Parameter	BailighPulse	Dozer	B-MAC
Retransmission #	3		
Slot Count	5	5	-
Packets / Slot	4	4	-
Buffer Size	20		
Clock drift	100 ppm		
Random Jitter	-	750 ms	-
Initial Backoff	-	-	20 ms
Packet Size	48 bytes		

Parameter	Value
P_{tx}	0 dbm
PL(d_0)	55
d_0	1
n	2.48
σ	4
RRC_0	3
Slot duration	5ms
Maintenance slot	25ms

*Includes microcontroller power consumption.

separately each of them is below the receiver sensitivity. In addition, the wireless signal propagation model takes into consideration factors such as transceiver modulation type (PSK[1]) and sensitivity to calculate bit error rate.

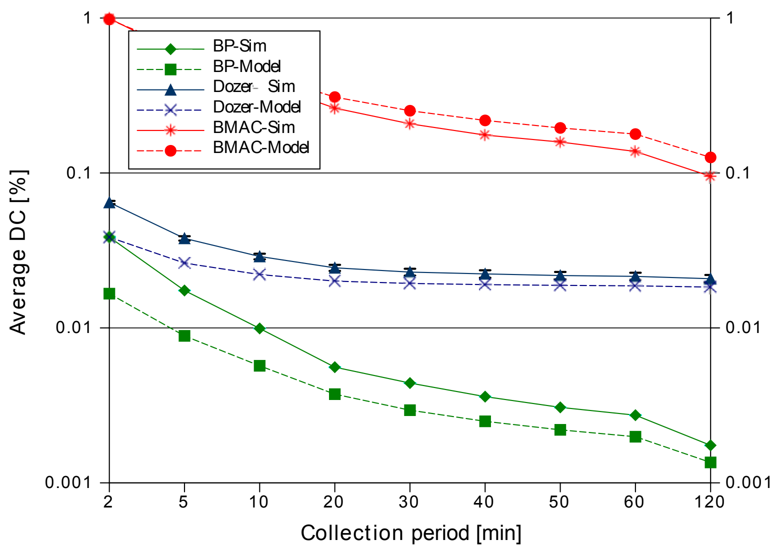
A model of the Chipcon CC2420 [35] transceiver, provided by Castalia, was used in the simulation. The transceiver is modeled as a finite state machine consisting of tree states: sleep, receive, and transmit. Delays in transition between respective states were modeled, which allows for precise calculation of the duty cycle and energy consumption.

Castalia also provides a generic model of asynchronous MACs. The model provided by Castalia lacks features such as packetized preamble and packet acknowledgments. As these features are required for both BailighPulse and B-MAC we extended Castalia by implemented the missing features.

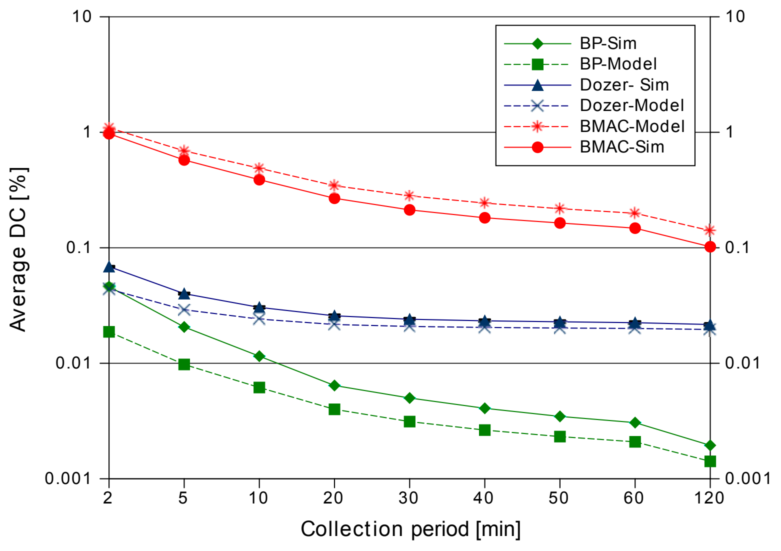
We compared BailighPulse's performance with that of Dozer [22] and B-MAC [15] combined with Shortest Path routing. We implemented the relevant features of Dozer i.e., parent-child slot based scheduling, periodic beaconing with a random jitter, and packet exchange with acknowledgments. We use Dozer as a reference state-of-art data gathering protocol. The model of B-MAC used in simulation includes collision avoidance, packetized preamble, and link layer acknowledgments. We compare BailighPulse with B-MAC combined with the shortest path routing to asses its performance against the simplest energy efficient data collection method.

For each scenario, we generated 10 different random topologies and for each topology the system performed 100 data collection cycles. The topologies were generated using a separate script with the same propagation model as Castalia. All topologies were generated such that each node in the network had at least one neighbour with high quality link, i.e., with PRR greater than 80%. In the case of BailighPulse, collision free slots were assigned withing a one hop neighbourhood.

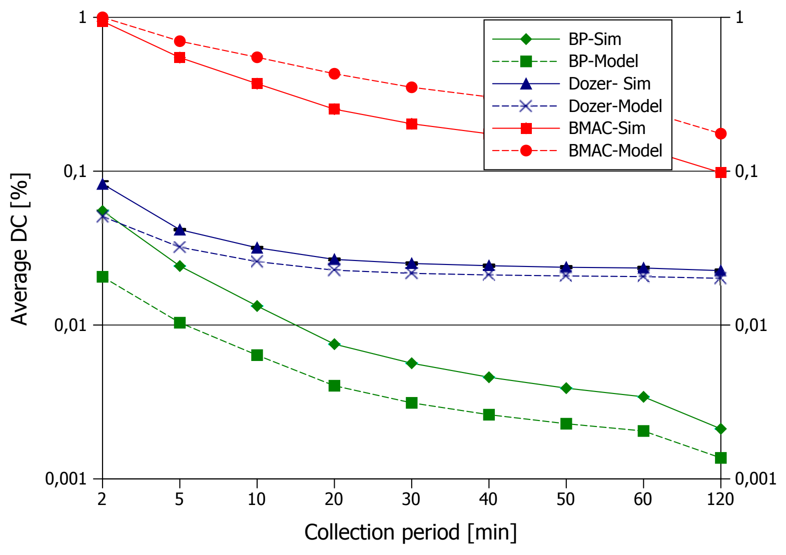
The values shown in the results are averaged over all independent runs. The error bars reflect the 95% confidence interval. Unless otherwise stated, the transceiver and protocol parameters in Table 2 were used. *Duty cycle* was used as a metric to indicate the energy efficiency of the protocol. *Delivery rate* was used as a metric to indicate the percentage of messages which were successfully delivered to the sink. Latency was not used as a metric since it is not a crucial metric for *mostly-off sensor networks.*



(a) 10 nodes



(b) 25 nodes



(c) 50 nodes

Fig. 8: Average DC as function of collection period (homogeneous schedule).

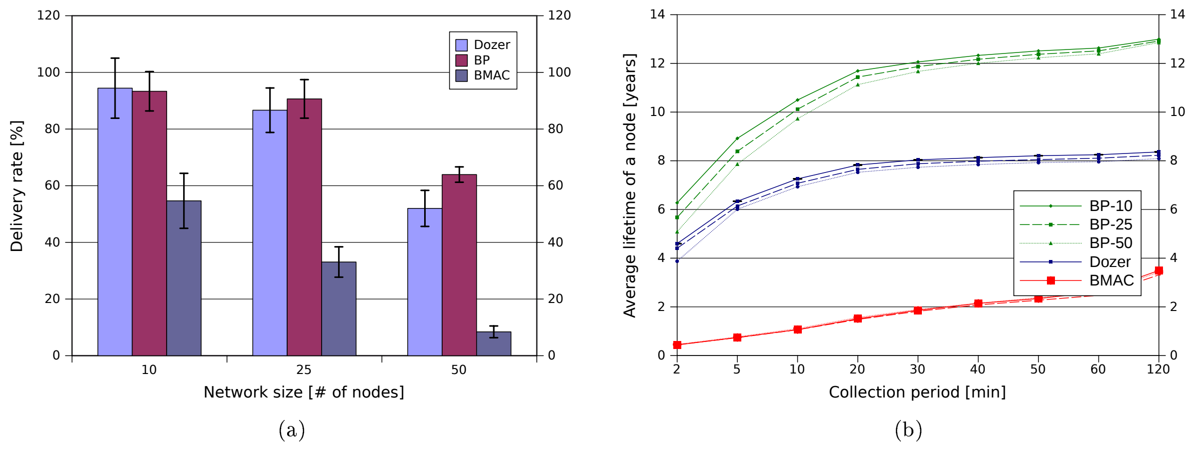


Fig. 9: Delivery rate ($T_{cp} = 5\text{min}$) (a) and estimate of an average lifetime of a node (b).

7.2 Results

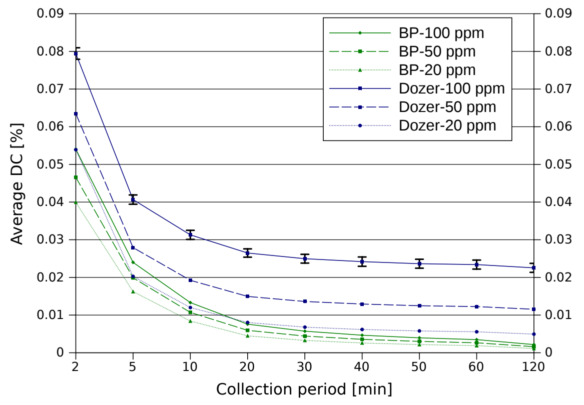
7.2.1 Homogeneous schedule

The first set of simulations were performed for a network using a homogeneous schedule. In this scenario, all nodes in the network have the same collection period which was varied from 5 to 120 min. For reference, we include the results from the analytical model in the following figures. Figure 8 compares the duty cycle for BailighPulse, Dozer, and B-MAC for various network sizes. As expected, in each case the average duty cycle of BailighPulse drops as the collection period is increased. This is mainly due to the energy efficient wake-up which minimizes the duty cycle with respect to collection period. The impact of clock drift on the duty cycle is particularly visible for Dozer, as DC does not decrease as the collection period is increased. This is because as the collection period grows, each node must include a longer guard time in order to receive its parent's beacon. For the network of 50 nodes BailighPulse reduces DC by 30% in the worst case ($T_{cp} = 2min$), and by 90% in the best case ($T_{cp} = 120min$). Note, that DC reduction for a network of 25 and 50 nodes, starting with collection period of 10 min, drops only slightly (c.a., 5%) which shows that BailighPulse scales well with network size.

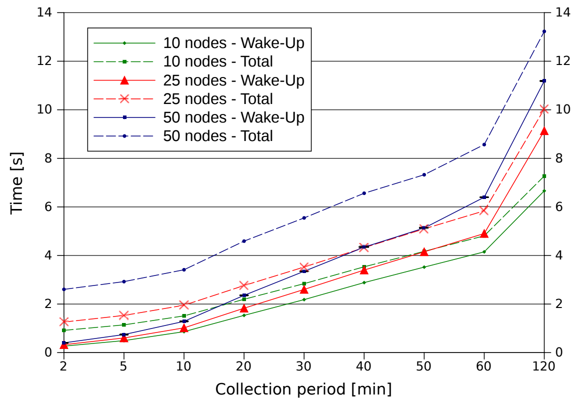
The average error between the simulation and analytic model Δ_{avg} is greater than 20% in all presented cases. We attribute this to the fact that the model does not take into consideration packet loss due to signal variability and collisions. For BailighPulse the average error in all cases is $\Delta_{avg} = 40\%$. The main factor in the error is packet loss due to signal variability. Lost packets must be retransmitted during additional rounds which increases the duty cycle. The impact of collisions is particularly visible for B-MAC. In a 10 node network the average error between model and simulation is 18% ($\Delta_{min} = 8\%$, $\Delta_{max} = 32\%$), whereas for the network of 50 nodes the average error $\Delta_{avg} = 60\%$ ($\Delta_{min} = 44\%$, $\Delta_{max} = 79\%$). With Dozer, which distributes transmission over beacon period and uses randomized jitter to avoid transmission alignment, the average error $\Delta_{avg} = 20\%$ is similar for all cases. This confirms that, as indicated by the authors, Dozer is less impacted by collisions.

Figure 9a shows a comparison of the delivery rate. As the delivery rate does not depend on the collection period, we present results for $T_{cp} = 5\text{min}$. Both BailighPulse and Dozer have a delivery rate close to 90% for

[1] PSK was used as Castalia does not support QPSK used by CC2420



(a)



(b)

Fig. 10: Impact of crystal accuracy on duty cycle (a) and nodes wake-up and collection time (b)

a network of 10 nodes. As the network size grows, the delivery rate of both protocols drops. For a network of 25 nodes Dozer shows better performance than BailighPulse by 10%. However, for a network of 50 nodes, the delivery rate of Dozer drops. This is because the density of the network is larger and collisions between beacons as well as data packets are more possible. BailighPulse schedules transmission during both wake-up and collection. Due to the use of individual transmission slots, the number of collisions is reduced. Note, that in the simulation nodes send data at the same time, hence the very poor performance of B-MAC. The high contention, caused by multiple senders accessing the channel at the same time, causes B-MAC to drop large number of packets. For the network of 50 nodes the delivery rate of B-MAC drops to 10%.

In Figure 9bestimates of average node lifetime are shown. The calculation was done using duty cycle values measured in simulation and parameters presented in Table 2. We assumed that a node is powered by a lithium CR2450 coin battery with capacity of 600mAh. Using this power source a node is able to operate for 8 years with T_{cp} = 5min to 13 years with T_{cp} = 120min.

Since BailighPulse is designed to operate in harsh environments where clock drift may vary considerably between nodes, we assess the impact of clock accuracy on the average duty cycle. We exclude B-MAC from this analysis as its duty cycle does not depend on synchronization accuracy. The results presented in Figure 10a show the average duty cycle of a node as a function of collection period using clock accuracies of 20ppm, 50ppm, 100ppm. Deteriorating accuracy from 20ppm to 100ppm results in an increase in the Duty Cycle by 1.4x for T_{cp} = 5min and by 1.8x for T_{cp} = 60min. For the same case, the duty cycle of Dozer increases by 2x for T_{cp} = 5min and by 4.2x for T_{cp} = 60min. This is due to the fact that Dozer uses idle listening during the guard time. As BailighPulse takes into consideration the accuracy of the clock to calculate polling period during wake-up phase, the impact of clock inaccuracy is minimized.

In Figure 10b, we show time required to wake-up the network and collect one data packet from all nodes is shown. A short time of operation is important for BailighPulse performance at it uses staggered collection schedule. As can be seen the time depends mainly on the collection period. This is due to the fact, that longer wake-up slots must be used to mitigate for clock drift. For example, for a network of 25 nodes, the wake-up takes 0.6s and 9s for collection period T_{cp} = 5min and T_{cp} = 120min, respectively. Network size has an impact

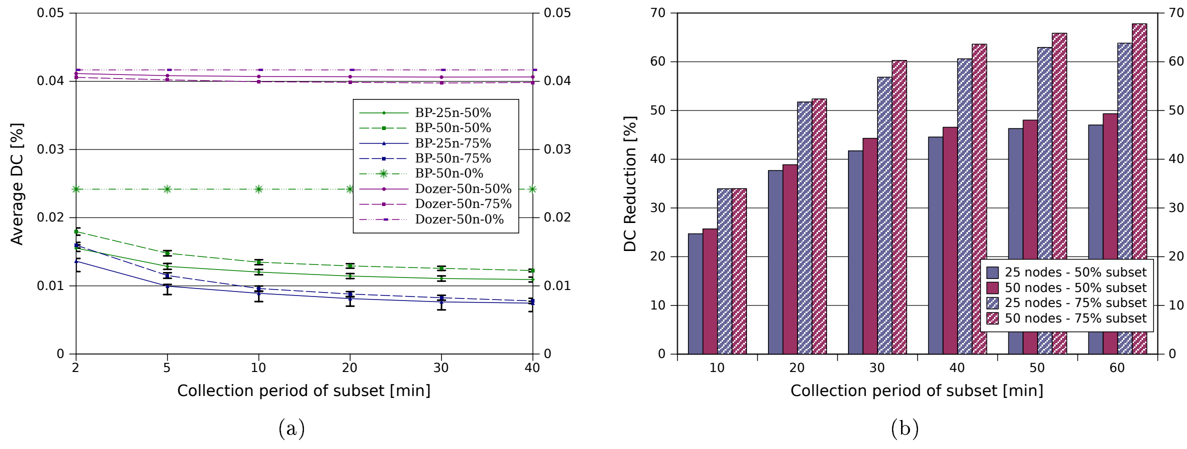


Fig. 11: Duty cycle comparison (heterogeneous schedule)

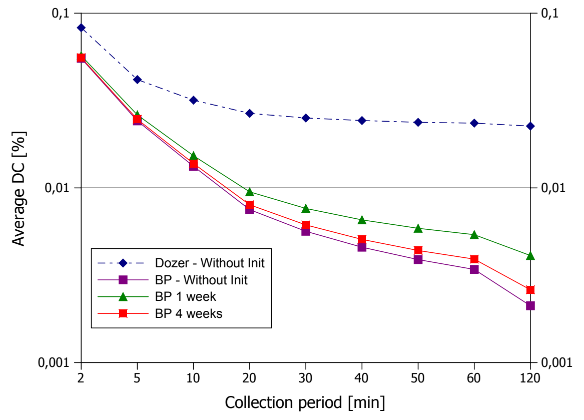
on collection time, however, this is of less significance since slots during collection are short. For example, the collection time for the network of 25 nodes is about 0.9s, whereas for the network of 50 nodes the collection time increases to about 2s. Note, that collection time does not depend on the collection period T_{cp}.

7.2.2 Heterogeneous schedule

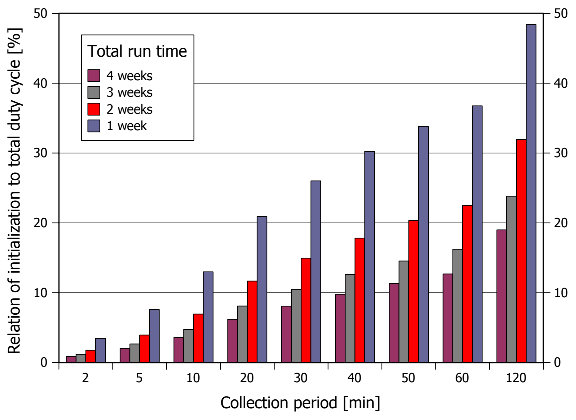
In the heterogeneous scenario, we divided the network into two sets. Set S_1 uses a collection period $T^1_{cp} = 5$min, whereas the collection period T^2_{cp} of set S_2 was varied from 10 to 60 min. The size of S_2 was varied between 50% and 75% of all nodes. The nodes in set S_2 were selected from leaf nodes, to ensure that maximum energy saving is achieved. For clarity we exclude B-MAC from this analysis. For the same reason we include results for Dozer only for network of 50 nodes. Figure 11a shows the duty cycle comparison of BailighPulse and Dozer. The results confirm that BailighPulse takes advantage of heterogeneous schedules. The duty cycle of BailighPulse drops as the collection period of S_2 is increased. This is unlike Dozer, which maintains almost a fixed duty cycle, due to broadcasting beacons at a fixed rate.

Figure 11b shows the duty cycle reduction of BailighPulse with relation to a whole network running a homogeneous schedule. The results shows how well BailighPulse is able to exploit schedule heterogeneity to reduce energy consumption. In the worst case, BailighPulse is able to reduce the duty cycle by 25% ($T^2_{cp} = 10\,min, |S_2| = 50\%$), and in the best case by 68% ($T^2_{cp} = 60\,min, |S_2| = 75\%$). Note, that reductions in Duty Cycle are similar for networks of 25 and 50 nodes ($\Delta_{max} = 5\%$), which shows that BailighPulse scales with network size when running heterogeneous schedules.

In is important to emphasize that the reduction in the duty cycle in the heterogeneous scenario depends on network topology. This is due to the fact that nodes running low rate schedules might be forwarding nodes for nodes running high rate schedules. In this case, the schedule of the forwarding nodes would have to match the high rate schedule.



(a) Comparison of protocols duty cycle including the cost of initialization



(b) Relation of initialization phase duty cycle to overall duty cycle

Fig. 12

No interference	WiFi Radio	WiFi Download	Bluetooth
DR: 100% DC: 0.013%	DR: 98% DC:0.014%	DR: 0% DC: 0.05%	DR:94% DC:0.018%

Table 3: BailighPulse performance when exposed to interference.

7.2.3 Impact of the Initialization Phase

BailighPulse does not prescribe how the network is initialized during the Initialization Phase. Hence, the results discussed in the previous sections focus on Duty Cycle in the Wake-Up and Collection phases. It is important, however, to analyze the impact of initialization on the overall duty cycle of the protocol. In earlier work we proposed a network initialization protocol called TrickleTree [28]. The protocol performs network discovery, collision free schedule assignment, and association of nodes. TrickleTree allows for initialization of a network of 50 nodes in 70 seconds with a duty cycle of 12%.

Obviously, the impact of the Initialization Phase depends on its frequency, duration, and Duty Cycle during operation. Herein we assume that the initialization phase is run only when the network must be completely reinitialized. This is necessary, for example, when a large number of nodes is removed or added to the network. In other cases, we expect that network maintenance performed during the maintenance slot is sufficient.

Figure 12a shows a comparison of the duty cycle of BailighPulse with TrickleTree and Dozer. It is assumed that TrickleTree is run at the start of an operational phase. The time of the operational phase is varied for 1 to 4 weeks. The Dozer Duty Cycle does not include the cost of protocol initialization. As expected, initialization increases the overall duty cycle, but the overall duty cycle is still lower than that the Duty Cycle of Dozer. The impact of initialization is clearly seen in Figure 12b. As the collection period grows the duty cycle of collection drops, and the impact of initialization becomes more significant. The impact can be reduced by running initialization as rarely as possible.

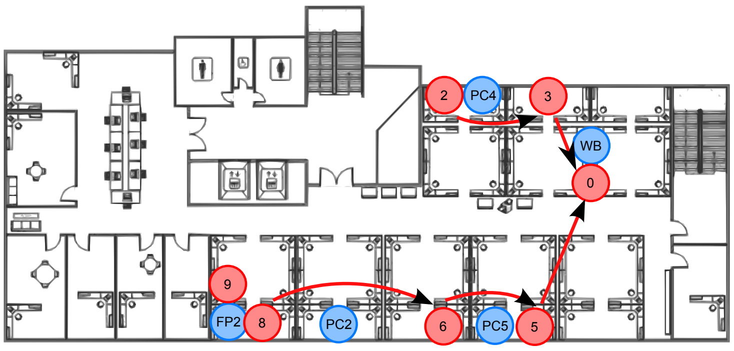


Fig. 13: Deployment map

7.2.4 Impact of Radio Interference

Typically WSNs operate in the 2.4GHz ISM band which is also used by WiFi and Bluetooth. As shown in [29] interference from these networks can have a severe impact on the protocol delivery rate. BailighPulse robustness against interference from WiFi and Bluetooth was tested using the empirical models of WiFi and Bluetooth interference provided in [29]. As in [29] a scenario with one receiver, one transmitter, and one interference node was modelled. The receiver and transmite run BailighPulse, while the inferferer generated an interference according to a selected model. The WiFi interference pattern was generated assuming a 802.11b transmitter, while the Bluetooth interference pattern was generated assuming 802.15.1 ver. 2.0. It was assumed that the interferring node continously transmits 1 Kb long packet. The transmitter node sent 100 packets, each 30 bytes long, every 120 s. The retransmission limit was set to 3.

Table 3 presents results of the simulation. The results show that BailighPulse is robust to non saturated WiFi traffic as well as to Bluetooth interference. In the case of Bluetooth interference the delivery rate is slightly lower. As the Bluetooth transceiver data rate is lower that of a WiFi transceiver, transmission of a packets occupies the channel for longer. Hence, the probability of interfering with BailighPulse increases. The experiment shows that BailighPulse is not robust to saturated traffic when the interfering node occupies channel for most of the time. However, in real deployment it is unlikely that saturated traffic is present at all times. Typically the channel is occupied for a certain period of time (e.g., tens of minutes). In this case the data can be delivered at the next collection period.

8 Case Study

An example application for BailighPulse is as part of a Heat Cost Allocation (HCA) system. Heat Cost Allocation systems are used in centrally heated buildings to divide energy cost among individual apartments on the basis of heat use [36]. To account for heat consumption, electronic devices are used to measure the temperature of individual radiators installed in each apartment. Usually, temperature measurements are taken every few minutes and data collection takes place every few hours [37]. Development of these systems is particularly challenging due to the very high and unpredictable clock drift caused by variations in radiator temperature.

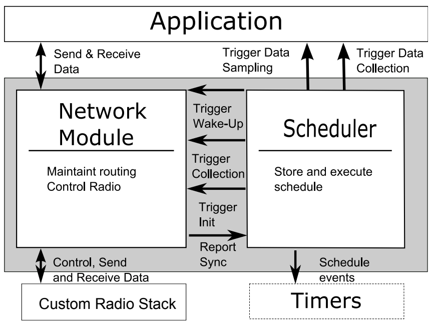


Fig. 14

8.1 Implementation and setup

The software for the proposed system was implemented using the TinyOS 2.1.1 [19] operating system. An overview of a node's software implementation is shown in Figure 14. The main component is the *Scheduler*. The *Scheduler* is responsible for executing schedules and has full control over events initiated in a node. The *Scheduler* ensures that no other component performs actions until requested to do so. In general, the *Scheduler* requests the *Network* module to enter one of four states: *Initialization, Wake-up, Collection, or Inactive.*

The default LPL network stack was modified to allow for the very short polling periods required by BailighPulse. The CC2420 radio is a packet based radio. This means that there is a short gap between transmission of consecutive packets. According to our measurements the gap is about 2.5 ms wide. In the default stack this is addressed by performing a number (400 by default ~ 12 ms) of channel checks which cover the gap. This strategy cannot be used in BailighPulse since BailighPulse uses very short polling periods in the range 5 ms to 20 ms. To address this issue, the stack was modified to use the CC2420 continuous transmission test mode (TXMODE=2). In this mode, the radio continuously transmits data present in its FIFO buffer. The modified stack fills the buffer with several copies of a packet, calculates CRC, and enables the test mode for a time equal to the polling period. Packets are continuously updated with the current timestamp (as well as CRC) required for proper time synchronization between nodes. Once the set time elapses, the radio is switched back to regular mode. The implementation takes 40 kB of ROM and 3 kB of RAM.

The experiments took place in an open floor of an office building. During the experiment 7 TelosB sensor nodes were used. The locations of the nodes in the experiment are shown in Figure 13. All nodes except node 0 and 9 were attached to a radiator at 2/3 of the radiator height. The sensor node was attached, so that the temperature sensor faced the radiator surface. Node 0 was designated as the sink, whereas node 9 was designated to measure ambient temperature near node 8. Both nodes were placed at the top of a cubicle approximately 1.5m from the floor.

The radio duty cycle was measured by recording in software the total radio on and off times. We do not differentiate between transmission and reception time. During the experiment 100 collection rounds were completed and in each collection round, a node generated one data packet. The duty cycle of the sink was also measured to account for scenarios where the sink is battery powered since it could simply be a more powerful node with a GSM transceiver.

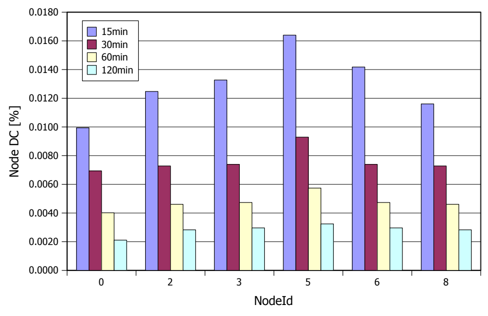


Fig. 15: Measured duty cycle of particular nodes

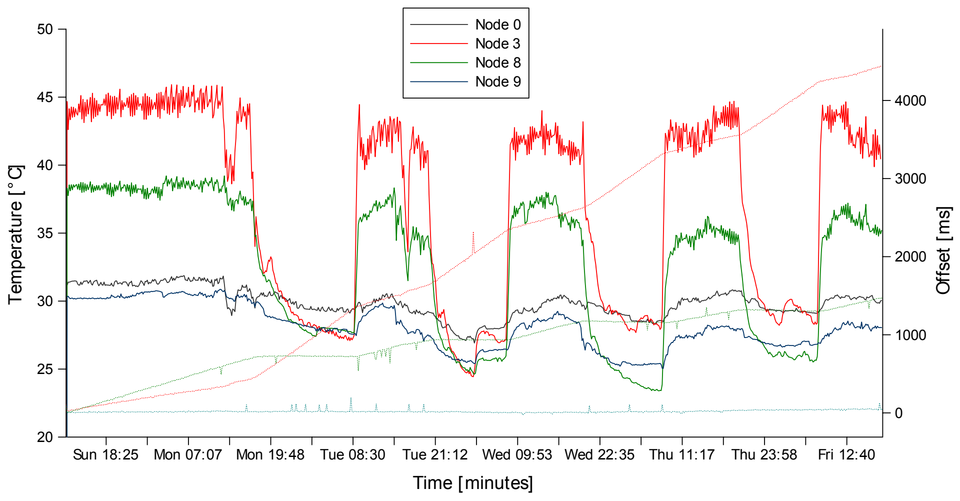


Fig. 16: Clock drift change caused by temperature

8.2 Results

Figure 15 shows the duty cycle of various nodes in the network. It can be seen that the duty cycle depends on a node's position in the network. For example, node 5 has the highest DC since it forwards data from nodes 6 and 8. Nodes 2 and 8 have the lowest duty cycle since these are leaf nodes. Node 0 does not upload messages. Hence its DC is lower that of node 5. The average duty cycles for all nodes were 0.0130%, 0.0076%, 0.0047%, 0.0028% for 15, 30, 60, 120 minute collection periods respectively. The simulation results for a network of 10 nodes are 0.0092%, 0.0044%, 0.0027%, 0.0017% for the respective collection periods. In general, the simulation results are about 40% lower than the testbed results. We attribute this to the fact that DC measurements in the testbed were performed in software. The simulation does not account for this effect. In all runs the delivery rate was above 99%.

During evaluation, we were also interested in the impact of temperature changes on clock drift. Figure 16 shows the result of an experiment in which the clock drifts between nodes 0, 3, 8, and 9 were measured. In order to establish the time difference, each mote was connected to a PC. Time synchronization between the PCs was maintained using the Precision Time Protocol [38]. The impact of temperature variation on clock drift can be clearly seen. The relative clock drift between nodes changes when the radiator is switched on. After 24 hours, the maximum drift reached almost 1.4s. Interestingly, high temperature increased the drift of node 8, but decreased the drift of node 6 presumably due to crystal cut angle variation. This confirms that temperature variation may cause unpredictable clock drift, hence the BailighPulse design assumptions were correct.

9 Conclusion

Herein, we propose BailighPulse, an energy efficient data gathering protocol for mostly-off WSN applications. BailighPulse incorporates a novel multi-hop wake-up scheme that allows for energy efficient recovery of network synchronization after long off periods. The scheme uses a staggered wake-up schedule and optimized channel polling during wake-up, based on knowledge of the pre-defined application-level schedule. An analytic model of BailighPulse is presented and lower bounds on performance established.A complex simulation model was implemented in OMNeT++ for evaluation of the protocol. The proposed protocol was compared with Dozer, a state of the art data gathering protocol, and B-MAC, an implementation of Low Power Listening technique. Simulation results show up to 80% reduction in duty cycle for homogeneous and 68% for heterogeneous schedules respectively.

References

1. Y. Li, W. Ye, J. Heidemann, R. Kulkarni, Design and evaluation of network reconfiguration protocols for mostly-off sensor networks, Ad Hoc Networks 6 (2008) 1301–1315.
2. N. Ramanathan, M. Yarvis, J. Chhabra, N. Kushalnagar, L. Krishnamurthy, D. Estrin, A stream-oriented power management protocol for low duty cycle sensor network applications, in: EmNets '05: Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors, pp. 53–61.
3. I. Lie, V. Tiponut, I. Bogdanov, S. Ionel, C. D. Caleanu, Automated meter reading system for heat costs allocation, WSEAS Trans. Cir. and Sys. 8 (2009) 177–186.
4. I. Talzi, A. Hasler, S. Gruber, C. Tschudin, PermaSense: investigating permafrost with a WSN in the Swiss Alps, in: Proceedings of the 4th workshop on Embedded networked sensors (EmNets), IEEE Computer Society, 2007, pp. 8–12.
5. P. Dutta, D. Culler, S. Shenker, Procrastination might lead to a longer and more useful life, in: Proceedings of HotNets-VI, Atlanta, GA, November, pp. 1–7.
6. T. Schmid, R. Shea, Z. Charbiwala, J. Friedman, M. B. Srivastava, Y. H. Cho, On the interaction of clocks, power, and synchronization in duty-cycled embedded sensor nodes, ACM Trans. Sen. Netw. 7 (2010) 24:1–24:19.
7. G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, M. Parlange, SensorScope: Out-of-the-Box Environmental Monitoring, in: Proc. of IEEE IPSN, IEEE Computer Society, 2008, pp. 332–343.
8. M. Miller, N. Vaidya, A mac protocol to reduce sensor network energy consumption using a wakeup radio, Mobile Computing, IEEE Transactions on 4 (2005) 228 – 242.

9. L. Yann-Ael, B. Gianluca, Round robin cycle for predictions in wireless sensor networks, in: Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005. Proceedings of the 2005 International Conference on, pp. 253 – 258.
10. S. Yoon, C. Shahabi, The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks, ACM Trans. Sen. Netw. 3 (2007).
11. J. C. Lim, C. Bleakley, Extending the lifetime of sensor networks using prediction and scheduling, in: Intelligent Sensors, Sensor Networks and Information Processing, 2008. ISSNIP 2008. International Conference on, pp. 563 –568.
12. B. Pazand, A. Datta, An energy-efficient node-scheduling scheme for wireless sensor networks based on minimum dominating sets, Int. J. Netw. Manag. 19 (2009) 75–99.
13. L. Wang, Y. Xiao, A survey of energy-efficient scheduling mechanisms in sensor networks, Mobile Networks and Applications (2006).
14. G. Anastasi, M. Conti, M. D. Francesco, A. Passarella, Energy conservation in wireless sensor networks: A survey, Ad Hoc Networks 7 (2009) 537 – 568.
15. J. Polastre, J. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in: Proc. of ACM SenSys, ACM, 2004, pp. 95–107.
16. A. El-Hoiydi, J.-D. Decotignie, Low power downlink mac protocols for infrastructure wireless sensor networks, Mob. Netw. Appl. 10 (2005) 675–690.
17. M. Buettner, G. V. Yee, E. Anderson, R. Han, X-MAC: a short preamble mac protocol for duty-cycled wireless sensor networks, in: Proc. of ACM SenSys, ACM, 2006, pp. 307–320.
18. O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, P. Levis, Collection tree protocol, in: Proc. of ACM SenSys, ACM, 2009, pp. 1–14.
19. P. A. Levis, TinyOS: An Open Operating System for Wireless Sensor Networks, in: IEEE International Conference on Mobile Data Management, volume 0, IEEE Computer Society, Los Alamitos, CA, USA, 2006, p. 63.
20. U. Colesanti, S. Santini, A. Vitaletti, DISSense: An adaptive ultralow-power communication protocol for wireless sensor networks, in: International Conference on Distributed Computing in Sensor Systems and Workshops, pp. 1 – 10.
21. R. Musaloiu-E., C.-J. M. Liang, A. Terzis, Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks, in: IPSN '08: Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008), pp. 421–432.
22. N. Burri, P. von Rickenbach, R. Wattenhofer, Dozer: ultra-low power data gathering in sensor networks, in: Proc. of IEEE IPSN, IEEE Computer Society, 2007, pp. 450–459.
23. W. Ye, J. Heidemann, D. Estrin, Medium access control with coordinated adaptive sleeping for wireless sensor networks, IEEE/ACM Trans. Netw. 12 (2004) 493–506.
24. I. Rhee, A. Warrier, M. Aia, J. Min, M. L. Sichitiu, Z-MAC: a hybrid MAC for wireless sensor networks, IEEE/ACM Trans. Netw. 16 (2008) 511–524.

25. T. van Dam, K. Langendoen, An adaptive energy-efficient mac protocol for wireless sensor networks, in: Proc. of ACM SenSys, ACM, 2003, pp. 171–180.
26. C. E. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers, SIGCOMM Comput. Commun. Rev. 24 (1994) 234–244.
27. W. Bober, C. Bleakley, Bailigh: Low power cross-layer data gathering protocol for Wireless Sensor Networks, in: Proc. of ICUMT, pp. 1–7.
28. W. Bober, C. J. Bleakley, X. Li, Trickletree: A gossiping approach to fast and collision free staggered scheduling, International Journal On Advances in Networks and Services 1 & 2 (2011).
29. C. Boano, T. Voigt, C. Noda, K. Romer, M. Zuniga, Jamlab: Augmenting sensornet testbeds with realistic and controlled interference generation, in: Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on, pp. 175 –186.
30. N. Bulusu, D. Estrin, L. Girod, J. Heidemann, Scalable coordination for wireless sensor networks: Self-configuring localization systems, in: Proceedings of the International Symposium on Communication Theory and Applications (ISCTA), IEEE Computer Society, 2001, pp. 24–60.
31. A. Varga, OMNeT++, http://www.omnetpp.org, Last accessed: 04/2010.
32. National ICT Australia, Castalia - Simulator for Wireless Sensor Networks and Body Area Networks, http://castalia.npc.nicta.com.au, Last accessed: 04/2010.
33. T. Rappaport, Wireless Communications: Principles and Practice, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.
34. A. Iyer, C. Rosenberg, A. Karnik, What is the right model for wireless channel interference?, in: QShine '06: Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks, ACM, New York, NY, USA, 2006, p. 2.
35. CC2420 Data Sheet, http://www.ti.com, Last accessed: 04/2010.
36. Y. Yao, S. Liu, Z. Lian, Key technologies on heating/cooling cost allocation in multifamily housing, Energy and Buildings 40 (2008) 689 – 696.
37. BrunataNet, http://brunata.com/products/net-systems/brunatanet, Last accessed: 07/2011.
38. K. Correll, N. Barendt, Design Considerations for Software Only Implementations of the IEEE 1588 Precision Time Protocol, in: Conference on IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, IEEE Instrumentation and Measurement Society, 2006, pp. 2–7.