

Distributed power-source-aware routing in wireless sensor networks

Metin Tekkalmaz¹ · Ibrahim Korpeoglu¹

© Springer Science+Business Media New York 2015

Abstract Although many applications use battery-powered sensor nodes, in some applications battery- and mains-powered nodes coexist. In this paper, we present a distributed algorithm that considers using mains-powered devices to increase the lifetime of wireless sensor networks for such heterogeneous deployment scenarios. In the proposed algorithm, a backbone routing structure composed of mains-powered nodes, sink, and battery-powered nodes if required, is constructed to relay data packets to one or more sinks. The algorithm is fully distributed and can handle dynamic changes in the network, such as node additions and removals, as well as link failures. Our extensive ns-2 simulation results show that the proposed method is able to increase the network lifetime up to 40 % compared to the case in which battery- and mains-powered nodes are not differentiated.

Keywords Wireless sensor networks · Network lifetime · Routing · Heterogeneous networks · Backbone · Power-source-aware · Mains-powered · Energy-efficiency

1 Introduction

Wireless sensor networks (WSNs) are used to monitor physical and environmental conditions in a wide range of civilian and military applications. Such applications include intrusion detection, disaster management,

environment and habitat monitoring, home automation, and industrial process control and monitoring.

In many application scenarios, sensor nodes are randomly deployed in large quantities using methods such as aircraft drops for reasons of safety, harsh environmental conditions, or ease of application. In those cases, all nodes are battery-powered in general, hence they have a limited source of energy. Due to this restriction, careful energy use is vital to maximize WSN lifetime. In other application scenarios, however, it is possible, preferred, or required to manually deploy at least some nodes. Moreover, the area where such a network is deployed may have other energy sources, such as AC power, as for example, in factories, office buildings, or houses. In such deployment environments, some nodes can benefit from the facility's continuous energy source and therefore can be mains-powered. Mains-powered nodes are preferred where possible to reduce maintenance costs, but battery-powered nodes are used where installing power lines is costly or impractical.

One of the most important issues in WSNs regarding energy usage is gathering data from sensor nodes and sending them to the sink node using an energy-efficient routing structure and algorithm. In a network with heterogeneous energy sources, the network protocols and algorithms should make use of this heterogeneity as much as possible to prolong the lifetime of the network. In this paper, we present a distributed power-source-aware routing algorithm, PSABR (power-source-aware backbone-based routing), which increases the lifetime of heterogeneous WSNs where battery- and mains-powered nodes coexist. Although the main focus of the proposed algorithm is on WSNs with battery- and mains-powered nodes, the algorithm can also be used in heterogeneous WSNs with different power source types, as discussed in Sect. 2.

The basic approach behind our PSABR algorithm is to form a backbone routing structure consisting of mains-

✉ Metin Tekkalmaz
metint@cs.bilkent.edu.tr

Ibrahim Korpeoglu
korpe@cs.bilkent.edu.tr

¹ Department of Computer Engineering, Bilkent University, Ankara, Turkey

powered nodes and the sink to relay packets between the sink and the sensor nodes. However, the sink and the mains-powered sensor nodes might not always form a connected topology. Therefore, battery-powered nodes are also used, as needed, to provide a connected backbone for the rest of the network. PSABR is tree-based and is fully distributed, operating without requiring any centralized control. It constructs and maintains a routing tree and establishes parent-child relationships among nodes. Data packets are forwarded according to this relationships. PSABR supports dynamic arrivals and departures of nodes, and it can adapt to node and link failures.

PSABR exploits the heterogeneous power sources to prolong network lifetime and it can work with identical nodes as far as processing power and communication range are considered. On the other hand, the nodes without energy constraints (e.g., mains-powered nodes) may provide more processing power and communication range compared to the nodes with such constraints. PSABR is designed in an asymmetric manner so that the computation intensive tasks are handled by the mains-powered nodes. Furthermore, mains-powered nodes with longer communication range would reduce the number of battery-powered nodes required to interconnect mains-powered nodes while constructing the backbone, which in turn further increases the network lifetime. Hence, PSABR can also benefit from such kinds of heterogeneity.

We implemented and validated our distributed PSABR algorithm in the ns-2 environment. We performed extensive ns-2 simulation experiments to evaluate the efficiency and effectiveness of our algorithm. Our results show that we can obtain up to 40 % increase in network lifetime compared to the shortest path routing algorithm where heterogeneity in power source types is not considered. Simulation results also show that PSABR increases the packet delivery ratio and provides a more balanced energy usage among battery-powered nodes compared to the shortest path routing algorithm. PSABR is scalable with respect to network size and it can react to node arrivals quickly according to the simulation results.

The remainder of this paper is organized as follows: In the next section, we summarize related previous studies. We present our basic routing approach in Sect. 3, and a detailed description of our PSABR algorithm in Sect. 4. We outline the simulation results presenting our algorithm's performance in Sect. 5, and in Sect. 6, we conclude the paper with some suggestions for future work.

2 Related work

In most studies concerning WSNs, nodes are assumed to be battery-powered, but several studies discuss alternative energy sources. Fuel-cells, heat engines, energy harvesting

methods, and power distribution techniques (e.g., through radio frequencies, acoustics, light, etc.) are discussed in [26, 37], and [17]. Some of these methods provide energy for a limited time, similar to batteries, whereas others, such as energy harvesting methods, have the potential to be a continuous source of energy. Therefore, although PSABR is originally designed for networks consisting of battery- and mains-powered nodes, it can be used in similar heterogeneous deployment cases as far as energy sources are concerned. Using a technique similar to the one presented in [23], nodes running PSABR can identify their power source types and act accordingly.

In recent years, energy harvesting methods to power WSNs have been covered by many studies (e.g., [4, 11, 28, 32]), because such methods have great potential to decrease maintenance costs relating to battery replacement and to greatly extend network lifetimes where replacing batteries is impractical. In these studies, different energy sources such as sun, wind, vibration, heat, or electromagnetics are considered for harvesting. In general, harvesting provides intermittent energy because of the sources' unreliable nature and the varying effectiveness of harvesting. Different approaches can be employed to increase the reliability of this method, such as using multiple sources of energy as in [24] and [33]. Another approach is to harvest energy using relatively more reliable ambient energy sources such as fluorescent lamps in hospitals or factories, where the lights are always on (as in [13]) or from air flow near ventilation exhausts (as in [18]). As long as some of the nodes can be powered by reliable methods, PSABR can make use of those nodes to increase the network lifetime.

Basically, PSABR exploits the nodes' heterogeneous power source, assuming some of the nodes have a continuous power source (e.g., mains electricity) and others do not (e.g., battery). There are other studies that use the superior nodes to increase the sensor network lifetime. Yarvis et al. in [42], show that even a modest number of mains-powered nodes has a significant impact on network lifetime. The authors use existing energy-aware routing protocols and try to find the optimum number of battery- and mains-powered nodes along with their locations. Although special placement of nodes can increase network lifetime, PSABR does not rely on such arrangements.

In [22], Ma et al. present a cluster-based topology formation and update protocol that takes the nodes' energy resources into account. Unlike that study, PSABR assumes that all nodes have a similar communication range, which means PSABR can work in network installations where all nodes, regardless of whether they are battery- or mains-powered, use the same wireless communication technology (for example, ZigBee). In [15] and [36], the authors consider node heterogeneity as far as their energy harvesting capabilities are concerned. Kansal et al. [15], propose a

routing method that uses nodes with a higher harvesting potential. Voight et al. [36], describe a modified directed diffusion approach in which solar powered nodes are taken into account.

Our PSABR algorithm employs a backbone-based approach to construct a routing tree by using topology control [19]. As Simplot-Ryl et al. enumerate in [30], backbone-based approaches for data dissemination and gathering are well-studied. As in related studies, [30] considers the backbone to be either a neighbor- or area-dominating set in a network. In the former, all nodes are either part of the backbone or within a one-hop distance, and in the latter, the whole area is within the sensing range of the nodes constituting the backbone. Since finding the minimum connected dominating set (CDS) is NP-complete, approaches in the literature are based on centralized and distributed heuristics. Although centralized algorithms can provide bounds on the size of the CDS, such as in [8], they require global information, increasing the messaging overhead. Localized approaches, in which only limited neighborhood information is shared, are based on either deterministic or probabilistic algorithms. Span, presented in [5], is an example of probabilistic algorithms. A node either sleeps or takes part in the backbone randomly, based on its residual energy and the benefit to its neighbors if it stays awake. In a similar algorithm called EAD [1], Boukerche et al. try to find a spanning tree with many leaf nodes. Nodes with higher residual energy have a higher chance of not being a leaf node. In another distributed algorithm, ASCENT [3], nodes participate in sensing and routing tasks according to packet losses due to a lack of relay nodes or collisions. Hence, the aim is to keep only a subset of the nodes alive to preserve energy. Cell-based approaches, which are also CDS-based, are employed in [39] and [27]. In both studies, the area is divided into cells and only a single node in each cell is kept alive for routing. The major drawback of this method is that the node locations must be known. In the studies mentioned so far, the aim is to find a CDS. In [7] the authors present different protocols that ensure the k -connectedness of dominating sets, in favor of fault tolerance. In another study that takes fault tolerance into account, Kashyap et al. [16] add relay nodes to the WSN to provide a k -connected backbone.

In our study, different from the previous studies based on a backbone, we assume that the sensor nodes are heterogeneous as far as their power source types are concerned. Although studies such as [8] and [5], which take residual energy into account, can be applied to this case, prior knowledge of different power source types enables specialized solutions because the nodes' energy change in time but power sources of individual nodes usually do not. Even if the power source type of a node is altered during the lifespan of a WSN, its frequency is expected to be low. Hence proactive solutions are not required to adapt such

changes. In this study, we also adapt the definition of backbone: in our case, a backbone consists of a connected set of mains-powered nodes, rather than the CDS of all nodes. As mentioned earlier, there are both centralized and localized algorithms for backbone construction, and with our approach presented here, both are possible. Several centralized algorithms is discussed in an earlier study [34], and we present a distributed algorithm in this paper. Our distributed algorithm is deterministic; if there is a connected backbone, the proposed approach is able to construct it, whereas in randomized algorithms backbone connectivity is highly affected by node density. Our proposed approach can also take fault tolerance into account, similar to [7] and [16]; different from those studies, however, our approach tries to increase the number of vertex disjoint paths between pairs of mains-powered nodes (if it is not possible to connect mains-powered nodes directly) on the backbone rather than trying to achieve k -connectedness of the whole backbone. That is because constructing a k -connected topology in a distributed manner is unnecessarily complex compared to ensuring existence of some alternative paths between nodes that are known to be permanent (i.e., mains-powered nodes). Also different from [16], we assume that the locations of the sensor nodes are fixed.

Zeng et al. [43] proposes a routing and scheduling technique that uses learning approach considering a highly mobile environment, but it does not optimize the routing for heterogeneous wireless sensor networks where some nodes can be mains-powered. We consider a more static network, but handle node heterogeneity in terms of power source and energy consumption. Zeng et al. [43] also uses geographic routing which requires knowledge of node positions, which is not required for our PSABR algorithm. Liu et al. [21] also provides a clustering and routing algorithm for highly mobile environments. It does not distinguish between battery- and mains-powered nodes. It constructs multilevel clusters assuming power control is possible to use the optimal transmission range. In PSABR, we do not assume power control and we provide a solution that optimizes for a heterogeneous environment.

We assume, our PSABR algorithm is used in environments where strict delay or bandwidth guarantees are not required. Therefore, our algorithm tries to find minimum cost paths to the sink, similar to EDAL algorithm described in [40] and [41], but unlike these studies, our algorithm does not take delay constraints into account.

3 Proposed routing approach

As mentioned earlier, we assume that battery- and mains-powered sensor nodes coexist in the network, and that the proposed approach uses mains-powered nodes to decrease

energy usage of battery-powered nodes, increasing the overall lifetime of the sensor network. Basically, the proposed approach forms a backbone routing structure that consists of the sink, all the mains-powered sensor nodes (which are accessible from the sink) and some of the battery-powered nodes wherever directly connecting mains-powered nodes is not possible. Then it uses this backbone structure to route packets between the sink and the sensor nodes.

Figure 1 shows a sample network to explain the proposed approach. The sink is located at the center of the area. In the figures, the battery-powered nodes are denoted by white circles and the mains-powered nodes are shown as black circles. Figure 1(a) shows a visibility graph of the network; if a node is in direct communication range of another node, there is an edge between the vertices representing these two nodes. Given this visibility graph, the approach can extract a backbone similar to the one shown in Figure 1(d). Connectivity information of the mains-powered nodes [Figure 1(b)], reduced to a spanning tree [Figure 1(c)], is used to form the backbone, which is explained later in this section. Please note that all the mains-powered nodes take part in the backbone, and in some cases battery-powered nodes are used to interconnect them. Finally, Figure 1(e) shows the routing tree formed as the rest of the nodes connect to the backbone.

The proposed approach can be described in a more formal manner by the following three-step procedure:

1. Reduce the visibility graph $G = (V, E)$ to a secondary graph $G' = (V', E')$ such that
 - (a) $V' \leftarrow \{v \in V \mid v \text{ is mains-powered}\}$,
 - (b) $\forall v_i, v_j \in V'$, the edge $(v_i, v_j) \in E' \iff (v_i, v_j) \in E$ or \exists a simple path $p = (v_1, v_2, \dots, v_n)$ between v_i and v_j in G s.t. v_1, v_2, \dots, v_n are all battery-powered and $|p| < T$, and then
 - (c) assign a cost value to each edge $e' \in E'$.
2. Extract a backbone:
 - (a) Find a spanning tree on G' ,
 - (b) Map the spanning tree on G' to a tree on G .

3. Connect the remaining nodes to the backbone.

This procedure is actually a framework for a class of algorithms rather than a complete description of a single algorithm because there are several alternatives for some of its steps. Let us explain the procedure step by step with the alternatives where necessary.

In the first step, the original network visibility graph is reduced to a secondary graph in which the vertices are the mains-powered nodes (Step 1a) and the edges represent the connectivity of these nodes. Two mains-powered nodes are assumed to be connected either if they are in direct communication range of each other or if there is a path between them with a length less than or equal to a threshold T and consisting of only battery-powered nodes (Step 1b). Figure 1(b) shows a sample secondary graph based on the visibility graph given in Fig. 1a and assuming T is 2. In Step 1c, cost values are assigned to the edges of the secondary graph to be used in Step 2 of the procedure. Following two alternatives are considered for this step, given two vertices representing the mains-powered nodes, and an edge between them: (1) Minimum number of battery-powered nodes that can interconnect the two mains-powered nodes and (2) A value inversely proportional to the number of vertex disjoint paths (shorter than or equal to T and consisting of only battery-powered nodes) between the two mains-powered nodes. The first alternative is expected to reduce the amount of energy consumed by the battery-powered nodes, whereas the second alternative is considered for fault tolerance, that is, if one of the paths between the mains-powered nodes becomes unusable due to a node failure, another path can be chosen from the alternatives.

In the second step of the procedure the backbone is formed. First a spanning tree on the secondary graph is found (Step 2a), similar to the one in Fig. 1(c). This spanning tree is used as the basis of the backbone on the actual network. A minimum spanning tree (MST) and shortest path tree (SPT) routed at the sink are considered as alternatives. Although the MST is expected to give a better network-wide result than the SPT, the latter has a less-complex distributed implementation. The backbone is yielded by mapping the spanning tree on the secondary graph back to a tree on the

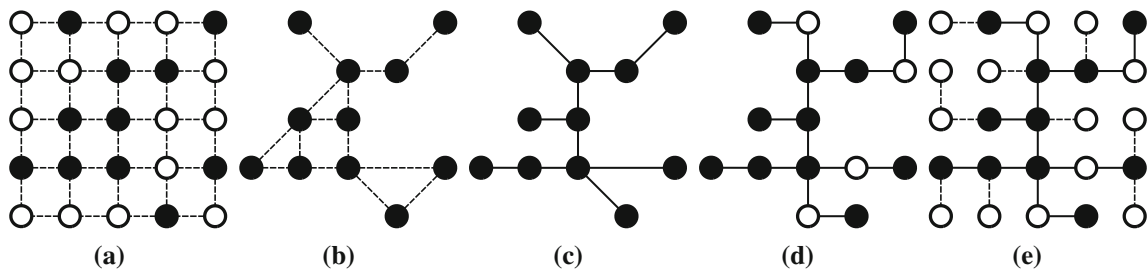


Fig. 1 **a** Visibility graph, **b** secondary graph, **c** spanning tree on the secondary graph, **d** mapping to the original graph, and **e** routing tree

original graph (Step 2b), which corresponds to mapping each edge of the secondary graph to a path (therefore battery-powered nodes constituting that path) between the endpoints of that edge (therefore mains-powered nodes). The mapping can be seen in Figures 1(c, d).

Finally, in the last step of the procedure, disconnected (battery-powered) nodes are connected to the nodes that are part of the backbone (Step 3), either directly or over multiple hops. The final routing tree is similar to the one in Fig. 1 (e). Several centralized algorithms based on the proposed approach were discussed in [34] previously. In this paper we propose a distributed algorithm and it is described in detail next.

4 Our distributed routing algorithm

In this section we propose a power-source aware routing algorithm, PSABR, that can generate a WSN routing tree based on the approach presented in the previous section. PSABR is a fully distributed routing algorithm consisting of a number of sub-algorithms described in detail below.

Let us first define some of the terms that we use in our PSABR algorithm description. Two nodes are *neighbors* if they are within communication range. The *peer* of a mains-powered node is another mains-powered node (which might be the sink) that is reachable through less than T battery-powered nodes. Battery- and mains-powered nodes send their data to their *parents* in the data gathering process. The parent of a mains-powered node is one of its peers, whereas the parent of a battery-powered node is one of its neighbors. Each node has an associated *cost* value. Cost of the sink is 0 and cost of any other node is the sum of its parent's cost and the cost to reach its parent.

In PSABR, battery- and mains-powered nodes have different behaviors. Mains-powered nodes maintain a list of peers, along with the possible paths to each peer. To achieve this, each mains-powered node stores a partial view of the global visibility graph, which contains nodes at most T hops away. Mains-powered nodes also gather the cost information of their peers. With this information, a mains-powered node chooses one of its peers as its parent and a path to reach that parent. The backbone consists of the mains-powered nodes as well as the battery-powered nodes chosen by the mains-powered nodes to reach their parents. Mains-powered nodes have an active role; they try to maintain the partial visibility graph and determine their parents, and in turn the backbone, in a distributed manner. Battery-powered nodes, on the other hand, are mostly passive. In the backbone construction and maintenance process, they mainly forward control messages sent by the mains-powered nodes.

PSABR provides means, for mains-powered nodes, to discover the peer nodes along with their cost values and to

obtain their up to date cost values. Having the current information each mains-powered node tries to minimize its own cost continually by choosing the most appropriate parent. Therefore, given a cost metric, PSABR guarantees that all the mains-powered nodes on the backbone have the least cost values.

PSABR can handle node arrivals and departures, therefore, it does not require a network-wide construction phase. As battery- and mains-powered nodes are added or removed, the algorithm constructs and maintains efficient backbone and routing paths. Our algorithm is also designed to work with any number of sinks. Similar to sensor nodes, sink nodes can be added at a later time.

The control messages used by PSABR are transferred by either broadcast or source routing [10]. The route information in a source-routed packet is extracted from the partial visibility graph maintained by the nodes. Data messages between a mains-powered node and its parent are transferred by table-driven routing. The intermediate battery-powered nodes on the backbone construct routing tables using the information extracted from the control messages they forward. We assume all nodes have periodic data to send to the sink. PSABR can work regardless of data aggregation [9, 20, 38] takes place. If data aggregation is applied, mains-powered nodes aggregate their own data with the data received from their child nodes and send the aggregated messages to their parents, battery-powered nodes that are on the backbone aggregate their data with the data they forward, and battery-powered nodes that are not on the backbone send the data packets to their parents which are one hop away. If data aggregation is not possible, mains-powered nodes and battery-powered nodes that are not on the backbone send their data to their parents whenever new data is available, and battery-powered nodes that are on the backbone send their data to any next-hop node that resides their routing table.

In the following subsections, we describe how mains-powered nodes maintain the partial visibility graph and how nodes choose their parent in a distributed manner. In Sect. 4.1, we present the control messages used to gather information and form the parent-child relations (i.e., the backbone). We show how the messages are used on a sample scenario, in Sect. 4.2. In Sect. 4.3, we describe the node behavior at different events, including when receiving the control messages. Finally, in Sect. 4.4, we discuss the messaging overhead of PSABR.

4.1 Messages

MDM (MP-node discovery message) An MDM is initiated to discover mains-powered nodes that are either direct neighbors or accessible through battery-powered nodes. The number of battery-powered nodes connecting mains-powered nodes is restricted by a threshold number T . MDM is transferred by

broadcast. An MDM(s, r, ps) contains the originator s of the message; the path r , which consists of battery-powered nodes, followed until the packet reaches its current receiver; and the power-source type ps of the originator.

MIM (MP-node information message) An MIM is initiated either by a mains-powered node as a response to an MDM or by a battery-powered node while joining the network. An MIM is transferred by unicast using source routing. An MIM(s, d, r, V, E, I) contains the originator s and the destination d of the message; the path r that the packet should follow; the node set V containing source, destination, and the nodes connecting them; the edge set E representing the one-hop connectivity of the nodes in V ; and the tuple set I containing the mains-powered node and cost pairs.

MUM (MP-node update message) An MUM is initiated by a mains-powered node to inform peers when its cost to reach the sink has changed. An MUM is transferred by unicast using source routing. An MUM(s, d, r, c) contains the originator s and the destination d of the message, the path r that the packet should follow, and the cost c of the originator. Note that the cost of a node is the number of battery-powered nodes between that node and the sink in the current routing settings.

BCM (backbone construction message) A BCM is initiated by a mains-powered node to establish a path to another mains-powered node, possibly through battery-powered nodes. A BCM is transferred by unicast using source routing. A BCM(s, d, r) contains the originator s , the destination d of the message, and the path r that the packet should follow. Each BCM should be replied by a BCMACK(s, d, r), in order to acknowledge a peer node's backbone (i.e., parent-child relation) construction request. A BCMACK is transferred by unicast using source routing and contains the same fields as a BCM.

LFM (link failure message) An LFM is initiated by a battery-powered node to inform the originator of a data or control message that could not be transferred to the next node about the link failure (i.e., the next intermediate battery-powered node is unreachable). An LFM can be caused by messages routed either by source routing or by table-driven routing. The LFM's routing method matches the routing method of the message that caused it. Hence, an LFM_{ldr} is generated for messages that are routed by table-driven mechanisms and an LFM_{sr} is generated for messages that are source routed. An $LFM_{ldr}(d, un, up)$ contains the destination d of the message, the unreachable battery-powered node un , and the unreachable mains-powered node up due to link failure. An $LFM_{sr}(d, r, un, up)$ contains the path r that the packet should follow, in addition to the information that an LFM_{ldr} contains.

NDM (neighbor discovery message) An NDM is initiated by a battery-powered node to discover its immediate neighbors. An NDM is transferred by broadcast. An NDM(s) contains the originator s of the message.

NIM (neighbor information message) An NIM is initiated by either a battery- or mains-powered node as the response to an NDM. An NIM is transferred by unicast. An NIM(s, d, ps, c) contains the originator s , the destination d of the message, the power-source type ps , and the cost c of the originator.

4.2 Sample backbone construction

In this section, we explain how the messages described in Sect. 4.1 are used to construct a backbone using a sample scenario shown in Fig. 2.

In Fig. 2, battery-powered nodes are shown with small circles and labeled with lower-case letters from a to h , whereas mains-powered nodes are shown with larger circles and labeled with upper-case letters K, L, M, N, P , and S . There is a line between nodes if they are in the communication range of each other. Parent-child relations are shown with solid lines. S is the sink of the sample network, hence its cost is 0. In this sample network, we assume T is 3, that is, peers can be at most 3 hops away.

We assume, initially the mains-powered nodes reachable from the sink are L, M , and N , as shown in Figure 2(a). K , on the other hand, is not reachable from the sink, therefore its cost is infinity. Assume that a new mains-powered node P joins to the network as shown in Fig. 2(b). As soon as the node joins it broadcasts an MDM. As shown in the figure, the MDM contains the originator P and the list of battery-powered nodes that the message has visited, which is initially empty (\emptyset). Note that the power source field is omitted in this example. Since MDM is a broadcast message, it is received by all the one-hop neighbors of P (i.e., b, d, g , and h). As shown in Fig. 2(c), all the battery-powered receivers add themselves to the list of battery-powered nodes field and rebroadcast the message. Similarly, as the two-hop battery-powered neighbors of P receive the MDMs, they update the messages adequately and rebroadcast them as shown in Fig. 2d. Note that, c receives two MDMs originated by P . It rebroadcasts both of them by adding itself to the message because the goal is to discover all paths between P and its potential peers (K, L , and M in this case). Another point worth mentioning here is, b and d receive the MDMs sent by a and c , but they do not rebroadcast these messages since they are already included in the list of battery-powered nodes of the MDMs. On the other hand, as shown in Fig. 2(e), e drops the MDMs sent by c , since T is 3. Battery-powered nodes check the length of the list of battery-powered nodes in MDMs to decide whether to rebroadcast or to drop these messages.

K, L , and M know all the possible paths to P by receiving all the MDMs originated from it. For example, L has received three MDMs from P , more specifically the following messages: MDM($P, [b, a]$), MDM($P, [b, c]$), and MDM($P, [d, c]$). Using these messages, L updates its partial

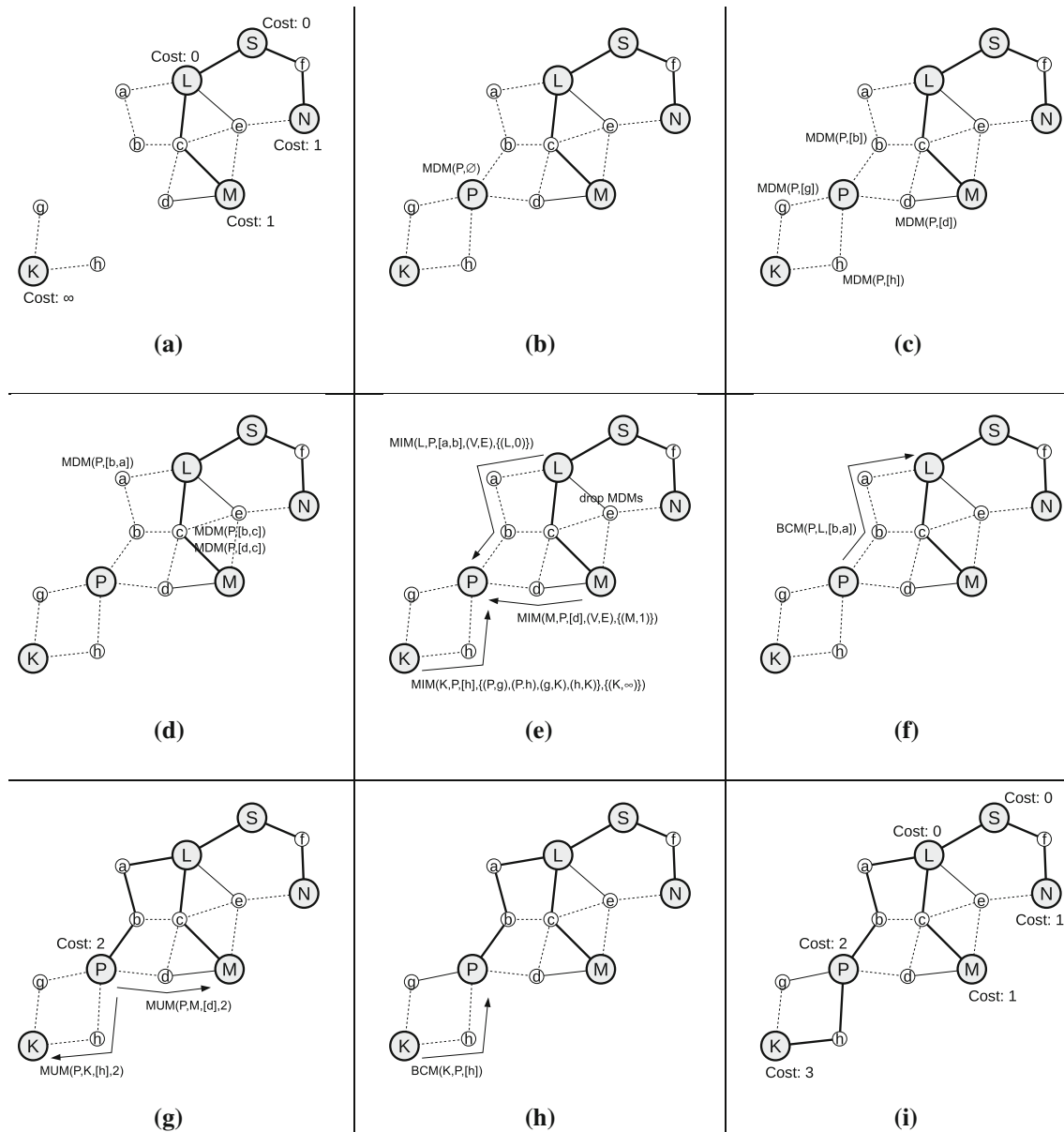


Fig. 2 Backbone construction

visibility graph to include the following edges: (P, b) , (b, a) , (a, L) , (b, c) , (c, L) , (P, d) , and (d, c) . Once the peers of P receive all the MDMs, they reply using MIM as shown in Fig. 2(e). MIMs contain the newly discovered paths between the peers, as well as the cost of the discovered peers. In the figure, only the contents of MIM sent by K is shown completely due to space limits, in the others the discovered nodes and edges are denoted by (V, E) .

As P receives the MIMs from K , L , and M , it checks whether there is a parent candidate among the newly discovered peers. Here, both L and M have cost less than infinity: L 's cost is 0, and M 's cost is 1. But shortest path length between P and L is 2, whereas it is 1 between P and M , meaning that cost of P will be 2 independent of its parent

choice. We assume P chooses L as its parent candidate and sends a BCM to form a new path in the backbone as shown in Fig. 2(f). L replies with a BCMACK to confirm the backbone path construction (not shown in the figures).

Once P becomes part of the backbone and its cost changes from infinity to 2, it sends MUMs to its peers as shown in Fig. 2(g). MUMs contain the updated cost information of P . As K receives the MUM from P , it sends BCM to P to become part of the backbone [Fig. 2(h)]. The final topology, after P and K exchange BCM and BCMACK, is depicted in Figure 2(i). Note that, although not shown in the figures, nodes also broadcast NIMs as their costs change, so that battery-powered nodes can update their parent if they are not on the backbone.

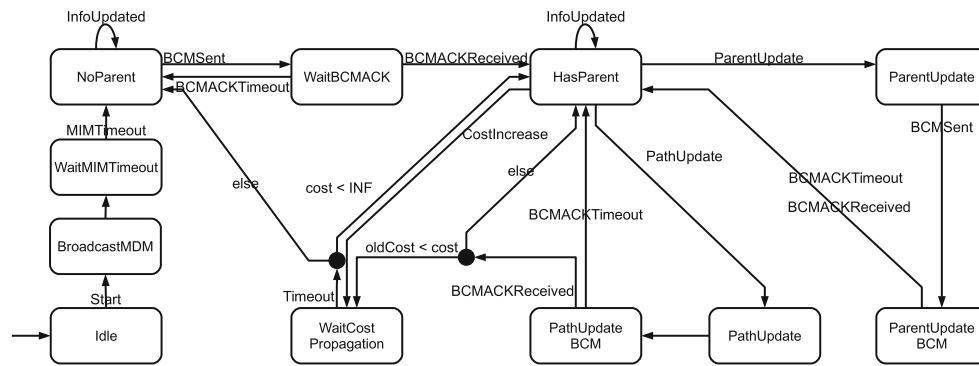


Fig. 3 Finite state machine for mains-powered nodes

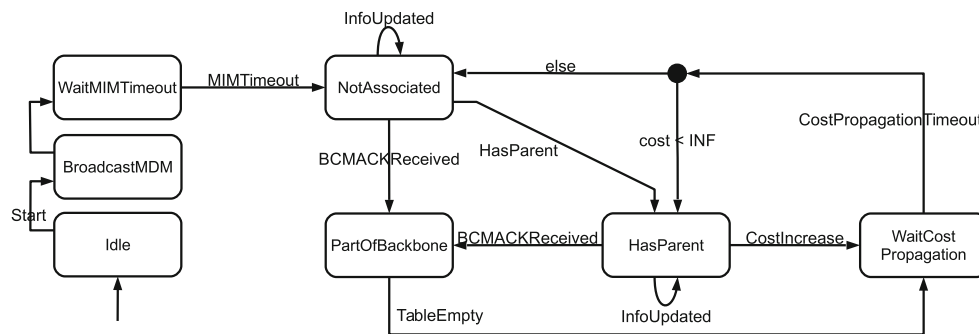


Fig. 4 Finite state machine for battery-powered nodes

4.3 Behavior

Node behavior, described in the form of finite state machines (FSM) are depicted in Figs. 3 and 4, and the algorithms are presented in Algorithms 1–8. The list of algorithms is given in Table 1. Each type of node has a different reaction to an external event depending on its current state. Since battery- and mains-powered nodes have different behaviors, they have separate FSMs and separate sets of algorithms. We explain the variables and the expressions used in the algorithms in Table 2.

Table 1 Algorithms

Alg		Title
1	MP	On power-up
2	MP	On entry of NoParent
3	MP	On entry of HasParent
4	MP	On message receive (1 of 2)
5	MP	On message receive (2 of 2)
6	BP	On power-up
7	BP	On entry of HasParent
8	BP	On message receive

The FSM of a mains-powered node is depicted in Fig. 3. Initially, a mains-powered node is in the *Idle* state. With the *Start* event, it transits into the *BroadcastMDM* state. *Start* is

Table 2 Variables and expressions

var./expr.	Usage
<i>self</i>	Address of the node executing the algorithm
<i>peers</i>	Set of peers of a mains-powered node, which is initially empty (i.e., \emptyset)
<i>neighbors</i>	Set of neighbors of a battery-powered node, which is initially empty (i.e., \emptyset)
<i>parent</i>	A node that is used to reach to the sink, which is initially undefined (i.e., \perp). The <i>parent</i> of a mains-powered node is one of its peers and <i>parent</i> of a battery-powered node is one of its neighbors
<i>cost</i>	Number of intermediate battery-powered nodes traversed to reach the sink, which is initially infinity (i.e., ∞)
<i>pathToParent</i>	The path used by a mains-powered node to reach its current parent
<i>peers[p].cost_{peer}</i>	Cost of the peer <i>p</i>
<i>peers[p].cost</i>	Cost of the node if the sink is reached through peer <i>p</i>
$ r $	Length of path <i>r</i>
$SP(s, d, V, E)$	The shortest path between <i>s</i> and <i>d</i> given a vertex set <i>V</i> and an edge set <i>E</i>

fired when the node is powered up, as shown in Algorithm 1. In *BroadcastMDM* state, a mains-powered node broadcasts an MDM and transits directly into *WaitMIMTimeout* state in which it starts a timer, t_{mim} , for corresponding MIMs. Each MIM received restarts the timer, as shown in Algorithm 4, line 2b. As t_{mim} expires, a *MIMTimeout* event is fired, which means that a certain amount of time has passed since the last MIM, and the node transits into the *NoParent* state.

Algorithm 1 On Power-up (MP-Node)

```

1:  $peers \leftarrow \emptyset$ 
2:  $parent \leftarrow \perp, cost \leftarrow \infty$ 
3:  $V \leftarrow \{self\}, E \leftarrow \emptyset$ 
4: fire Start
  
```

From the *Idle* state to the *NoParent* state, a mains-powered node discovers all other mains-powered nodes that are accessible through less than T battery-powered nodes, and alternative paths to them. As an MDM is received by a battery-powered node, it adds itself to the path that the packet has followed thus far and rebroadcasts it (if less than $T - 1$ battery-powered nodes have been traversed), as shown in Algorithm 8. Therefore, as an MDM is received by a mains-powered node, a path from the originator to that node is discovered, and when all MDMs originating from the same mains-powered node are received, all possible paths (bounded by length T) between these two mains-powered nodes are known. When a mains-powered node receives all the MDMs originating from a mains-powered node, it replies with an MIM, which contains all the alternative paths to that node, as shown in Algorithm 4. Finally, as a mains-powered node receives all the MIMs corresponding to the MDM it has sent, discovery of its peers and possible paths to them is completed. Through this process, the node has a partial view of the global visibility graph.

Note that an existing mains-power node discovers a newly joined mains-powered node (and possible paths to it) by the MDMs originating from the new node and following different paths. On the other hand, a newly joined mains-powered node discovers existing mains-powered nodes by the MIMs, which are replies to the MDM it has sent. A third method of discovery of new peers or new paths to known peers is presented later in this section.

Algorithm 2 On Entry of NoParent

```

1:  $candidate \leftarrow p$  s.t.  $peers[p].cost$  is minimum
2: if  $candidate \neq \perp$  then
3:    $path \leftarrow SP(self, candidate, V, E)$ 
4:   send BCM( $self, candidate, path$ )
5:   schedule  $t_{bcmack}$ 
6:   fire BCMSent
7: end if
  
```

Although MDMs are broadcast messages, they not flooded to the whole network since they are not rebroadcast by mains-powered nodes (Algorithm 4) and they are broadcast by the battery-powered nodes only if the packet is rebroadcast less than $(T - 1)$ times and the current node is not already included in the path that the message traversed so far (Algorithm 8, line 1a).

Algorithm 3 On Entry of HasParent (MP-Node)

```

1: if  $cost < \infty$  then
2:   if  $parent \in peers$  then
3:     if  $peers[parent].cost > cost$  then
4:       if  $peers[parent].cost < \infty$  then
5:          $cost \leftarrow peers[parent].cost$ 
6:       else
7:          $parent \leftarrow \perp, cost \leftarrow \infty$ 
8:       end if
9:       fire CostIncrease
10:   else
11:     if  $\exists p \in peers$  s.t.  $peers[p].cost < peers[parent].cost$  then
12:       fire ParentUpdate
13:     else
14:       if  $pathToParent$  still exists then
15:         if  $cost \neq peers[parent].cost$  then
16:            $cost \leftarrow peers[parent].cost$ 
17:           send MUM to peers
18:         end if
19:       else
20:         fire PathUpdate
21:       end if
22:     end if
23:   end if
24: else
25:    $parent \leftarrow \perp, cost \leftarrow \infty$ 
26:   send MUM to peers
27:   fire CostIncrease
28: end if
29: else
30:   send MUM to peers
31:   fire CostIncrease
32: end if
  
```

As the entry action of the *NoParent* state, the node tries to find a parent candidate and sends a BCM to the best parent candidate to establish a parent-child relation with that node. With the *BCMSent* event, which is fired when a BCM is sent to a parent candidate, the node transits into the *WaitBCMACK* state. As the BCM is sent, a timer, t_{bcmack} , is also started. If t_{bcmack} expires before the corresponding BCMACK message is received (which fires a *BCMACKTimeout* event) the node returns to the *NoParent* state. If the BCMACK is received on time (which fires a *BCMACKReceived* event) it transits into the *HasParent* state. The entry action of the *NoParent* state is given in Algorithm 2.

If the parent role is acknowledged by the parent candidate using BCMACK, the mains-powered node transits into the *HasParent* state. The BCM/BCMACK messages allow

construction of part of the backbone also by informing the battery-powered nodes (see Algorithm 8) between the parent and child mains-powered nodes. On the entry to the *HasParent* state, the node checks whether it can still access its current parent, whether the path to the current parent has changed and whether the cost of the current parent has changed. Then, if required, it takes the appropriate action among the following: starts updating the parent, starts updating the path to its current parent, or starts disseminating the cost change to its peers. The exact procedure is given in Algorithm 3.

Algorithm 4 On Message Receive (MP-Node) - 1 of 2

When MDM(s, r, ps) arrives:

```

1a: if  $s \neq self$  then
2a:   if first MDM from  $s$  then
3a:      $V_s \leftarrow \{self, s\}, E_s \leftarrow \emptyset$ 
4a:   end if
5a:   for all nodes  $n_1, n_2, \dots, n_k$  on path  $r$  do
6a:      $V_s \leftarrow V_s \cup \{n_i\}$ 
7a:   end for
8a:    $E_s \leftarrow E_s \cup \{(self, n_1)\}$ 
9a:   for  $i = 1$  to  $k - 1$  do
10a:     $E_s \leftarrow E_s \cup \{(n_i, n_{i+1})\}$ 
11a:  end for
12a:   $E_s \leftarrow E_s \cup \{(n_k, s)\}$ 
13a:  if first MDM from  $s$  then
14a:    Schedule timer  $t_s$  for  $s$ 
15a:  else
16a:    Reschedule  $t_s$ 
17a:  end if
18a:  Wait until  $t_s$  expires
19a:  send MIM( $self, s, SP(self, s, V_s, E_s), V_s, E_s, \{(self, cost)\}$ )
20a:  if  $ps = MP$  then
21a:     $peers \leftarrow peers \cup \{s\}$ 
22a:     $peers[s].cost_{peer} \leftarrow \infty$ 
23a:     $V \leftarrow V \cup V_s, E \leftarrow E \cup E_s$ 
24a:    fire InfoUpdated
25a:  end if
26a: end if

```

When MIM(s, d, r, V', E', I) arrives:

```

1b: if  $|I| = 1$  then
2b:   Reschedule  $t_{mim}$ 
3b: end if
4b:  $V \leftarrow V \cup V', E \leftarrow E \cup E'$ 
5b: update  $V$  and  $E$  s.t.  $\forall v \in V, SP(self, v, V, E) \leq T$ 
6b: for all  $(i_0, i_1) \in I$  do
7b:   if  $i_0 \in V$  then
8b:     $peers \leftarrow peers \cup \{i_0\}$ 
9b:     $peers[s].cost_{peer} \leftarrow i_1$ 
10b:     $peers[s].cost \leftarrow i_1 + |SP(self, i_0, V, E)|$ 
11b:  end if
12b: end for
13b: if  $|I| > 1$  then
14b:   fire InfoUpdated
15b: end if

```

If a parent candidate that leads the node to a lower cost value is found in the *HasParent* state (Algorithm 3, line 11), a *ParentUpdate* event is fired and the node transits into the *ParentUpdate* state. On entry to the *ParentUpdate* state, the node sends a BCM to the best parent candidate to establish a parent-child relation with, starts a timer, and fires a *BCMSent* event. When the corresponding BCMACK

is received (i.e., *BCMACKReceived* event) or the timer expires (i.e., *BCMACKTimeout* event), the node returns to the *HasParent* state, with its parent updated, or preserves its previous parent.

Algorithm 5 On Message Receive (MP-Node) - 2 of 2

When MUM(s, d, r, c) arrives:

```

1c:  $peers[s].cost_{peer} \leftarrow c$ 
2c:  $peers[s].cost \leftarrow c + |SP(self, s, V, E)|$ 
3c: fire InfoUpdated

```

When LFM($*, un, up$) arrives:

```

1d: if  $un \neq up$  then
2d:   for all  $e \in E$  do
3d:     if  $e$  is incident to  $un$  then
4d:        $E \leftarrow E - \{e\}$ 
5d:     end if
6d:   end for
7d:   for all  $v \in V$  do
8d:     if  $\nexists$  a path between  $self$  and  $v$  in  $G(V, E)$  or  $\forall$  path  $p$  between  $self$  and  $v$  in  $G(V, E)$ ,  $\exists$  a mains-powered vertex  $v$  on  $p$  then
9d:        $V \leftarrow V - \{v\}$ 
10d:     end if
11d:   end for
12d:   for all  $p \in peers$  do
13d:     if  $p \in V$  then
14d:        $peers[p].cost \leftarrow peers[p].cost_{peer} + |SP(self, p, V, E)|$ 
15d:     else
16d:        $peers \leftarrow peers - \{p\}$ 
17d:     end if
18d:   end for
19d: else
20d:    $V \leftarrow V - \{up\}$ 
21d:   for all  $e \in E$  do
22d:     if  $e$  is incident to  $up$  then
23d:        $E \leftarrow E - \{e\}$ 
24d:     end if
25d:   end for
26d:    $peers \leftarrow peers - \{up\}$ 
27d: end if
28d: fire InfoUpdated

```

When BCM(s, d, r) arrives:

```

1e: send BCMACK( $self, s, reverse(r)$ )

```

When BCMACK(s, d, r) arrives:

```

1f:  $parent \leftarrow s$ 
2f:  $cost \leftarrow peers[s].cost$ 
3f: for all  $p \in peers$  do
4f:   send MUM( $self, p, SP(self, p, V, E), cost$ )
5f: end for
6f: broadcast NIM( $self, *, MP, cost$ )

```

When NDM(s) arrives:

```

1g: send NIM( $self, s, MP, cost$ )

```

If the path to the current parent needs updating due to a link failure on the current path, in the *HasParent* state, a *PathUpdate* event is fired (Algorithm 3, line 20) and the node goes into the *PathUpdate* state. On entry to the *ParentUpdate* state, the node sends a BCM to the current parent to update the path to that node and then transits into the *PathUpdateBCM* state. If a BCMACK is not received

on time, the node transits into the *HasParent* state without a successful path update process. Otherwise (i.e., that is the corresponding BCMACK is received), the node goes to the *WaitCostPropagation* state or back to the *HasParent* state, depending on the current and previous cost values.

The node cost increases in the following two cases: the parent cost increases (Algorithm 3, line 3) or the current parent becomes unreachable (Algorithm 3, line 24), both of which lead to the *HasParent* \rightarrow *WaitForCostPropagation* state transition; or a higher-cost path to its parent needs to be established, which leads to *HasParent* \rightarrow *PathUpdate* \rightarrow *PathUpdateBCM* \rightarrow *WaitForCostPropagation* state transitions. In these cases, the node needs to advertise the new cost and wait for the information to disseminate before attempting to find a better-cost parent; otherwise parent-child loops will occur, if a node connects with one of its descendants. When the timer for disseminating the increased cost information expires, a *Timeout* event is fired and the node transits into the *HasParent* state if the cost of its parent is less than infinity and it can still reach its parent; otherwise it transits into the *NoParent* state.

When node information such as set of peers, cost of peers, paths to peers, etc. changes (see algorithms 4 and 5 for such cases), an *InfoUpdated* event is fired. Note that this event causes self-transitions in *NoParent* and *HasParent* states, so the node can check for a parent candidate (Algorithm 2) or the validity of the current parent (Algorithm 3), respectively.

The FSM of a battery-powered node is depicted in Fig. 4. Initially, a battery-powered node is in the *Idle* state. With the *Start* event, it transits into the *BroadcastMDM* state. A *Start* event is fired when the node is powered up, as shown in Algorithm 6. In *BroadcastMDM* state, a battery-powered node broadcasts MDM and transits directly into *WaitMIMTimeout* state in which it starts a timer, t_{mim} , for corresponding MIMs. Each MIM received restarts the timer. As t_{mim} expires, a *MIMTimeout* event is fired, which means that a certain amount of time has passed since the last MIM, and the node transits into the *NotAssociated* state.

Algorithm 6 On Power-up (BP-Node)

```

1:  $neighbors \leftarrow \emptyset$ 
2:  $parent \leftarrow \perp, cost \leftarrow \infty$ 
3:  $V \leftarrow \{self\}, E \leftarrow \emptyset, I \leftarrow \emptyset$ 
4: fire Start
  
```

A battery-powered node discovers nearby mains-powered nodes by broadcasting an MDM and receiving corresponding MIMs (Algorithm 8, line 4b), which is a similar process to the initial peer discovery of mains-powered nodes. But here, this information (i.e., a partial

graph obtained as in Algorithm 8, line 6b) is not consumed by the battery-powered node but is distributed back to the mains-powered nodes when t_{mim} expires. Therefore, mains-powered nodes can discover new peers or new paths to their existing peers with the help of newly joined battery-powered nodes.

On entry to the *NotAssociated* state, the node tries to determine its parent. If it has one or more neighbors whose cost is less than infinity, it sets the one with the minimum cost as its parent and fires a *HasParent* event, which makes it transit into the *HasParent* state.

Algorithm 7 On Entry of HasParent (BP-Node)

```

1: if  $parent \in neighbors$  then
2:   if  $neighbors[parent].cost > cost$  then
3:     if  $neighbors[parent].cost = \infty$  then
4:        $parent \leftarrow \perp, cost \leftarrow \infty$ 
5:     else
6:        $cost \leftarrow neighbors[parent].cost + 1$ 
7:   end if
8:   broadcast NIM( $self, *, BP, cost$ )
9:   fire CostIncrease
10: else
11:   if  $\exists n \in neighbors$  s.t.  $neighbors[n].cost < (cost - 1)$  then
12:      $parent \leftarrow n, cost \leftarrow neighbors[n].cost + 1$ 
13:     broadcast NIM( $self, *, BP, cost$ )
14:   end if
15: end if
16: else
17:    $parent \leftarrow \perp, cost \leftarrow \infty$ 
18:   broadcast NIM( $self, *, BP, cost$ )
19:   fire CostIncrease
20: end if
  
```

On entry to the *HasParent* state, the node checks whether it can still access its current parent (Algorithm 7, line 1), whether the cost of the current parent has changed (Algorithm 7, line 2), and whether there is a parent candidate with a better cost (Algorithm 7, line 11) and takes the appropriate action among the following: updates its parent, starts disseminating the cost change, or fires a *CostIncrease* event.

If, in the *NotAssociated* or *HasParent* states, a node receives a BCMACK message (which indicates that it is now on the backbone), a *BCMACKReceived* event is fired (Algorithm 8, line 3g) and the node transits into the *PartOfBackbone* state. As shown in Algorithm 8, when a battery-powered node receives BCM and BCMACK messages, it establishes backward and forward routing entries. As long as a battery-powered node is part of the backbone, it forwards data packets from one mains-powered node to another using table-driven routing. Unused entries expire and are removed from the table.

If, when in the *PartOfBackbone* state, the routing table of a node becomes empty (i.e., no data packet is forwarded recently) and therefore the node realizes that it is no longer on the backbone (which results in a *TableEmpty* event), or, when it is in the *HasParent* state because its cost has increased (which results in a *CostIncrease* event), it transits into the *WaitCostPropagation* state. When the timer expires and a *CostPropagationTimeout* event is fired, the node transits into the *HasParent* state if it still has a parent (i.e., its cost is less than infinity) or transits into the *NotAssociated* state otherwise.

When information such as a set of neighbors, cost of neighbors, etc. changes, an *InfoUpdated* event is fired. Note that this event causes self-transitions in *NotAssociated* and *HasParent* states; therefore the node can check for a parent or the validity of the current parent.

4.4 Analysis

This section analyzes PSABR messaging overhead. In the analysis, n is the total number of nodes in the network, r is the communication range of each node, and R is the diameter of the deployment area, assuming that it is circular. Furthermore, m is the mains-powered node ratio, where $0 \leq m \leq 1$. Hence, there are mn mains-powered nodes and $(1 - m)n$ battery-powered nodes.

Assuming the nodes are deployed uniformly, the expected number of t -hop neighbors, k_t , of a node is given in Eq. 1. It is the total number of nodes multiplied by the ratio of the area of the ring, whose inner and outer radii are $(t - 1)r$ and tr , to the whole area.

$$k_t = n \left(\frac{\pi[(tr)^2 - ((t - 1)r)^2]}{\pi R^2} \right) = n(2t - 1) \left(\frac{r}{R} \right)^2 \quad (1)$$

The most expensive operation in PSABR is mains-powered node discovery, which involves transmitting MDMs and MIMs. Once an MDM is broadcast by the originator, it is rebroadcast by the battery-powered nodes until time to live (TTL) expires, and replied by the mains-powered nodes, using MIM. Assuming that a mains-powered node is allowed to have peers at most T hops away (i.e., TTL is T), an upper bound for the expected number of packets transmitted due to a mains-powered node discovery, $C_{discover}$, is given in Eq. 2. The left operand of the addition is the total number of MDMs transmitted. It is the summation of total number of messages transmitted after each rebroadcast. Since the MDMs are dropped by the T th battery-powered nodes, the summation is from 1 to $(T - 1)$. The right

operand of the addition is the summation of number of mains-powered nodes for each hop count multiplied by the hop count (i.e., the number transmission required for a MIM to reach from a mains-powered node to the originator of the MDM). As mentioned earlier, a battery-powered powered node does not rebroadcast an MDM if it is already included in the path that the message traversed so far. Eq. 2 provides an upper bound, since it does not take this behavior into account. $C_{discover}$ is the number of messages due to the mains-powered node discovery process of a single node, therefore $nC_{discover}$ gives the maximum number of messages sent in the network for this purpose.

$$C_{discover} \leq \sum_{t=1}^{T-1} [k_1(1 - m)]^t + \sum_{t=1}^T k_t m t \quad (2)$$

Although mains-powered node discovery is a rather expensive operation, because it is performed only once by each node (upon joining the network) its cost is amortized by the benefits it provides, as shown in simulation results, in Sect. 5. In the same section, we also compare Eqs. 1 and 2 with the simulation results.

Assuming parent-child relations are formed once between mains-powered nodes, the upper bound for the total number of messages in the network required for this purpose is $2nmT$. This value is the total number of BCM/BCM-ACK messages exchanged by the mains-powered nodes that are at most T hops away. Contrary to the assumption, parent-child relations might be established several times for each mains-powered node, due to battery-powered node deaths or discovery of lower-cost paths to the sink. Therefore, it is hard to present an equation for the number of messages required for establishing parent-child relations (i.e., forming the backbone) for the duration of the network.

From time to time, mains-powered nodes need to advertise changes in their cost values. The expected number of MUMs sent for this purpose, $C_{advertise}$, is given in Eq. 3, and is basically the number of transmissions required to send MUMs to each of the peers, that is summation of the number of peers at each level multiplied by the distance to the peers. Similar to the case in the total cost of backbone construction, it is hard to predict the total number of cost changes during the network lifetime. But note that, the cost change of a mains-powered node causes cost changes in all of its descendants. Furthermore, if the cost has decreased, non-child peers might chose the node as a parent, causing cost updates in other mains-powered nodes.

$$C_{advertise} = \sum_{t=1}^T k_t m t \quad (3)$$

Algorithm 8 On Message Receive (BP-Node)

When $\text{MDM}(s, r, ps)$ arrives:

1a: **if** $(|r| < (T - 1))$ **and** $(self \notin r)$ **then**

2a: $r' \leftarrow r + self$

3a: **broadcast** $\text{MDM}(s, r', ps)$

4a: **end if**

When $\text{MIM}(s, d, r, V', E', I')$ arrives:

1b: **if** $s \neq self$ **then**

2b: $i \leftarrow \text{index of } self \text{ in list } r$

3b: **send** $\text{MIM}(s, d, r, V, E, I)$ to $r[i + 1]$

4b: **else**

5b: **Reschedule** t_{mim}

6b: $V \leftarrow V \cup V', E \leftarrow E \cup E', I \leftarrow I \cup I'$

7b: **end if**

When $\text{MUM}(s, d, r, c)$ arrives:

1c: $i \leftarrow \text{index of } self \text{ in list } r$

2c: **send** $\text{MUM}(s, d, r, c)$ to $r[i + 1]$

When $\text{LFM}_{tdr}(d, un, up)$ arrives:

1d: **send** $\text{LFM}_{tdr}(d, un, up)$ to $next-hop[d]$

When $\text{LFM}_{sr}(d, r, un, up)$ arrives:

1e: $i \leftarrow \text{index of } self \text{ in list } r$

2e: **send** $\text{LFM}_{sr}(d, r, un, up)$ to $r[i + 1]$

When $\text{BCM}(s, d, r)$ arrives:

1f: $i \leftarrow \text{index of } self \text{ in list } r$

2f: **if** $i = 0$ **then**

3f: $next-hop[s] \leftarrow s$

4f: **else**

5f: $next-hop[s] \leftarrow r[i - 1]$

6f: **end if**

7f: **if** $i = (|r| - 1)$ **then**

8f: $next-hop[d] \leftarrow d$

9f: **else**

10f: $next-hop[d] \leftarrow r[i + 1]$

11f: **end if**

12f: **send** $\text{BCMACK}(s, d, r)$ to $next-hop[d]$

When $\text{BCMACK}(s, d, r)$ arrives:

1g: *same as the lines [1f, 11f]*

2g: **send** $\text{BCMACK}(s, d, r)$ to $next-hop[d]$

3g: **fire** BCMACKReceived

When $\text{NDM}(s)$ arrives:

1h: **send** $\text{NIM}(self, s, BP, cost)$

When $\text{NIM}(s, d, ps, c)$ arrives:

1i: $neighbors \leftarrow neighbors \cup \{s\}$

2i: $neighbors[s].cost \leftarrow c$

3i: **fire** InfoUpdated

5 Performance evaluation

We implemented and simulated our proposed routing algorithm as a network layer protocol in the ns-2 [35] (version 2.34) simulation environment. IEEE 802.15.4 [14], which is already available in ns-2, is used as the

underlying MAC and physical layer protocol. Packet loss model implemented in ns-2 is used to better reflect real life behavior of PSABR. To compare PSABR's performance, we use a shortest-path routing implementation. In the shortest-path routing, packets are forwarded from a node to the sink through the path with the minimum-hop distance among all possible paths.

We use several parameters in the simulations to observe the impact of different conditions on the algorithm's performance (Table 3). These parameters include network size, n , battery-powered node ratio, m , node density, ρ , and number of sinks, σ . The value of a data point is obtained by averaging the results across 20 simulation runs, unless otherwise stated. In each simulation run, the locations of the nodes are determined pseudo-randomly, based on the approach described in [2]. Node arrival times are also determined randomly, keeping all the aforementioned parameters intact.

$$E = P_{tx}T_{tx} + P_{rx}T_{rx} \quad (4)$$

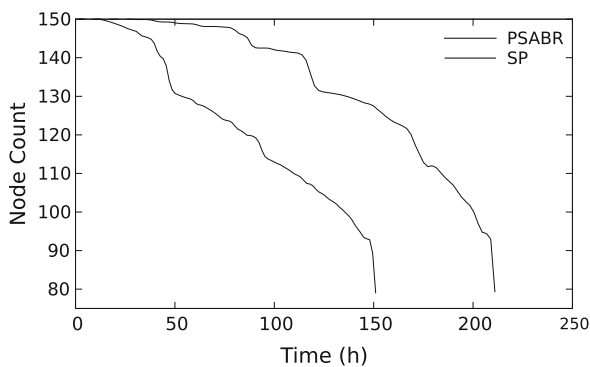
In the simulations, we used the energy consumption model already available in ns-2 as shown in Eq. 4. In the equation, E , P , and T denote energy, power, and time, whereas subscripts tx and rx denote transmission and reception, respectively. Power values used for transmission and reception are based on [25]. According to [25], nodes consume 0.0807 W as they transmit and 0.0801 W as they receive. The initial energy of battery-powered nodes is 3 J. Although this value is known to be rather low for a battery, our experiments with different initial energy values show that factor does not proportionally affect performance, so we kept it low to obtain the simulation results in a reasonable time. We assume each node has periodic data to send and the data is aggregated as it is routed to the sink. In the simulations each node sends a packet with a 32-byte payload every 60 s.

We consider a node to be *reachable* if the algorithm establishes a routing path between the node and the sink. Even if a node is alive, unless there is a routing path to the sink, it cannot contribute to the sensor network. Therefore, we think a reachable-node count better reflects the algorithms' performance compared to an alive-node count. We assume the network terminates when half the nodes become unreachable, and we run the simulations accordingly, unless otherwise noted.

Figure 5 presents the change in the reachable-node count over time. The figure is given for a certain node count and mains-powered node ratio, but the algorithm exhibits similar behavior for other values of the node count and mains-powered node ratio. While PSABR is hesitant to use battery-powered nodes as forwarding nodes, the shortest-path routing does not distinguish between battery- and mains-powered nodes. Therefore, in PSABR, the reachable-node count remains mostly flat with sudden

Table 3 Simulation parameters

Abbr.	Parameter	Value
n	Node count	100, 150,..., 300
m	Mains-powered node ratio	10, 15,...,40 %
ρ	Node density	60, 80,...,140 %
σ	Sink count	1, 2, 4
T	Max. hops between peers	3
	Transmit power	0.0807 W
	Receive power	0.0801 W
	Initial energy	3 J
	Data packet payload size	32 bytes
	Data packet interval	60 s

**Fig. 5** Number of reachable nodes over time ($n = 150$, $m = 20\%$)

drops, whereas in the shortest-path routing, it decreases almost linearly over time.

Figure 6 presents a more detailed look into the algorithm behavior. As mentioned earlier, PSABR's basic approach is to eliminate battery-powered nodes on the routing paths; they are pushed down to the leaves of the routing tree. Figure 6(a) shows how the average in-degree of battery-powered nodes changes over time. Note that an average in-degree of 0 means that none of the battery-powered nodes forwards data packets. The average in-degree is below 0.2 for PSABR and remains rather stable until the end of the network, which shows PSABR is rather successful in its basic approach. On the other hand, the in-degree for the shortest-path routing is around 0.9 at the beginning (meaning that on the average almost every battery-powered node is an intermediate node on a routing path) and it decreases to around 0.3 linearly as time passes. This behavior of shortest-path routing is primarily due to a decrease in the battery-powered node ratio because of battery depletion.

In Fig. 6(b, c), we show the experimental results related to energy usage. Figure 6(b) shows that the total energy usage of battery-powered nodes for different time frames is

rather stable for PSABR, which is a direct result of the stable average in-degree value for the battery-powered nodes. Earlier time frames show almost twice as much energy usage by battery-powered nodes for the shortest-path routing, but this decreases linearly according to the decrease in the average in-degree of the battery-powered nodes and the battery-powered node count. Figure 6(c) depicts the total residual energy of the battery-powered nodes, which decreases almost linearly in both algorithms, but the decrease in the shortest-path routing has a steeper slope.

The number of packets transmitted in the network is shown in Fig. 6 (d), as is a breakdown for control traffic and actual data traffic. Initially, PSABR exchanges a relatively higher number of control packets for network construction, and from time to time it requires some control packets for self-organization due to node deaths. In general, however, a higher number of data packets are delivered to the sink in PSABR compared to shortest-path routing, since earlier battery-powered node deaths reduces the number of packets sent and forwarded in shortest-path routing.

Figure 7 shows how PSABR reacts to new node arrivals. The figure depicts the number of battery- and mains-powered nodes (dark and light gray areas, respectively) over time, and the number of reachable nodes (solid line). Note that number of battery- and mains-powered nodes are plotted as a stacked chart, hence they sum up to the total number of nodes. Values are taken from a single simulation run, i.e., not averaged over multiple runs, in order to visualize the algorithm's reactions in more detail. 300 nodes arrive in about 600 s, with uniformly distributed random arrival times. As evident from the figure, the number of reachable nodes is close to the total number of nodes, with sudden decreases followed by increases, from time to time. These fluctuations are due to switches to better routing paths, which become possible as new nodes arrive.

Figure 8 depicts network lifetime under different conditions, with Fig. 8a showing the algorithm's performance with respect to the total node count. In general, the performance is unaffected by node count. Algorithm performance with respect to the mains-powered node ratio is shown in Fig. 8(b), and as evident, PSABR performs better than the shortest-path routing overall, but it achieves the best results in the 15–25 % range. For lower ratios, the mains-powered nodes do not confer significant advantage to PSABR. For higher ratios, coincidental exploitation of the mains-powered nodes is high enough for the shortest-path routing to achieve results similar to PSABR. Figure 8(c) shows the effect of node density on network lifetime. In the other experiments, node density is around 1 node per 44 unit² (note that the communication range is around 20 units). Here, the lifetime is given for different density values relative to the usual case, ranging from 60 %

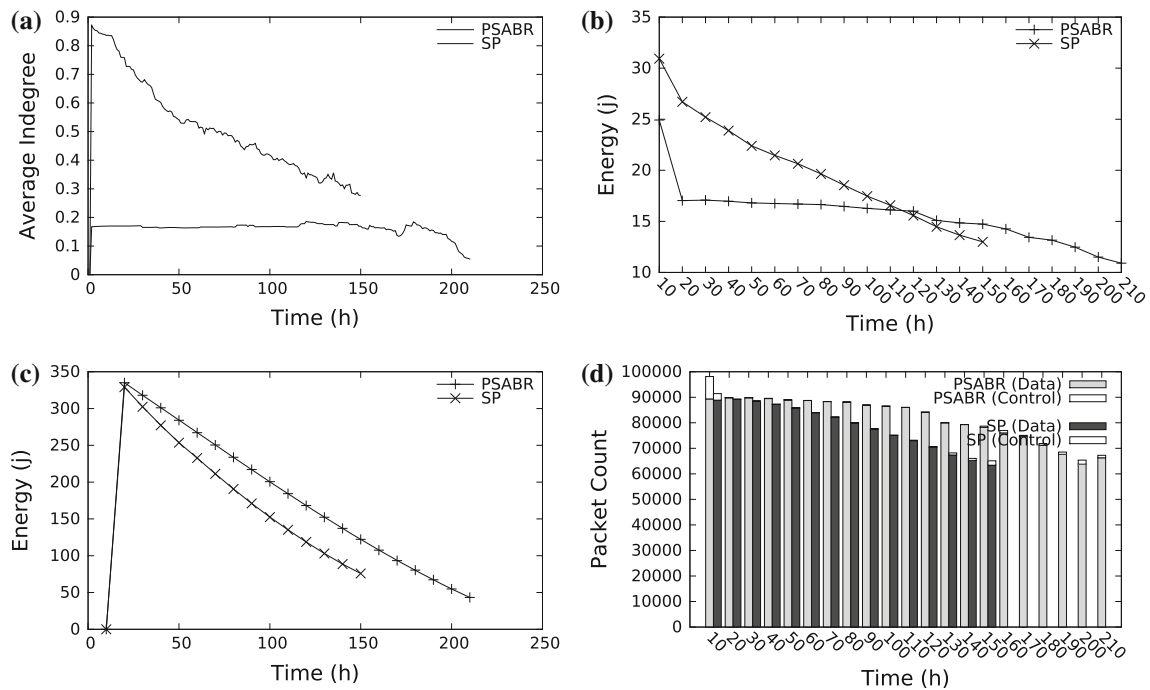


Fig. 6 **a** Average in-degree of battery-powered nodes, **b** energy consumption of battery-powered nodes, **c** total residual energy of battery-powered nodes, and **d** number of packets transmitted over time ($n = 150$, $m = 20\%$)

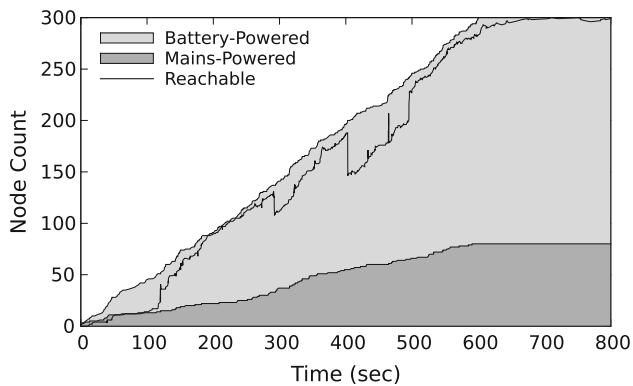


Fig. 7 Node counts as new nodes arrive and the network is constructed by PSABR ($n = 300$, $m = 25\%$)

(approx. 1 node per 73 unit²) to 140 % (approx. 1 node per 31 unit²). As the density increases, the lifetime of the network also increases because it is possible to eliminate more battery-powered nodes on the paths to the sink.

Figure 8 also shows the experimental upper bound. Given the simulation parameters (i.e., 32-byte packets transmitted every 60 s, 0.0807 W transmission energy, and 3 J initial energy), the bound is the time required until the energy of a battery-powered device drains completely, assuming that it does not exchange control packets or forward others' data but only transmits its own data packets. Note that the bound is not tight for cases where battery-powered devices are required to forward data

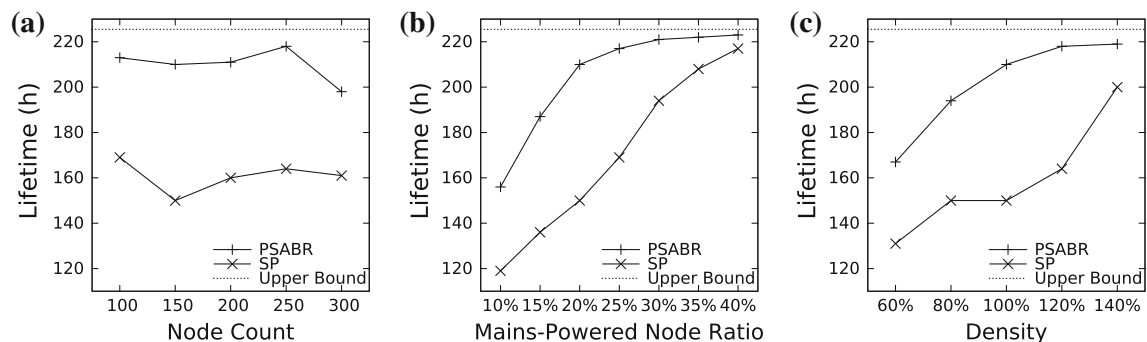


Fig. 8 Network lifetime (assuming lifetime is the time passed until half of the nodes become unreachable from the sink) depending on **a** node count ($m = 20\%$, $\rho = 100\%$), **b** mains-powered node ratio ($n = 150$, $\rho = 100\%$), and **c** density ($n = 150$, $m = 20\%$)

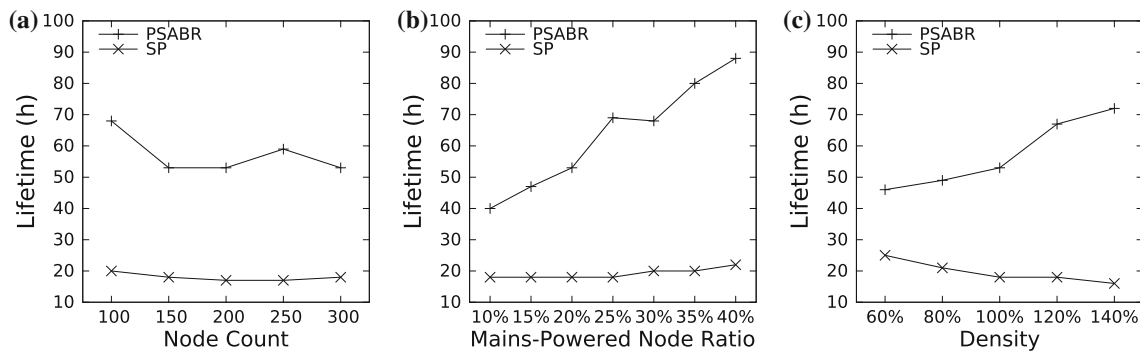


Fig. 9 Network lifetime (assuming lifetime is the time passed until the first node death) depending on **a** node count ($m = 20\%$, $\rho = 100\%$), **b** mains-powered node ratio ($n = 150$, $\rho = 100\%$), and **c** density ($n = 150$, $m = 20\%$)

packets, such as with low node density or a low mains-powered node ratio.

For the simulation results given in Fig. 8, we assume the lifetime is the time passed until the half of the nodes become unreachable as mentioned earlier. In Fig. 9, we give the corresponding results if the lifetime is defined as the time passed until the energy of a node depletes completely (first node death). Compared to the previous results PSABR in this case exhibits similar behavior, that is, node count does not have significant effect on the algorithm performance but as the mains-powered node ratio or the density increases, PSABR performs better. Shortest-path routing, on the other hand, is mostly unaffected by the change in node count, mains-powered node ratio and density. Note that the lifetime values for this case are much less than the upper bound. This is because for the given parameters it is hard to achieve the case in which none of the battery-powered nodes forwards any data. Still, in first node death case, PSABR performs much better than the shortest-path routing, as shown in the figures.

The number of control packets required to construct the initial routing tree with respect to network size is presented in Fig. 10. As shown in the figure, PSABR requires three to five times more control packets for construction. This scenario is expected because PSABR has a more complicated control messaging scheme compared to shortest-path routing. Regardless, the simulation results show that PSABR increases the network lifetime (Fig. 8). It is notable that the increase in the control packet count is linear in PSABR as well as in shortest-path routing, showing that the algorithms are scalable for different network sizes.

The proposed algorithm is designed to run with an arbitrary number of sinks, and Fig. 11 presents how the algorithms perform for various values of sink counts. As the number of sinks increases, network lifetime increases for both algorithms, as expected, but the performance difference in PSABR is not obvious because it performs close to the experimental upper bound even for the single-sink case. On

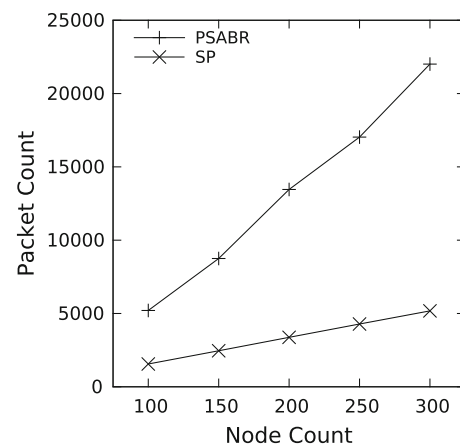


Fig. 10 Number of control packets required to construct the network ($m = 20\%$)

the other hand, the average path length between the nodes and the sink almost halves as the number of sinks increases from one to four. If fast delivery of data packets is important, using multiple sink can still be considered.

In Fig. 12, we compare the number of messages required for a mains-powered node to discover all of its peers found by analysis as given in Sect. 4.4, with the simulation results. In the comparison, we fixed the total number of nodes to 300 and the mains-powered node ratio to 20 % and we gave the results for different node densities, which are obtained by changing the size of the deployment area. As shown in the figure, the values computed according the equations are very close to the values obtained from the simulations. The computed values are higher, since the Eq. 2 is an upper bound on the expected number of messages required for mains-powered node discovery. Note that, these values are for a single mains-powered node discovery, if all the nodes exist in the network. But if the nodes join to the network gradually, as in the rest of the simulations, earlier discoveries require less number of messages, due to lower node density. Hence the

Fig. 11 Lifetime and average path length to sink for different sink counts ($n = 150$, $m = 20\%$)

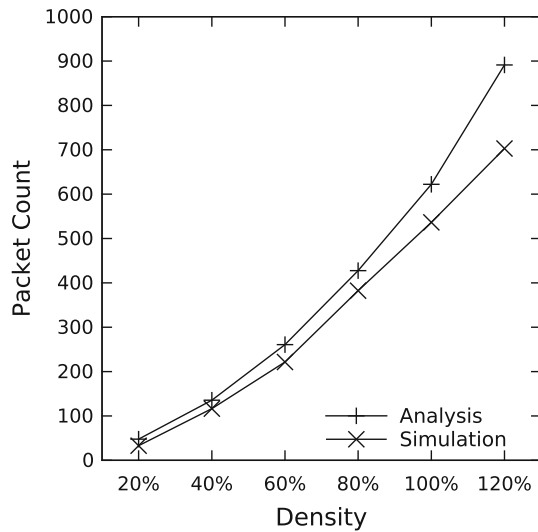
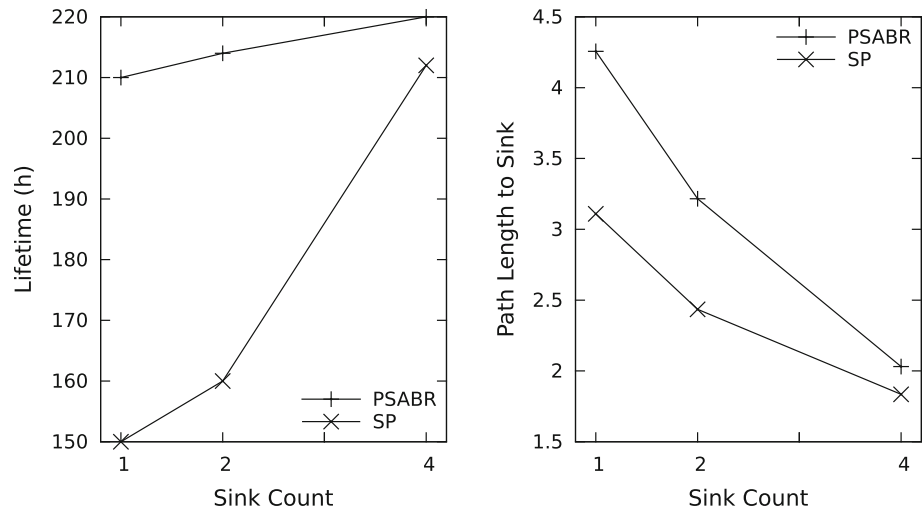


Fig. 12 Number of messages (MDM and MIM) required to discover a peer ($n = 300$, $m = 20\%$)

total number of messages required for mains-powered node discovery would be much less than $nC_{discover}$, given in Sect. 4.4.

6 Conclusions

In this paper, we described a routing approach and proposed a distributed routing algorithm (PSABR) based on this approach, which is able to increase the lifetime of WSNs where different power-source types for nodes exist. Our PSABR algorithm first forms a backbone in a distributed fashion to relay the data packets. The backbone consists of mains-powered nodes that are assumed to coexist with battery-powered nodes.

In addition to the theoretical analysis of PSABR, we also presented the simulation results. As the results show, distinguishing between sensor nodes according to their power source types increases network lifetime by as much as 40 %. This result is achieved mainly by eliminating battery-powered nodes as forwarding nodes. Although PSABR has a higher control message overhead, we showed that it is scalable with network size and is still more energy efficient than a conventional routing approach that does not distinguish between power-source types. Simulation results also revealed that PSABR is able to react to node additions rather quickly. We also presented the effects of node count, mains-powered node ratio, density, and sink count, on PSABR performance. In most cases, PSABR performs close to the theoretical upper bound and much better than conventional shortest-path routing, as far as the network lifetime is concerned.

Currently, PSABR uses the number of battery-powered nodes as the cost metric while forming the backbone. As shown in Sect. 3, other cost metrics are possible, such as the number of vertex disjoint paths, which is expected to favor reliability. The effect of such cost metrics can be explored in a future study.

In the proposed algorithm, all nodes are kept in idle mode and the simulation results are obtained accordingly. Putting non-backbone nodes into sleep mode could further extend network lifetime [12, 29] and the benefits of such a scheme could also be analyzed in a future study.

As described in Sect. 4, an intermediate battery-powered node is assumed to be dead if packet loss occurs and an alternative route is established. On the other hand, a packet loss does not always indicate a node failure but might occur due to congestion. In a future study, distinguishing node failures and congestion, as in [6], can be studied for the sake of better resource utilization.

In its current form, PSABR establishes routing paths for a given set of battery- and mains-powered nodes with fixed locations. In [31], a method for using additional nodes to cope with minimal exposure problem is proposed. A similar approach, to reduce the burden on battery-powered nodes can be employed in PSABR context, as a future work.

Acknowledgments This work is supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK), Project Number 113E274.

References

1. Boukerche, A., Cheng, X., & Linus, J. (2003). Energy-aware data-centric routing in microsensor networks. In *Proceedings of the 6th ACM international workshop on modeling analysis and simulation of wireless and mobile systems (MSWIM '03)* (pp. 42–49). New York, NY, USA. doi:[10.1145/940991.941000](https://doi.org/10.1145/940991.941000).
2. Camilo, T., Silva, J. S., Rodrigues, A., & Boavida, F. (2007). GENSEN: A topology generator for real wireless sensor networks deployment. In *5th IFIP workshop on software technologies for future embedded and ubiquitous systems*.
3. Cerpa, A., & Estrin, D. (2004). ASCENT: Adaptive self-configuring sensor networks topologies. *IEEE Transactions on Mobile Computing*, 3(3), 272–285. doi:[10.1109/TMC.2004.16](https://doi.org/10.1109/TMC.2004.16).
4. Chalasani, S., & Conrad, J. (2008). A survey of energy harvesting sources for embedded systems. In *IEEE SoutheastCon* (pp. 442–447). doi:[10.1109/SECON.2008.4494336](https://doi.org/10.1109/SECON.2008.4494336).
5. Chen, B., Jamieson, K., Balakrishnan, H., & Morris, R. (2002). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5), 481–494.
6. Chilamkurti, N., Zeadally, S., Vasilakos, A., & Sharma, V. (2009). Cross-layer support for energy efficient routing in wireless sensor networks. *Journal of Sensors*. doi:[10.1155/2009/134165](https://doi.org/10.1155/2009/134165).
7. Dai, F., & Wu, J. (2006). On constructing k-connected k-dominating set in wireless ad hoc and sensor networks. *Journal of Parallel and Distributed Computing*, 66(7), 947–958. doi:[10.1016/j.jpdc.2005.12.010](https://doi.org/10.1016/j.jpdc.2005.12.010).
8. Das, B., Sivakumar, R., & Bharghavan, V. (1997). Routing in ad hoc networks using a spine. In *6th international conference on computer communications and networks (ICCCN'97)* (p. 34). doi:[10.1109/ICCCN.1997.623288](https://doi.org/10.1109/ICCCN.1997.623288).
9. Fasolo, E., Rossi, M., Widmer, J., & Zorzi, M. (2007). In-network aggregation techniques for wireless sensor networks: A survey. *IEEE Wireless Communications*, 14(2), 70–87.
10. Feibel, W. (Ed.). (1995). *Encyclopedia of networking* (2nd ed.). The Network Press. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.394.3596&rep=rep1&type=pdf>.
11. Gilbert, J., & Balouchi, F. (2008). Comparison of energy harvesting systems for wireless sensor networks. *International Journal of Automation and Computing*, 5(4), 334–347. doi:[10.1007/s11633-008-0334-2](https://doi.org/10.1007/s11633-008-0334-2).
12. Han, K., Luo, J., Liu, Y., & Vasilakos, A. V. (2013). Algorithm design for data communications in duty-cycled wireless sensor networks: A survey. *IEEE Communications Magazine*, 51(7), 107–113.
13. Hande, A., Polk, T., Walker, W., & Bhatia, D. (2007). Indoor solar energy harvesting for sensor network router nodes. *Microprocessors and Microsystems*, 31(6), 420–432. doi:[10.1016/j.micpro.2007.02.006](https://doi.org/10.1016/j.micpro.2007.02.006). Special Issue on Sensor Systems.
14. IEEE Computer Society LAN/MAN Standards Committee. (2006). IEEE standard for local and metropolitan area networks—Part 15.4: Low-rate wireless personal area networks (LR-WPANs).
15. Kansal, A., Hsu, J., Srivastava, M., & Raghunathan, V. (2006). Harvesting aware power management for sensor networks. In *Proceedings of the 43rd annual design automation conference (DAC'06)* (pp. 651–656). ACM, New York, NY, USA. doi:[10.1145/1146909.1147075](https://doi.org/10.1145/1146909.1147075).
16. Kashyap, A., Khuller, S., & Shayman, M. A. (2006). Relay placement for higher order connectivity in wireless sensor networks. In: *INFOCOM*.
17. Knight, C., Davidson, J., & Behrens, S. (2008). Energy options for wireless sensor nodes. *Sensors*, 8(12), 8037–8066. doi:[10.3390/s8128037](https://doi.org/10.3390/s8128037).
18. Lee, S., Youn, B. D., & Jung, B. C. (2009). Robust segment-type energy harvester and its application to a wireless sensor. *Smart Materials and Structures*, 18(9), 095021.
19. Li, M., Li, Z., & Vasilakos, A. V. (2013). A survey on topology control in wireless sensor networks: Taxonomy, comparative study, and open issues. *Proceedings of the IEEE*, 101(12), 2538–2557.
20. Liu, X. Y., Zhu, Y., Kong, L., Liu, C., Gu, Y., Vasilakos, A. V., & Wu, M. Y. (2014). CDC: Compressive data collection for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 26(8), 2188–2197.
21. Liu, Y., Xiong, N., Zhao, Y., Vasilakos, A. V., Gao, J., & Jia, Y. (2010). Multi-layer clustering routing algorithm for wireless vehicular sensor networks. *IET Communications*, 4(7), 810–816.
22. Ma, Y., Dala, S., Alwan, M., & Aylor, J. (2003). ROP: A resource oriented protocol for heterogeneous sensor networks. In: *Virginia tech symposium on wireless personal communications*.
23. Mikhaylov, K., & Tervonen, J. (2011). Node's power source type identification in wireless sensor networks. In: *International conference on broadband and wireless computing, communication and applications (BWCCA'11)* (pp. 521–525). doi:[10.1109/BWCCA.2011.84](https://doi.org/10.1109/BWCCA.2011.84).
24. Park, C., & Chou, P. (2006). AmbiMax: Autonomous energy harvesting platform for multi-supply wireless sensor nodes. In: *3rd annual IEEE communications society on sensor and ad hoc communications and networks (SECON '06)* (vol. 1, pp. 168–177). doi:[10.1109/SAHCN.2006.288421](https://doi.org/10.1109/SAHCN.2006.288421).
25. Prince-Pike, A. (2009). *Power characterisation of a zigbee wireless network in a real time monitoring application*. Ph.D. thesis, AUT University.
26. Roundy, S., Steingart, D., Frechette, L., Wright, P., & Rabaey, J. (2004). Power sources for wireless sensor networks. In *Wireless Sensor Networks, Lecture Notes in Computer Science* (Vol. 2920, pp. 1–17). Berlin: Springer.
27. Santi, P., & Simon, J. (2004). Silence is golden with high probability: Maintaining a connected backbone in wireless sensor networks. In *Wireless Sensor Networks, Lecture Notes in Computer Science* (Vol. 2920, pp. 106–121). Berlin Heidelberg: Springer.
28. Seah, W. G., Eu, Z. A., & Tan, H. (2009). Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP)—Survey and challenges. In *1st international conference on wireless communication, vehicular technology, information theory and aerospace electronic systems technology (Wireless VITAE'09)* (pp. 1–5). doi:[10.1109/WIRELESSVITAE.2009.5172411](https://doi.org/10.1109/WIRELESSVITAE.2009.5172411).
29. Sengupta, S., Das, S., Nasir, M., Vasilakos, A. V., & Pedrycz, W. (2012). An evolutionary multiobjective sleep-scheduling scheme for differentiated coverage in wireless sensor networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 42(6), 1093–1102.

30. Simplot-Ryl, D., Stojmenovic, I., & Wu, J. (2005). Handbook of sensor networks, chap. 11. Energy-efficient backbone construction, broadcasting, and area coverage in sensor networks. (pp. 343–380). Wiley Series on Parallel and Distributed Computing. Wiley: New York.
31. Song, Y., Liu, L., Ma, H., & Vasilakos, A. V. (2014). A biology-based algorithm to minimal exposure problem of wireless sensor networks. *IEEE Transactions on Network and Service Management*, 11(3), 417–430.
32. Sudevalayam, S., & Kulkarni, P. (2011). Energy harvesting sensor nodes: Survey and implications. *IEEE Communications Surveys and Tutorials*, 13(3), 443–461. doi:10.1109/SURV.2011.060710.00094.
33. Tan, Y., & Panda, S. (2011). Energy harvesting from hybrid indoor ambient light and thermal energy sources for enhanced performance of wireless sensor nodes. *IEEE Transactions on Industrial Electronics*, 58(9), 4424–4435. doi:10.1109/TIE.2010.2102321.
34. Tekkalmaz, M., & Korpeoglu, I. (2010). Power-source-aware backbone routing in wireless sensor networks. In *IEEE international conference on communication systems (ICCS'10)* (pp. 46–50). doi:10.1109/ICCS.2010.5686105.
35. The Network Simulator ns-2. (2013). Retrieved July 8, 2013 from <http://www.isi.edu/nsnam/ns/>
36. Voigt, T., Ritter, H., & Schiller, J. (2003). Utilizing solar power in wireless sensor networks. In: *Proceedings of the 28th IEEE annual international conference on local computer networks (LCN'03)* (pp. 416–422). doi:10.1109/LCN.2003.1243167.
37. Watral, Z., & Michalski, A. (2013). Selected problems of power sources for wireless sensors networks. *IEEE Instrumentation Measurement Magazine*, 16(1), 37–43. doi:10.1109/MIM.2013.6417056.
38. Xiang, L., Luo, J., & Vasilakos, A. (2011). Compressed data aggregation for energy efficient wireless sensor networks. In *IEEE 2011 8th annual IEEE communications society conference on sensor, mesh and ad hoc communications and networks (SECON)* (pp. 46–54).
39. Xu, Y., Heidemann, J., & Estrin, D. (2001). Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on mobile computing and networking (MobiCom'01)* (pp. 70–84). ACM, New York, NY, USA. doi:10.1145/381677.381685.
40. Yao, Y., Cao, Q., & Vasilakos, A. V. (2013). Edal: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for wireless sensor networks. In *2013 IEEE 10th international conference on mobile ad-hoc and sensor systems (MASS)* (pp. 182–190).
41. Yao, Y., Cao, Q., & Vasilakos, A. V. (2015). Edal: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks. *IEEE/ACM Transactions on Networking*, 23(3), 810–823. doi:10.1109/TNET.2014.2306592.
42. Yarvis, M., Kushalnagar, N., Singh, H., Rangarajan, A., Liu, Y., & Singh, S. (2005). Exploiting Heterogeneity in Sensor Networks. In *Proceedings of the 24th annual joint conference of the IEEE computer and communications societies (INFOCOM'05)* (Vol. 2, pp. 878–890). doi:10.1109/INFCOM.2005.1498318.
43. Zeng, Y., Xiang, K., Li, D., & Vasilakos, A. V. (2013). Directional routing and scheduling for green vehicular delay tolerant networks. *Wireless Networks*, 19(2), 161–173.



Metin Tekkalmaz received his B.S., M.S., and Ph.D. degrees in computer engineering from Bilkent University, Ankara, Turkey, in 2002, 2004, and 2013, respectively. Currently, he is a software engineer in ASELSAN Inc. His research interests include wireless ad hoc and sensor networks. He is a member of IEEE.



Ibrahim Korpeoglu received his Ph.D. and M.S. degrees in computer science from University of Maryland, College Park, USA, and B.S. degree in computer engineering from Bilkent University, Ankara, Turkey. He is currently an associate professor in the Department of Computer Engineering of Bilkent University. Prior to that, he has worked in several companies in USA including Ericsson, IBM T.J. Watson Research Center, Bell Laboratories, and Telcordia

Technologies. His research interests include wireless ad hoc and sensor networks, mobile computing, peer-to-peer networks, and distributed systems. He is a member of IEEE and ACM.