Mobile Netw Appl (2011) 16:460–474
DOI 10.1007/s11036-011-0326-2

An Automatic Presence Service for Low Duty-Cycled Mobile Sensor Networks

Shouwen Lai · Binoy Ravindran

Published online: 18 June 2011

Abstract We consider providing presence service for duty-cycled wireless sensor networks through a multi-hop approach. The presence service is to ensure automatic network monitoring by which each node would know whether the sink node is reachable or not. Towards providing such presence service, we tackle three problems: 1) efficient neighbor discovery due to not-always-awake nature of duty-cycling and the mobile environment, 2) light presence message passing from the sink node to all reachable nodes given broadcasting is expensive and difficult in an embedded duty-cycling network, and 3) automatic network monitoring if there is node failure and network partition. In our protocol, in order to save power consumption, an online node which is reachable from the sink node only book-keeps the broadcast schedule of its parent in a breadth-first-search spanning tree in order to trace the online status all along. The offline node which is not reachable from the sink node stays awake periodically based on quorum-based wakeup scheduling, and probes the beacons which may come from online nodes. The presence protocol can automatically detect link failure or network partition, and it can also automatically recover online status for each sensor node if there is a path to the sink node, which is significant for applications that are sensitive to end-to-end latency constraints. The presence protocol proposed is implemented through a layered approach so that it is independent from any specific MAC and routing protocols. We make extensive simulations in order to validate the energy efficiency and reliability of our design.

Keywords wireless sensor networks · network mobility management · automatic network monitoring · duty cycling · quorum systems

S. Lai (✉)
Qualcomm Inc, 5775 Morehouse Dr, San Diego, CA 92121, USA
e-mail: shouwenl@qualcomm.com

B. Ravindran
ECE Department, Virginia Tech, Blacksburg, VA 24061, USA
e-mail: binoy@vt.edu

1 Introduction

The design and deployment of a mobile wireless sensor network (WSN) is driven by the requirements of specific applications, including target tracking, healthcare monitoring, environmental monitoring, habitat monitoring, etc, which vary both in terms of area size, deployment locations, and mobility of the nodes. As the number of applications grow, automatic network management services [1–4] are necessary to meet different application requirements in mobile WSNs.

In contrast to infra-structured networks such as the Internet, a network management system designed for mobile WSNs should provide controlling or monitoring service through an energy-efficient, reliable and localized approach. The network management services in mobile WSNs are wide-ranging, including fault detection [2], time synchronization [5], power management [4], security provision [6], administrations [1] etc. The services can be proactive, reactive, or passive depending on specific designs. Usually, developing a general purpose network management protocol is a big challenge and most network management protocols target specific services.

In this paper, we consider providing *presence service* in low duty-cycled mobile WSNs. The presence service [7] in the Internet is a service which accepts, stores and distributes information that conveys ability and willingness of a potential communication partner to communicate. Using a presence service, a sender knows the availability of its partner and the connection status of the network. Similarly, presence service in mobile WSNs can be used by a sensor node to decide whether the sink node is reachable, or whether an actuator in a mobile sensor and actuator network is available.

An energy-efficient presence service is significant for applications with real-time transmission [8] and power saving requirements. The service can reduce energy waste for data delivery. For example, by the presence service, if the sink node is not reachable due to link failure or network partition, the source sensor node can hold the transmission until the path is recovered, hence reducing unnecessary retransmission and routing discovery. In addition, the presence service can reduce latency of data delivery in case of link failures or network partitions via automatical recovery in advance.

In contrast to the Internet, there is no infrastructure such as proxy server or registration server in mobile WSNs. Thus, the presence service we propose for mobile WSNs is not implemented in the same way as in the Internet, but using an ad-hoc approach. The difficulties of providing such presence service in low duty-cycled mobile WSNs is due to the following reasons:

1. Neighbor discovery is not always guaranteed since nodes are not always awake; and
2. Presence message passing is expensive since broadcast is not available as nodes are not always awake and the resource is limited in embedded sensor nodes; and
3. There are radio collisions for broadcasting due to wireless nature, especially in a dense network; and
4. Node mobility and transient link failures may cause network partition.

In this paper, we design a presence protocol over low duty-cycled mobile WSNs, named as PPL. PPL provides following functionalities to tackle the problems mentioned above:

1. Efficient neighbor discovery; and
2. Light presence message passing from the sink to other nodes; and
3. Automatic detection of reachability to sink nodes; and
4. Automatic recovery from link failure and network partition.

In our design, each node that is reachable to the sink node is called online node. Online nodes will periodically broadcast beacon messages which carry online status with low frequency. The node which is not reachable to the sink node is referred as offline node. Offline nodes asynchronously stay awake and attempt to detect the broadcast messages. We utilize our previous work (cqs-pair [9]) for node wakeup scheduling and beacon broadcasting arrangement to reduce energy consumption as much as possible. We also propose an algorithm to avoid beacon collision. The algorithm builds collision-free schedules. Since each online node is broadcasting online status through beacons, in order to control the beacon flooding, we use a breadth-first-search (BFS) tree for message passing. A node determines its online status by checking whether it has received beacon messages from its parent in the BFS tree.

PPL is a service protocol and it is not designed for general data transmission. It can work with most MAC protocols developed for WSNs, like S-MAC [10] or B-MAC [11]. Basically, it can reside between MAC layer and network layer. The routing layer can utilize the information from PPL to check whether the sink node is reachable from a node. If not, the routing layer can make further decision, i.e., discovering a new route or waiting for the route recovery, which is based on routing protocol designs.

Our contributions are as follows:

1. We design energy-efficient and asynchronous neighbor discovery mechanism for mobile sensor nodes; the mechanism can reduce the number of beacon messages and the ratio of awake time to whole operation time for offline nodes; and
2. We propose light presence message passing mechanism through a multihop approach; and
3. We design automatic network management by constructing a BFS spanning tree over which an online node only book-keeps the status of its parent, in order to control flooding; and
4. We present a collision minimized beacon broadcasting algorithm for all online nodes.

To the best of our knowledge, PPL is the first work on providing multihop presence service in low duty-cycled mobile WSNs. Our simulation-based measurements validate the effectiveness and efficiency of our design.

The rest of paper is organized as follows: We discuss past and related works in Section 2. In Section 3, we state the assumptions and preliminary knowledge for our design. In Section 4, we present the principle of efficient neighbor discovery. In Section 5, we explain

how the automatic network management is working. We discuss some implementation issues in Section 6. Simulation results are discussed in Section 7. We conclude in Section 8.

2 Related work

The problem of providing presence service has not been well studied for mobile WSNs. We summarize related works including those on network management service, presence service, asynchronous neighbor discovery, and collision free transmission schedules.

Network management service in WSNs This provides a set of management functions that integrate configuration, operation, administration, security, and maintenance in a sensor network. Basically, it can be categorized as centralized and distributed.

In centralized systems, such as BOSS [1] and SNMS [3], the base station collects information from all nodes and controls the entire network. It is usually assumed that the base station has unlimited resources to perform complex management tasks and that the base station has the global knowledge of the entire network. The centralized approach has the scalability problem when applying to large-scale networks. Distributed management systems [4] employ multiple management entities which acts as a local manager. Each manager controls a subnetwork and may communicate with other manager stations distributively to perform management. Distributed management has lower communication costs comparing to centralized one, and provides better reliability. However, it may suffer from high system complexity for synchronization and collaboration.

Presence service This was first defined by IETF [7]. The applications include 3G IP Multimedia Subsystem (IMS)and instant messengers. A presence service system allows users to subscribe to each other and be notified of changes in states (i.e., online or offline). The presence protocol usually has an internal structure involving multiple servers and proxies. There may be complex patterns of redirection [7] and proxying, or direct communication among clients.

However, due to data-centric nature in WSNs, there is usually no communication between two arbitrary nodes. Shaila et al. [12] introduced a presence service for security key management, which is close to our work. However, they assumed a powerful mobile sink node which can move around the network to convey the presence, which is an one-hop presence service. In our work, we focus on providing presence service in a multihop way given the many-to-one communication nature of WSNs. And we implement the presence service with a multihop approach.

Asynchronous neighbor discovery This means that two low duty-cycled nodes do not need to synchronize their time clock before communication. This mechanism is based on the non-empty intersection properties of quorum system. It was first introduced in [13] where the protocol divides time into a sequence of beacon intervals which are grouped into sets of m^2 contiguous intervals arranged as a two-dimensional $m \times m$ array. A node arbitrarily selects one column and one row of entries, also defined as a grid quorum, to transmit and receive, respectively. This work was later extended by Zheng et al. [14] and Lai et al. [9].

Asynchronous neighbor discovery mechanism (i.e., [15]) have also been developed using the Chinese Remainder Theorem [16]. For example, if one node wakes up once every 3 time slots and another node wakes up once every five time slots, they will definitely wakeup simultaneously in every 3×5 time slots. The main limitations for such mechanism includes that the discovery latency is usually too long to satisfy real-time requirements. Also, it is often difficult to find out the relatively prime numbers for all adjacent nodes.

Collision-free transmission scheduling This is usually used for TDMA [17] mechanisms in wireless networks to ensure no collision between two neighbor nodes for data sending and receiving. It can be solved by graph coloring algorithms. In general, the colors could represent time slots or frequencies assigned to the nodes. Minimizing the maximum number of colors is very desirable in most cases, but is known to be NP-hard [18]. The distributed solution for the problem includes heuristic algorithms and randomized algorithms [19]. In our work, we do not address the exact same collision-free coloring problem, but the problem of minimizing the collisions, given a color palette.

3 Assumptions and objectives

3.1 Network model and assumptions

We model a mobile WSN as a directed graph $G = (V, E)$, with $|V|$ nodes and $|E|$ links. We assume that all nodes operate with duty cycle when there is no data for transmission or receiving, which means that a node is not always awake. We also use nodes and vertices interchangeably in the context of graph theory and WSNs.

We assume each node has the same fixed transmission range and an edge $(u, v) \in E$ exists if u and v are within the fixed transmission range of each other. Time is discrete and arranged as time slots. Since the medium of transmission is wireless, whenever a node transmits a message, all its neighbors which are awake hear the message. If two or more neighbors of a node n_i transmit at the same time, n_i will be unable to receive any of those messages. In this case we also say that n_i experiences collision. In any time slot, a node can either receive a message, experience collision, or transmit a message but cannot do more than one of these.

We assume that two nodes are reachable to each other when there is at least one path between the two nodes. Here we use the term "reachable" loosely, meaning that a topologically connected path in our context may not be connected at any time; instead, all nodes in the path are reachable from another node in the path within a finite amount of time.

We assume there is only one sink node in the network for our presentation. But our protocol can be easily extended to the scenario of multiple sink nodes. We also assume that a message arrives correctly within finite time from a sender to a receiver, which can be achieved by any MAC-layer transmission mechanism, like B-MAC [11] or S-MAC [10].

The time is not necessarily assumed to be synchronized. We further assume that all nodes have the same time frequency, or their clocks drift at relatively slow speeds(i.e., 100 us per second for oscillators with 1 MHZ frequency [20]).

3.2 Preliminaries and definitions

We use the following definitions for quorum systems which are used for wakeup scheduling. Given an integer n, let $U = \{0, \cdots, n-1\}$ be an universal set.

Definition 1 A *quorum system* $\mathcal{Q}$ under U is a superset of non-empty subsets of U, each called a quorum, which satisfies the intersection property: $\forall G, H \in \mathcal{Q}$: $G \cap H \neq \emptyset$.

Definition 2 A quorum system $\mathcal{Q}$ under U is said to have the *rotation closure property* if $\forall G, H \in \mathcal{Q}$, $i \in \{0, 1, ...n-1\}$: $G \cap (H + i) \neq \emptyset$, where $H + i = \{(x + i)\ mod\ n : x \in H\}$.

There are two widely used quorum systems, *cyclic quorum system* (cqs) and *grid quorum system* (gqs) [9, 21], that satisfy the *rotation closure property*. We mainly focus on cqs which usually can be defined as $C(A, \mathbb{Z}_n)$ which represents all rotations of quorum A over $\mathbb{Z}_n$.

An example of cqs is $\{\{1, 2, 4\}, \{2, 3, 5\} \cdots, \{7, 1, 3\}\}$ where each subset (i.e., $\{1, 2, 4\}$) is a quorum and it will intersect with its rotation. By applying cqs into wakeup scheduling, if two nodes wakeup on same quorum schedule from a cqs, like in the 1st, 2nd and 4th time slots in every seven consecutive slots, their wakeup period will overlap regardless of clock drift. The overlap can be utilized for neighbor discovery if a node sends out beacons in the beginning of a wakeup time slot.

Definition 3 (cyclic quorum system pair (cqs-pair)) Given two cyclic quorum $\mathcal{X} = C(A, \mathbb{Z}_N)$ and $\mathcal{Y} = C(B, \mathbb{Z}_M)$, suppose $N \leq M$. We call $(\mathcal{X}, \mathcal{Y})$ a *cqs-pair* if: $\forall (A + i)^p \subseteq \mathcal{X}^p$ and $(B + j) \subseteq \mathcal{Y}$, $(A + i)^p \cap (B + j) \neq \emptyset$ [9].

An example for cqs-pair is shown in Fig. 1: Let $A = \{1, 2, 4\}$ and $\mathcal{X} = C(A, \mathbb{Z}_7)$; $B = \{7, 9, 14, 15, 18\}$ and $\mathcal{Y} = C(B, \mathbb{Z}_{21})$. The pair $(\mathcal{X}, \mathcal{Y})$ is a cqs-pair. Also, both $(\mathcal{X}, \mathcal{X})$ and $(\mathcal{Y}, \mathcal{Y})$ are cqs-pairs. Cqs-pair can be used for heterogenous power saving for different nodes and meanwhile guarantees the neighbor discovery within bounded time.

Graph coloring is widely used for collision-free broadcast scheduling. The problem can be defined as follows,

Definition 4 (Distance-2 Vertex Coloring) Distance-2 Vertex Coloring (d2-coloring) is an assignment of colors to the vertices of the graph such that every vertex has a color and two vertices which are d2-neighbors of each other are not assigned the same color [22].

Vertices which are assigned the same color belong to the same color class, and nodes belonging to the same color class can broadcast messages simultaneously without any collisions.

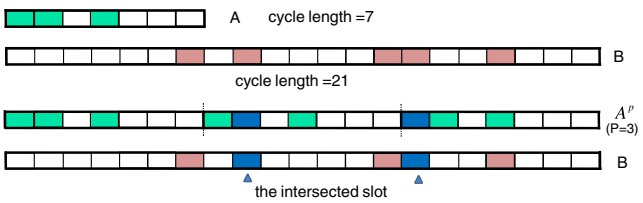

Fig. 1 Heterogenous rotation closure property between two cyclic quorum systems: A with cycle length of 7 and B with cycle length of 21. A quorum from A's p-extension A^p will overlap with a quorum from B

3.3 Objectives and problems

The primary objective by designing PPL in this paper is to detect the availability of a path to the sink node. Specifically, the PPL provides functionalities:

1. power-efficient neighbor discovery; and
2. automatic detection of reachability to sink nodes; and
3. automatic recovery from link failure and network partition.

Since we target to low duty-cycled mobile WSNs, the problems we need to solve in our design include:

1. asynchronous neighbor discovery between offline nodes and online nodes; and
2. efficient fault discovery and recovery with low message overhead or flooding control; and
3. minimizing collision for online beacons broadcasting in the network.

4 Neighbor discovery and energy efficiency

Since sensor nodes are operated in low duty-cycle fashion, in order to guarantee neighbor discovery within bounded time, quorum-based wakeup scheduling [14] or cqs-pair [9] is a good candidate for this. However, for online nodes, the energy efficiency is still not high using pure quorum-based wakeup schedules (i.e., ratio of awake time slots to time slots in a cycle is $\geq 3/7$ for a cyclic quorum system with cycle length of seven). To optimize the energy consumption for online nodes, we adopt sliding windows for book-keeping the incoming online broadcast beacons. Outside sliding windows, an online node may go to sleep in order to conserve energy consumption.

4.1 Wakeup modes

In PPL, there are two possible status for a node: *offline* and *online*, which are similar to that of the presence protocol for the Internet. Offline status means a node is not reachable to the sink node. Online status is in contrary. Accordingly, we design three wakeup modes: offline mode, transition mode, and online mode.

In *online mode*, a node only sends out beacon message at the beginning of time slots selected by a wakeup scheduling. In *offline mode*, a node will stay awake on time slots simply according to quorum-based schedule. *Transition mode* is a mode between online mode and offline mode. In such a mode, a node will wakeup based on quorum scheduling. The difference form offline mode is that there is one beacon message that is sent in the beginning of all wakeup time slots.

Consider a cqs-pair [9] $(C(A, \mathbb{Z}_N), C(B, \mathbb{Z}_M)(N \leq M)$, where $A = \{a_1, a_2, \cdots, a_p\}$ and $B = \{b_1, b_2, \cdots, b_q\}$, the wakeup scheduling and beacon broadcasting arrangement in PPL for the three modes are as follows.

Wakeup scheduling for offline mode a node stays wakeup in the $a_1^{th}, a_2^{th}, \cdots, a_p^{th}$ time slots for every N consecutive time slots (a frame).

Wakeup scheduling for transition mode a node stays wakeup in the $a_1^{th}, a_2^{th}, \cdots, a_p^{th}$ time slots for every N consecutive time slots (a frame). Meanwhile, a node sends out beacon messages in the beginning of all wakeup time slots.

Beacon broadcasting arrangement for online mode A node only broadcasts beacon messages in the beginning of the $b_1^{th}, b_2^{th}, \cdots, b_q^{th}$ time slots in every $L(L \geq M)$ consecutive time slots. In the remaining part of the selected slots, a node will return to sleep. Besides, an online node maintains sliding windows in order to trace the incoming beacon messages which carrying the presence status. We will introduce the sliding window in Section 4.2.

We call N or M as *quorum frame*. We refer L as to *super frame* since it is bigger than M. We choose $L \geq M$ for the purpose of energy conservation. We also refer to the beacon message in online mode as a *presence message*.

An example is shown in Fig. 2, where the quorum frame for offline and transition mode is 7 and the super frame for the online mode is 14.

Now we show the energy efficiency for all modes. We roughly use the ratio of awake time to whole operation time as energy consumption ratio (defined as the ratio of awake time to total running time) which is denoted as α thereafter. For offline mode and transition mode, we have

$$\alpha = \frac{p}{N} \tag{1}$$

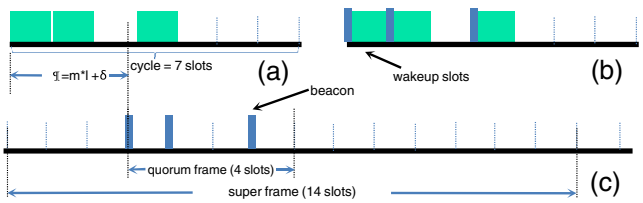

Fig. 2 Quorum-based wakeup modes: **a** offline mode; **b** transition mode; **c** online mode

For a cyclic quorum system (cqs) with cycle length of 7, $\alpha = 3/7 = 42.8\%$, and for a (cqs) with cycle length of 13, $\alpha = 4/13 = 30.7\%$.

For the online mode, suppose the ratio of duration for a beacon message and sliding window to one duration of one time slot is β (if the duration of a beacon message is 5 ms, the length of one sliding window is 10 ms and the slot length is 100 ms, $\beta = 15\%$), we have

$$\alpha = \frac{q}{L} * \beta \tag{2}$$

For the online mode shown in the Fig. 2, $\alpha = \frac{3}{14} * 15\% = 3.21\%$. If sliding window is not existing in the $b_1^{th}, b_2^{th}, \cdots, b_q^{th}$-th time slot, α is even lower.

Based on our design for PPL, the energy efficiency for online nodes is significant higher than offline nodes. This is because offline nodes should stay awake for more time to scan beacons message from any possible online node in neighborhood. Since we consider mobile networks, it is not possible to introduce time synchronization for offline nodes. As a result, it is difficult to improve the energy consumption ratio for offline nodes if we want to realize easy neighbor discovery.

In the following section, we will show the neighbor discovery property for the proposed wakeup modes.

4.2 Neighbor discovery between online nodes

PPL uses sliding windows for online nodes to bookkeep the online status from neighbor online nodes. Suppose the length of sliding window is T_{slide}, then during the T_{slide}, online nodes listen to the broadcast beacon message from their parents in the BFS spanning tree (described in Section 5.1) in order to trace the online status continually.

The starting position of sliding window is dynamically changing due to clock drift. For WSNs, the clock drift is usually slow such that the updating period for starting position can be quite long. Sommer and Wattenhofer [20] point out that the maximum drift is 100 us per second for oscillators with 1 MHZ frequency. Hence the maximum clock drift is 1 ms every 10 s. In PPL, the starting position of the sliding window will be updated by a node every time the beacon messages are received from the parent in the BFS spanning tree (Section 5.1) in order to compensate for clock drift.

Although a long sliding window can tolerate fairly large clock drifts, it may not be energy efficient. In our implementation, the size of the sliding window is decided by combination of the rate of clock drift and the duration of a beacon message: $T_{\text{slide}} = 2 * T_{\text{beacon}} + 2 * T_{\text{drift}}$. Here, T_{beacon} is the time duration for one beacon message. The typical value of T_{beacon} can be 5 ms. The value of T_{drift} is decided by the speed of clock drift and the duration of one super frame in the online mode. For example, if the maximum drift is 0.5 ms per second and the length of the super frame is 10 s, we have $T_{\text{drift}} = 10 * 0.5 = 5$ ms, and hence $T_{\text{slide}} = 20$ ms.

Suppose, initially an online node receives the beacon message at T_0 (when the node is still in transition mode), the initial starting position for this sliding window will be set to: $T_{\text{begin}} = T_0 - \frac{1}{2} * T_{\text{beacon}} - T_{\text{drift}}$, in order to make sure that the beacon message is in the middle of the sliding window. After initial step, the starting position is dynamically changed due to clock drift in order to keep the beacon message always in the middle of sliding window after adjustment.

With sliding window, an online node can always detect the online status conveyed by its parent in the BFS tree and is able to judge whether it is reachable by the sink node.

4.3 Neighbor discovery between offline nodes and online nodes

In PPL, we adopt cqs-pair [9] $(C(A, \mathbb{Z}_N), C(B, \mathbb{Z}_M)$ $(N \leq M)$ for wakeup schedules and beacon arrangement. Online nodes use $C(B, \mathbb{Z}_M$ and offline nodes use $C(A, \mathbb{Z}_N$. Based on non-empty intersection property of cqs-pair, we have,

Theorem 1 *A node in offline mode can hear at least one beacon message from the node in online mode within L time slots, regardless of clock drift and assuming they are neighbors to each other.*

Proof Suppose the quorum for the schedule of online mode is G (G can be a rotation of B) for node n_1, and the quorum adopted for the schedule of offline mode or transition mode is H (H can be a rotation of A)for node n_2. Also, suppose the duration of one time slot is I and the clock drift is Δ between n_1 and n_2, as shown in Fig. 2.

If the clock drift is $\Delta = m * I$, where m is a non-negative integer, then based on the non-empty intersection property of cqs-pair [9], node n_2 will hear the beacon message from the node n_1 in a time slot within M $(M \leq L)$ slots. If $M < L$, n_2 can detect beacon message from n_1 within L time slots.

Otherwise, suppose the clock drift is $\Delta = m * I + \delta$, where $0 < \delta < I$, and suppose the intersection slot is the a_k^{th} time slot for the case of $\Delta = m * I$. Then node n_1 will hear the beacon message from node n_2 in the a_k^{th} time slot. □

Lemma 1 *A node with offline mode will hear at least one beacon message from the node with transition mode, regardless of clock drift and supposing they are neighbors to each other.*

The proof for Lemma 1 is similar to that of Theorem 1 and we therefore omit it. Now, we introduce the reason why we design transition mode and the principle for neighbor discovery between nodes with transition mode and nodes with other modes.

4.4 Neighbor discovery by nodes with transition mode

We introduce transition mode in PPL because there is a time span for an offline node becoming an online node. When an offline node detects online beacon message from its neighbor, it does not necessarily mean that the offline node is able to reach the sink node. Due to mobility, the neighbor nodes are probably temporarily "reachable" to the sink. This means that the offline nodes should still stay in offline mode even if the receive beacon message sometimes.

In the implementation for PPL, there is a "time_stamp" field in the beacon message, and the "time_stamp" is set by sink node and its value is increased sequentially. If the received beacon messages have same "time_stamp" value, the node in transition mode should go back to offline mode. This can avoid a local loop from being formed by offline nodes. The state machine between transition mode and other modes is described in Section 6.2.

Lemma 2 *Nodes with transition mode will discover each other within N time slots if they are neighbors to each other.*

The proof for Lemma 1 is rooted from non-empty intersection property of cqs systems [9]. By this property, two quorums $G, H \in C(A, \mathbb{Z}_N)$ will have $G \cap H \neq \emptyset$. Thus, a node with wakeup schedule of G and another node with wakeup schedule of H have overlapped awake times. With the help of beacon message at the beginning of wakeup slots, two nodes in transition mode can discover each other within N time slots.

Lemma 3 *A node with transition mode will hear at least one beacon message from the node in online mode within L time slots, regardless of clock drift and supposing they are neighbors to each other.*

The proof for Lemma 3 is similar to that of Theorem 1 and we skip it.

5 Automatic network monitoring

5.1 BFS tree for schedules book-keeping

A node in online mode has to periodically check its online status by receiving presence messages from its neighbors. A straightforward way to do this is to book-keep the schedules of all its neighbors. However, a node has to wakeup more frequently to receive presence messages from its neighbor by doing so.

In our design, in order to reduce the listening frequency of an online node, we organize all online nodes with a tree structure. In the tree, an online node selects its parent from which presence messages will be received. In order to quickly maintain the tree and reduce the average response latency for all online nodes, we adopt a breadth-first-tree (BFS) spanning tree for book-keeping broadcast schedules.

We use self-stabilizing algorithm inspired by the work in [23], for the BFS spanning tree construction. When a node n_i receives a message from its neighbor n_j, n_i will set n_j as its parent if the following condition is satisfied:

$$level_i < level_j + 1 \;\&\&\; n_j \text{ is online}$$

The proof for the correctness of our algorithm is very similar to the self-stabilizing BFS tree algorithm in [23]. The extension in our work is the consideration of presence status. The time complexity is $O(D_{\max} * |V|)$, where $D_{\max}$ is the diameter of all online nodes. This is because, a node will take at most $D_{\max}$ rounds to determine its layer information and parent, and in each round, there are at most $|V|$ messages sent to a node. Since the layer information is contained in the beacon messages which are sent out continuously by neighbors in the online mode or the transition mode, we do not estimate the message complexity.

There is a possibility of looping paths when passing presence messages. Consider the following example. Suppose there are node n_a and its child n_b, and node n_c who is n_b's child. If node n_a does not get presence messages from its original parents, it will switch to the transition mode to receive broadcast beacons. If n_a receives messages from n_c, which is broadcasting beacons, indicating its online status, n_a will set n_c as its parent. Now, n_a, n_b, and n_c will follow a loop. However, the BFS spanning tree constructed by our algorithm will avoid such a possibility.

Lemma 4 *Loop-free: There will be no loops formed for presence message passing over BFS spanning trees.*

Lemma 4 is true since no node will select a node with lower level as its parent so that the loop is avoided. The loop free property is significant for maintaining the stability of PPL given the node mobility or link failure which results in a node updating its parent frequently.

5.2 Determination of presence status

Theorem 2 *A node which is reachable to the sink node will become online within bounded time.*

Proof We prove this by induction. Suppose the node is n_v. If n_v is in the first layer in the BFS tree, then its parent is the sink node. If n_v is offline initially, it can hear beacons from the sink node based on Theorem 1. If it is in transition mode, n_v will hear beacons from the sink node within L time slots based on Lemma 3. Assume the claim is true for n_v in the i^{th} layer.

Consider the case of n_v in the $(i+1)^{th}$ layer. There exists one node, say n_u, in the i^{th} layer and reachable to n_v, which is online based on the induction step. Irrespective of whether n_v is in offline or transition mode, it will receive beacon message from n_u and become online. The claim holds. □

Theorem 3 *An node which is not reachable to the sink node will not become online.*

Proof If the node n_v is not reachable to the sink node, its neighbor is either offline (case 1) or in transition mode (case 2). For case 1, n_v will stay offline since it cannot detect beacon messages.

For case 2, n_v can receive beacon message. However, the *time_stamp* field in the beacon message is kept unchanged. Thus n_v will stay in offline mode. □

Theorem 4 *Nodes in transition mode will leave the mode within bounded time.*

Based on Theorems 2 and 3, a node will eventually become either online or offline. The node cannot stay in transition mode for a long time. We introduce transition mode in PPL just in order to prevent local loop formed by offline nodes.

5.3 Online status recovery from link failures

There are many reasons which cause link failures, like node mobility, power off, out of battery or node failure. When there is link failure, an online node, denoted as n_v, can not receive beacon messages from its parent on the BFS tree anymore. In such case, the online node n_v will switch to transition mode and try to detect beacon messages again.

If there is another online node in the neighborhood, the node n_v can detect the online node within L time slots based on Lemma 3 and hence the online status of n_v can be recovered.

If there is no online node in the neighborhood, it means that there is network partition, n_v will go to offline mode based on Theorem 3. However, once an neighborhood node of n_v becomes online, n_v will switch to online node from Theorem 2.

Meanwhile, the recovery procedure couples with BFS tree adjustment automatically. The procedure of adjusting BFS tree is self-stabilizing, which is described in Section 5.1.

5.4 Collision minimization for beacon messages

A node in online mode will periodically broadcast beacon messages (or presence messages) in its super frame. The broadcast schedules of all online nodes in a hop-2 neighborhood may collide with each other if the broadcast happens in the same sliding window at the receiver side, as illustrated in Fig. 3. When collision occurs, it is possible that a node cannot get presence messages from other nodes and hence will misjudge its online status.

In Fig. 3, there is collision on only one sliding window and the receiver can still receive one or two beacon messages correctly. However, if node n_{v1} and node n_{v2} conflict with each other on all beacons, for example both of them start broadcasting in the first time slot of a super frame and there is no clock shift between their timing, then node n_u will lose the online status forever.

Our purpose for collision minimization is to ensure that any two nodes inside a 2-hop neighborhood have different beacon broadcast schedules. Therefore, none of them will send out all beacon messages in same

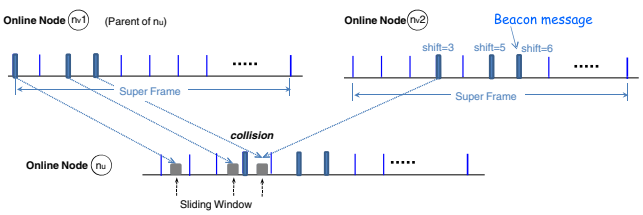

Fig. 3 Illustration to collision of beacon message

moment of time. To guarantee this, our algorithm in PPL contains two steps:

1. step 1, all online nodes within a 2-hop neighborhood distributively choose a different shift for their beacon broadcast schedules; and
2. step 2, all online nodes relatively synchronize with the sink node, and send out the beacon message based on the shift chosen in the first step by each online node;

The two steps can be coupled together in run-time execution.

We assume the time slot in which the sink node sends out the first beacons as TS_0. We now build up the connection between coloring and time slot shift.

Definition 5 (Color) A color is the time slot shift in term of integer slots. We assign the color 0 to the schedule in which the first beacon message is sent out in the first slot of every super frame, and color i to the the schedule in which the first beacon message is sent out in the i-th time slot in every super frame.

For a schedule with a super frame equal to L time slots, there are L colors. In Fig. 3, node n_{v1} has color 0, and node n_{v2} has the color 3 since its first beacon is shifted by 3 time slots.

The first step for collision minimization is to ensure each online node within a 2-hop neighborhood has different shift. It is done by Algorithm 1, which we will explain later.

Now, we show how to use the different colors we obtained in the first step to minimize collision. In the second step, we first do slot synchronization which makes all online nodes have TS_0 at relative same time moments. Each beacon message contains time slots shifting information related to TS_0. When nodes in the fist layer of the BFS tree receive the beacon from the sink node, they are automatically synchronized based on the shift information in the beacon message (we omit the propagation latency since we are not doing accurate time synchronization). Specifically, when node n_u receives beacon message which contains the time slot shift value TS_{received} (the distance to TS_0), node n_u is able to derive the position of TS_0: suppose the time slot when the beacon is received is TS_{curr}, the relative TS_0 position in a super frame is at the $(TS_{\text{curr}} - TS_{\text{received}})$ mod L-th slot. By doing this, all online nodes can be relatively synchronized after receiving beacon messages.

Therefore, if node n_u obtains a color TS_u (the time slot shift) from step 1, it should send out the first beacon message after

$$TS_\delta = (TS_u - TS_{\text{received}}) \mod L \tag{3}$$

time slots from the time of receiving the beacon message (TS_{curr}). If in step1, all online node can perfectly get different colors, then we can achieve collision-free beacon broadcast schedules.

Although the collision-free scheduling can be done by a d2-coloring algorithm which is defined in Definition 4, it may not be practical for us to achieve this. The first reason is that a mobile WSN usually does not have a global knowledge of maximum node degree (i.e., due to mobility), which will affect the maximum color number adopted. Secondly, the maximum number of colors is mapped to the length of the super frame in online mode. We cannot use an arbitrarily long super frame because it may result in a large latency in neighbor discovery.

In our design, we try to minimize the collision given a color palette (or the length of a super frame L). Let the duration of a time slot be T_s—i.e., $T_s = 100$ ms. For a cqs design via $(7, 3, 1)$-difference set, let the length of a super frame be $L * T_s$, where $L \geq 7$. Now, our algorithm for minimizing the color collision works as follows: A node n_u in the transition mode chooses a number TS_u from $\{0, \cdots, L-1\}$ such that there is minimum collision among the distance-2 neighborhood. Then node n_u will broadcast beacons in the TS_u-th time slot in a super frame once it is relatively synchronized.

The algorithm, described in Algorithm 1, is based on a randomized approach. In our algorithm, we denote the hop-2 neighbors of node v as $N_2(v)$, and the priority as P_v for node v. Each node v contains three parameters:

- layer: $layer(v)$;
- random value: $rnd(v)$;
- palette of forbidden colors which were used by its hop-2 neighbors: $usedcolor(v)$ (initially empty).

Each node also has an order based on its priority. Let $v_1, v_2 \in V$. We say that v_1 has a higher priority than v_2 (i.e., $P_{v_1} > P_{v_2}$) if:

$$layer(v_1) < layer(v_2) \; or$$
$$layer(v1) = layer(v2) \&\& (rnd(v_1) > rnd(v_2))$$

Algorithm 1: Collision Minimization Algorithm for all Online Node n_v:

```
1:  Initialization:
2:    usedcolor(v) = ∅;   S_c = {1, ··· , L};
3:  Sending Request message:
4:  if (v is uncolored) then
5:      choose parameter rnd(v) ∈ [0..1];
6:      if S_c − usedcolor(v) ≠ ∅ then
7:          Request(v).color = the first legal color
            L_v ∈ S_c − usedcolor(v);
8:      else
10:         Request(v).color = a random color
            L_v ∈ S_c − usedcolor(v)
11:     Request.layer = layer(v); Request(v).rand =
        rnd(v);
12:     Send Request(v) to all nodes in N2(v);
13: Sending Request(j) message from n_j:
14: if msg is Request then
15:     if (v is colored) then
16:         if Request(j).color ∈ usedcolor(v) ∩ color_v then
17:             send ACK-rej to n_j;
18:     else if P_v is the largest among that of ∀n_j ∈ N_2(v)
        then
19:         color_v = L_v;
20:         send ACK-rej to n_j ∈ N_2(v) with
            Request(j).color = L_v;
21:         send ACK-succ to n_j ∈ N_2(v) with
            Request(j).color ≠ L_v
22:     else if P_v < P_j then
23:         send ACK-succ to n_j;
24: else if msg is ACK-succ or ACK-rej then
25:     if Received ACK-succ from all nodes in N_2(v) then
26:         color_v = L_v;
27:     else
28:         usedcolor(v) = usedcolor(v) ∩ Request(j).color$
            when receiving ACK-rej from n_j;
29:         send Request message again;
```

In each round, every uncolored node v executes the following steps. An uncolored node n_v chooses the parameter $rnd(v) \in [0..1]$, and sends the following parameters to all its neighbors: $layer(v)$, $rnd(v)$, and the first legal color (which is not in the list of forbidden colors). Node n_v then compares its own parameters with that received from the hop-2 neighbors and checks which node has the highest priority. If node v has the highest priority, it will keep the proposed color and will stop. Otherwise, the node n_v updates the list $usedcolor(v)$. When the available palette is empty, then there is a collision for coloring. To minimize the collision, a node will randomly select a color from the color palette $\{0, \cdots, L-1\}$ in our algorithm.

Theorem 5 *The execution of Algorithm 1 results in a interference-minimized beacon transmission schedule.*

Proof

Case 1 The number of colors (length of a super cycle) is bigger than the number of nodes in a 2-hop neighborhood. No node can get a color unless there is a node with highest priority in its hop-2 neighborhood. In this case, at least two nodes with the lowest layer have equal priority. It is possible to neglect the probability of this situation by increasing the precision of tossing the parameter random value. Hence, in each round, in the whole graph, at most one new color can be assigned. After finite number of rounds, all nodes will be colored with an unique color, which is collision-free coloring.

Case 2 The number of colors (length of a super cycle) is less than the number of nodes in a 2-hop neighborhood. In each round, in the whole graph, at most one new color can be assigned if there is still available color. When there is not enough color, based on Line 10 in the Algorithm, a random color from $[1..L]$ will be selected for coloring a node with the highest priority. After finite rounds, all nodes will be colored and the collision is minimized. □

The number of rounds will be our measure of efficiency. The worst case of time complexity in Algorithm 1 is $O(n)$ based on Theorem 5 since in each round there is at least one node which is colored. The best case complexity of the algorithm is $O(D_{\max})$ where $D_{\max}$ is the the depth of the BFS tree.

6 Implementation

6.1 Protocol stacks

PPL can be implemented with a layered approach. The protocol provides presence service for applications and routing protocols in order to determine the path availability to the sink. It is independent of the underlying MAC protocols. The stack model which illustrates the relationship between the presence protocol and other layers is shown in Fig. 4.

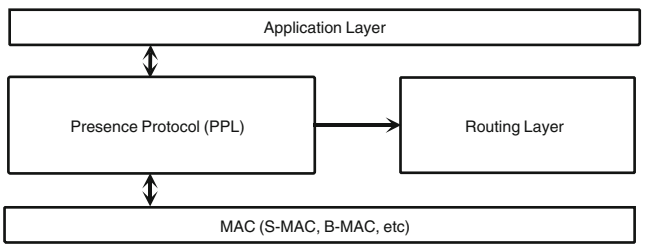

Fig. 4 Stacks model

PPL does not depend on accurate time synchronization [5] for all online nodes. But it is necessary for a node to book-keep the schedules of beacon broadcasts of its parent in the BFS spanning tree for collision minimization of beacon messages.

Since all beacons are sent out by broadcast, PPL does not rely on any specific routing protocol. The beacon message conveys the online status and layer information. Due to the hop-by-hop nature, PPL is independent of any specific MAC protocol. For one hop notification and passing color messages among nodes in the two-hop neighborhood, any MAC protocol, like S-MAC [10] or B-MAC [11] can provide a mechanism for data transmission.

Some MAC protocols [10] or time synchronization mechanisms [20] broadcast beacons periodically, which causes interfere with the beacons in PPL. However, such interference can be neglected if the super frame in online mode is large enough, e.g., $8-10$ s.

6.2 State machine

Now, we introduce the state machine for mode transitions in PPL as described in Fig. 5.

Initially, all nodes do not have presence information from the sink node and are in offline status. They operate in offline mode by asynchronous quorum-based wakeup scheduling. When a node detects presence information by receiving beacon broadcasts from its neighbor, it will first transfer to the transition mode in which the node will send out beacons in each wakeup time slot. In the transition mode, the node will synchronize with other presence nodes and will decide its color in order to determine which frame it should broadcast beacons by the coloring algorithm described in Section 5.4. The node will also decide its parent for book-keeping the beacon schedules, which been described in Section 5.1.

Once the coloring and the parent are set, a node will switch to the online mode and broadcast its presence status within beacon messages in every super frame.

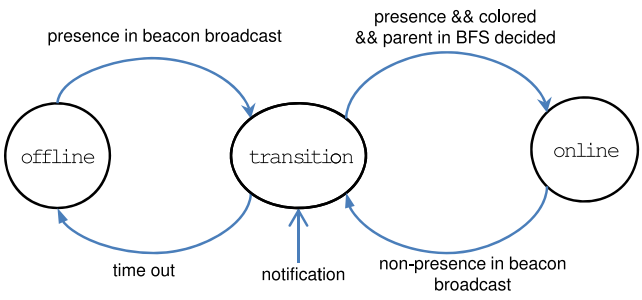

Fig. 5 State machine for wakeup mode transitions

Since a node should book-keep the broadcast schedule of its parents, if the node can continuously receive beacon messages from its parent at every super frame, the node will keep staying in the online mode.

If a node does not receive a presence message from its parent after timeout, the node will switch from the online mode to the transition mode.

In the transition mode, if a node still does not receive any presence message from its neighbor after a time out, the node will switch to the offline mode, which indicates the total offline status. The configuration of timeout is application-dependent and can be set to several super frame durations.

One additional message in the state machine is for notification. When a mobile node moves to a network and detects presence messages, the node will select a node as its parent after coloring, and will then send notification to its neighbors about its online status in order to update their parents in the BFS spanning tree for schedule keeping. After receiving the notification, a node will change from the online mode to the transition mode in order to reselect its parent and colors.

6.3 Beacon formats

In our implementation, the beacon message is send out via broadcast by each online node. Also, a node in the transition mode also sends out beacon messages. The beacon broadcast is significant for each nodes to trace its current presence status, i.e., online or offline.

The beacon message contains four fields:{*indic, node_id, layer, time_stamp*}. The field of *indic* has two types of value: indic=0 (indicating that the node is in the offline mode), indic = 1 (online mode). *node_id* is used to distinguish different nodes. The value of *layer* indicates the depth information of nodes in the BFS spanning tree. The field of *time_stamp* is used to identify whether two beacon messages are actually identical. Only the sink node can set the value of *time_stamp*.

When a node receives beacon messages, it will compare the "time_stamp" value to the previous received beacon message if there is one. If the "time_stamp" value is not changed, the node will not be switched to online mode. This can avoid a local loop formed by offline nodes.

7 Performance evaluation

We evaluated the performance of PPL through extensive simulations using the OMNET++ simulator. We measured the energy consumption ratio which is the ratio of awake time to total running time,

Table 1 Network size and time slot

	G_1	G_2	G_3	G_4
$\|V\|$	50	100	200	400
	T_1	T_2	T_3	T_4
T_{slot}	50	100	150	200

time and message cost, and the recovery time after node failure to validate the energy efficiency. The configurations of our simulations were compatible with typical configurations such as in [11]. We adopt the wireless loss model used in [24] which considers the oscillation of radio. The wireless communication range was set to 10 meters. The node mobility model selected in our simulation is based on Random Waypoint [25, 26] which was used by the community of mobile networks. Each sensor node moved with a velocity between 1 and 2 meters per second towards random directions.

We generated 4 network size sets with varying network sizes, $G_1, \cdots, G_4$, which are listed in Table 1. For each network size, we randomly generated 10 topologies. Each data point presented in this section is the average of 10 topologies with 10 runs on each topology. The length of time slots in our simulation was varied from 50 to 200 ms as listed in Table 1.

7.1 Energy consumptions and neighbor discovery latency

In this section, we first demonstrate the energy efficiency of quorum-based wakeup scheduling and then show the trade-off between energy efficiency and average neighbor discovery latency in PPL. As the processor consumes an extremely small amount of energy in comparison with the radio, we only evaluated the awake time of radio in our simulation, rather than measuring real energy consumption.

We set the size of beacon message as 160 bytes and the transmission rate as 256 kb/s. The network size is set by G_2 in Table 1. The size of sliding window for per beacon is set to 20 ms to book-keep the broadcast schedule of parents for an online node. We defined the energy consumption ratio as wakeup time to the whole operation time in our simulation, and measured the energy consumption ratio for nodes at online mode and at offline mode separately. From Fig. 6, by changing the cqs cycle length, it is shown that the cqs with larger cycle length is more energy efficient. The energy consumption ratio of offline mode with the cqs based on $(21, 5, 1)$-difference set is about 23.8% comparing 43.8% for $(7, 3, 1)$-difference set. Accordingly, there is

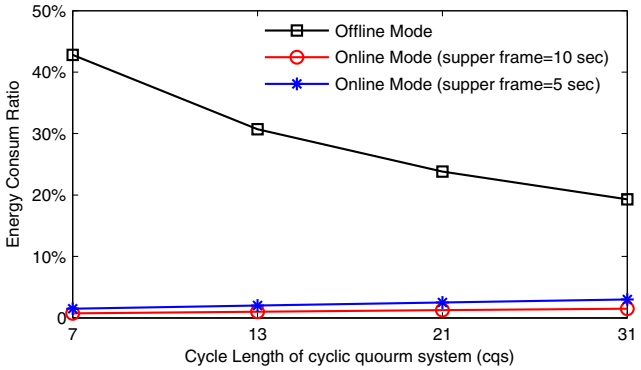

Fig. 6 Impact of cycle length of cqs on energy consumption

similar trend for online mode since a cqs with larger cycle length has a larger quorum size.

We also measured the average neighbor discovery latency between a node at offline mode and a node at transition mode, in order to show the performance tradeoff of quorum-based wakeup scheduling. Time slot length was varied from 50 to 200 ms. In Fig. 7, it is shown that larger time slot length resulted in longer neighbor discovery latency. Also the average neighbor discovery latency increased with larger cycle length, which is opposite to the changing trend of energy consumption ratio.

7.2 Validation of collision minimization algorithms

We made comparisons between PPL for collision minimization with previous collision-free distance-2 coloring algorithms, such as DRAND [17] and D2color [22]. As the main purpose of PPL is to minimize the collision given a number of colors, it needs less number of colors for coloring all nodes in a network, but suffers coloring collisions which do not exist in DRAND and D2color.

Color number Figure 8 shows the maximum color number for the three algorithms under different d2-

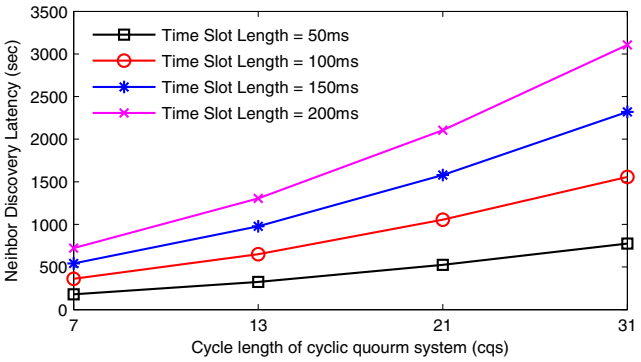

Fig. 7 Impact of time slot length on neighbor discovery latency

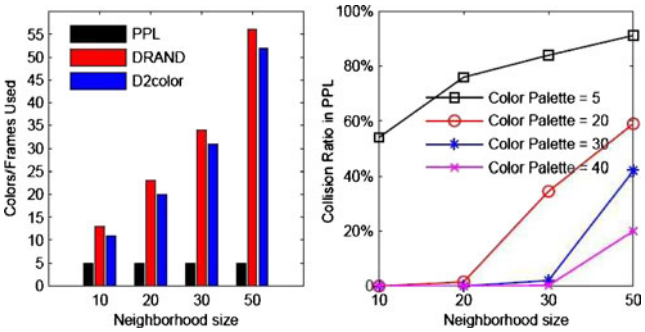

Fig. 8 Comparison of number of colors and collision rate

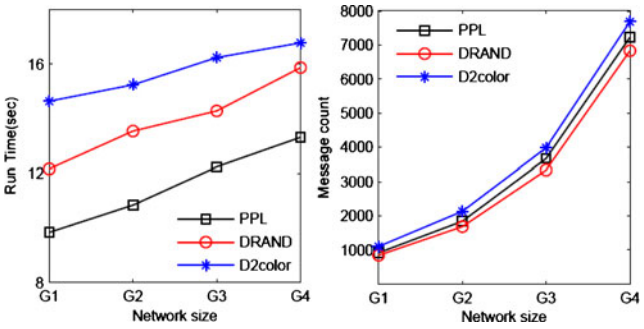

Fig. 9 Comparison of time and message complexities

neighborhood configurations. We varied the maximum size of d2-neighborhood from 10 to 50 and fixed the length of time slot as 100 ms. Since PPL uses constant size of color palette, its color number does not change very much. It was observed that the color number needed by DRAND had the largest value since it uses an asynchronous distributed algorithms.

Collision rate We also measured the collision rate of PPL when the color palette is less than the maximum size of d2-neighborhood. The collision rate in our simulation was defined as:

$$collisionrate = \frac{\#nodesselectinganoccupiedcolor}{\#totalnodesinthenetwork}$$

Since, DRAND and D2color are collision-free coloring solutions, their collision rate is zero. From Fig. 8, it is shown that the collision rate decreased with larger color palette and smaller size of maximum d2-neighborhood. When the color palette is small, the collision rate is relatively high (i.e. $\geq$80%) for a large d2-neighborhood.

Time and message complexity We varied the network size and observed the average time and message costs. Since PPL does not pursue a collision free solution, it has the lowest time complexity. DRAND can achieve better performance than D2color with the price of more color used to achieve collision free coloring when compared with D2color. Regarding message complexities, the DRAND can get better message count because it uses asynchronous message passing mechanism. In PPL, nodes have to wait for the broadcast messages from the sink node before determining its final color. Therefore the message complexity of PPL is lightly higher than DRAND since more messages should be broadcast as what is described in Section 5.4.

Figure 9 shows the comparison of time and message cost between PPL, DRAND and D2color. We measured the average time cost and the message costs with varied network sizes. Since PPL does not pursue a collision free solution, it can quickly find out a solution and has the lowest time cost. DRAND achieved better performance in terms of time cost than D2color, with the price of more color needed to achieve collision-free coloring in all d2-neighborhoods. Regarding message cost, the DRAND algorithm achieved the best performance due to its asynchronous message passing mechanism.

7.3 Stabilization to node failures

We tested the stabilization of PPL when there were frequently node failures which resulted in neighbor discovery failure and message lost. We set the failure probability of each node from $p = 0.1$ to $p = 0.4$. The length of super frame of an online node was set to 5 s. We measured the average time for an online node from node failure discovery to reaching a new online status. The failure probability is a Poisson distribution in our simulation for the whole network.

We chose node set G2 and set time slots as 100 ms. In Fig. 10, we observe that the re-stabilization time increased with larger node failure probability. In addition, with the network size increasing, the re-stabilization time was slightly decreased since it was

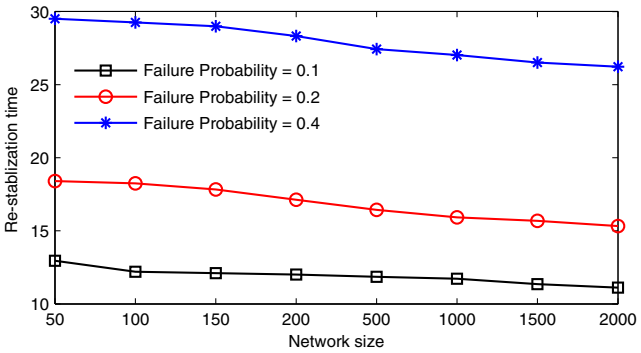

Fig. 10 Average time for re-stabilization of the online status for each node after failure of neighbors

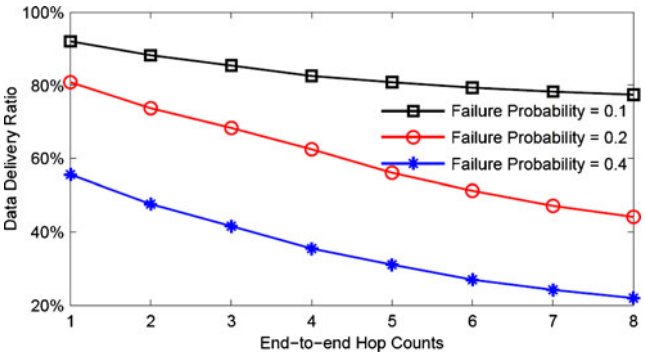

Fig. 11 Delivery ratio in case of node failure

easier for a node to get online beacon broadcast from neighbors which were not failed.

7.4 Data delivery ratio

We defined delivery ratio as the number of packets received by the sink divided by the number of packets sent out by all nodes. We adopt one-hop retransmission mechanism and set the retry times as 3 when there was failure. We varied the network size in different set of experiments. The duration of time slot was set to 100 ms. The traffic load is 1 packet/second for each node. We then measured the average data delivery ratio from nodes which were different hops away from the sink node.

From Fig. 11, we observe that the delivery ratio was decreasing with the increasing of node failure probability. With the same failure probability, the delivery ratio was decreased with the increasing of end-to-end hop count.

8 Conclusion

In this paper, we presented a presence protocol, PPL, over mobile WSNs operated by low duty-cycle fashion. PPL protocol is designed through a multihop and energy efficient approach given the nature of embedded networks. In our design, there are three wakeup modes: online mode, transition mode, and offline mode. The wakeup scheduling of these modes are based on cqs-pair [9] for the sake of energy efficiency and asynchronous neighbor discovery. All nodes in online mode broadcast beacon messages which conveys the online status. Nodes not in online mode asynchronously wake up and try to detect broadcasts from online nodes. The collision of beacon messages from online node was minimized by distributed coloring algorithm. To control the beacon message flooding, we use a breadth-first-search (BFS) spanning tree for schedule book-keeping. All online nodes only book-keep the schedule of their parents in the BFS tree and determine their online or offline status by whether or not they receive beacon broadcasting from their parents. PPL can be implemented with a layered approach without depending on any specific routing protocol or MAC protocol. Our simulation studies demonstrated that PPL is stable and efficient for online status management over mobile sensor networks.

There are several directions for future works. Examples include theoretical analysis of the performance, hierarchical proxy selections for the presence protocol and implementation-based experimental studies.

Acknowledgements This work was supported by the Ministry of Knowledge Economy (MKE) of South Korea. [2008-F-052, Scalable/Mobile/Reliable WSN Technology].

References

1. Song H, Kim D, Lee K, Sung J (2005) UPnP-based sensor network management architecture and implementation. In: Second international conference on mobile computing and ubiquitous networking
2. Ruiz L, Nogueira J, Loureiro A (2003) Manna: a management architecture for wireless sensor networks. IEEE Commun Mag 41(2):116–125
3. Tolle G, Culler D (2005) Design of an application-cooperative management system for wireless sensor networks. In: EWSN, pp 121–132
4. Ramanathan N, Yarvis M, Chhabra J, Kushalnagar N, Krishnamurthy L, Estrin D (2005) A stream-oriented power management protocol for low duty cycle sensor network applications. In: EmNets, pp 53–61
5. Lamport L (1978) Time, clocks, and the ordering of events in a distributed system. Commun ACM 21(7):558–565
6. Deng J, Han R, Mishra S (2006) Secure code distribution in dynamically programmable wireless sensor networks. In: IPSN, pp 292–300
7. Day M, Rosenberg J, Sugano H (2000) A model for presence and instant messaging. IETF RFC 2778, February
8. He T et al (2006) Achieving real-time target tracking using-wireless sensor networks. IEEE RTAS 0, 37–48
9. Lai S, Zhang B, Ravindran B, Cho H (2008) Cqs-pair: cyclic quorum system pair for wakeup scheduling in wireless sensor networks. In: International conference on principles of distributed systems (OPODIS), vol 5401. Springer, pp 295–310
10. Ye W, Heidemann J, Estrin D (2004) Medium access control with coordinated adaptive sleeping for wireless sensor networks. IEEE/ACM Transactions on Networking 12:493–506
11. Polastre J, Hill J, Culler D (2004) Versatile low power media access for wireless sensor networks. In: ACM Sensys, pp 95–107
12. Shaila K, Yeri V, Arjun A, Venugopal K, Patnaik L (2010) Adaptive mobility and availability of a mobile node for efficient secret key distribution inwireless sensor networks. In: Second international conference on machine learning and computing (ICMLC), pp 137 –141

13. Tseng Y, Hsu C, Hsieh T (2002) Power-saving protocols for ieee 802.11-based multi-hop ad hoc networks. In: IEEE international conference on computer communications (INFOCOM), pp 200–209
14. Zheng R, Hou JC, Sha L (2003) Asynchronous wakeup for ad hoc networks. In: MobiHoc, pp 35–45
15. Dutta P, Culler D (2008) Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications. In: Sensys, pp 71–84
16. Niven I, Zuckerman HS, Mongomery HL (1991) Introduction to the theory of numbers. John Wiley & Sons
17. Rhee I, Warrier A, Min J, Xu L (2006) Drand: distributed randomized tdma scheduling for wireless ad-hoc networks. In: MobiHoc, pp 190–201
18. Krumke SO, Marathe MV, Ravi SS (2001) Models and approximation algorithms for channel assignment in radio networks. Wirel Netw 7(6):575–584
19. Kubale M, Kuszner L (2002) A better practical algorithm for distributed graph coloring. In: International conference on parallel computing in electrical engineering (PARELEC), pp 72–75
20. Sommer P, Wattenhofer R (2009) Gradient clock synchronization in wireless sensor networks. In: The international conference on information processing in sensor networks (IPSN), pp 37–48
21. Luk W, Huang T (1997) Two new quorum based algorithms for distributed mutual exclusion. In: ICDCS, pp 100–106
22. Parthasarathy S, Gandhi R (2004) Distributed algorithms for coloring and domination in wireless ad hoc networks. In: FSTTCS, pp 447–C459
23. Sur S, Srimani PK, Srimani PK (1992) A self-stabilizing distributed algorithm to construct BFS spanning trees of a symmetric graph. Comput Math Appl 30:171–179
24. Zuniga M, Krishnamachari B (2004) Analyzing the transitional region in low power wireless links. In: IEEE SECON, pp 517–526
25. Johnson DB, Maltz DA (1996) Dynamic source routing in ad hoc wireless networks. Mobile Computing, Kluwer Academic Publishers, pp 153–181
26. Yoon J, Liu M, Noble B (2003) Random waypoint considered harmful. In: IEEE INFOCOM, vol 2, pp 1312–1321