POWER-SOURCE-AWARE ADAPTIVE ROUTING IN WIRELESS SENSOR NETWORKS

A DISSERTATION SUBMITTED TO
THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Metin Tekkalmaz
July, 2013

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. İbrahim Körpeoğlu (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. Özgür Ulusoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. Tolga Mete Duman

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Uğur Güdükbay

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Ahmet Coşar

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

POWER-SOURCE-AWARE ADAPTIVE ROUTING IN WIRELESS SENSOR NETWORKS

Metin Tekkalmaz
Ph.D. in Computer Engineering
Supervisor: Assoc. Prof. Dr. İbrahim Körpeoğlu
July, 2013

A wireless sensor network (WSN) is a collection of sensor nodes distributed over an area of interest to accomplish a certain task by monitoring environmental and physical conditions and sending the collected data to a special node called sink. Most studies on WSNs consider nodes to be powered with irreplaceable batteries, which limits network lifetime. There are, however, perpetual power source alternatives as well, including mains electricity and energy harvesting mechanisms, which can be utilized by at least some portion of the sensor nodes to further prolong the network lifetime.

Our aim here is to increase the lifetime of such WSNs with heterogeneous power sources by centralized or distributed routing algorithms that distinguish battery- and mains-powered nodes in routing, so that energy consuming tasks are carried out mostly by mains-powered nodes. We first propose a framework for a class of routing algorithms, which forms and uses a backbone topology consisting of all mains-powered nodes, including the sinks, and possibly some battery-powered nodes, to route data packets. We propose and evaluate a set of centralized algorithms based on this framework, and our simulation results show that our algorithms can increase network lifetime by up to more than a factor of two. We also propose a fully distributed power-source-aware backbone-based routing algorithm (PSABR) that favors mains-powered nodes as relay nodes. We validate and evaluate our distributed algorithm with extensive ns-2 simulations and our results show that the proposed distributed algorithm can enhance network lifetime significantly with a low control messaging overhead.

Besides wireless technology independent routing solutions, we also propose a technology specific power-source-aware routing solution (PSAR) for sensor and

ad hoc networks which use 802.15.4/ZigBee as the wireless technology. Our solution is fully distributed, tree-based, and traffic-adaptive. It utilizes some protocol specific properties of ZigBee, such as distributed and hierarchical address assignment, to eliminate battery-powered nodes on the routing paths as much as possible. To validate and evaluate our ZigBee-specific algorithm, we first implemented ZigBee extensions to ns-2 simulator and then implemented and simulated our protocol in this extended ns-2 environment. Our results show that the proposed algorithm operates efficiently and can increase network lifetime without increasing the path lengths significantly, compared to the default ZigBee routing algorithm.

Keywords: wireless sensor networks, network lifetime, routing, heterogeneous networks, backbone, power-source-aware, mains-powered, ZigBee, energy-efficiency.

ÖZET

KABLOSUZ ALGILAYICI AĞLAR İÇİN GÜÇ KAYNAĞI BİLİNÇLİ DEVİNGEN YOL ATAMA

Metin Tekkalmaz
Bilgisayar Mühendisliği, Doktora
Tez Yöneticisi: Doç. Dr. İbrahim Körpeoğlu
Temmuz, 2013

Kablosuz algılayıcı ağlar, belli bir amacı gerçekleştirmek için ilgilenilen bölgeye dağıtılmış algılayıcı düğümlerden oluşur. Bu düğümler çevresel ve fiziki şartları izleyerek toplanan veriyi belli bir merkezi düğüme gönderir. Kablosuz algılayıcı ağlar ile ilgili çoğu çalışmada algılayıcı düğümlerin gücünün ağ yaşam süresini sınırlayan değiştirilemez piller ile sağlandığı varsayılmaktadır. Diğer taraftan, algılayıcı düğümlerin en az bir kısmına güç sağlayabilecek ve böylece ağ yaşam süresini uzatabilecek şebeke elektriği, enerji hasadı mekanizmaları gibi daimi güç kaynakları da bulunmaktadır.

Burada amacımız pil ve şebeke elektriği ile beslenen farklı tipteki algılayıcı düğümlere sahip bu tarz heterojen kablosuz algılayıcı ağların yaşam süresini merkezi ve dağıtık yol atama algoritmaları kullanarak ve bu esnada fazla enerji gerektiren işleri şebeke elektriği ile beslenen düğümlere atayarak uzatmaktır. İlk olarak bir sınıf yol atama algoritması üretiminde kullanılabilecek bir çerçeve öneriyoruz. Üretilen algoritmaların ortak özelliği şebeke elektriği ile beslenen düğümleri, hedef düğümü ve gerekiyorsa pil ile beslenen bazı düğümleri içeren bir omurga oluşturarak veriyi aktarmak için bu omurgayı kullanmalarıdır. Simülasyon sonuçlarımız bu çerçeveyi kullanarak ürettiğimiz merkezi algoritmaların ağ yaşam süresinde iki kata kadar artış sağladığını göstermiştir. Ayrıca bu çalışmada veri iletimi için şebeke elektriği ile beslenen düğümlere öncelik veren güç kaynağı bilinçli, omurga temelli ve tam dağıtık bir yol atama algoritması da öneriyoruz. Önerdiğimiz bu algoritmayı geçerlemek ve değerlendirmek amacıyla ns-2 ortamını kullanarak elde ettiğimiz simülasyon sonuçları göstermektedir ki, algoritmamız düşük bir ek haberleşme yükü ile ağ yaşam süresini belirgin şekilde arttırmaktadır.

Bu çalışmada, bahsi geçen teknoloji bağımsız yol atama çözümlerimizin yanısıra, 802.15.4/ZigBee kablosuz ağ teknolojisine özel, güç kaynağı bilinçli, tam dağıtık, ağaç tabanlı ve trafiğe uyumlanabilen bir yol atama çözümü de önermekteyiz. Önerdiğimiz yol atama çözümü pil ile beslenen düğümlerin haberleşme yolları üzerinde yer almasını mümkün olduğunca önlemek amacıyla, dağıtık ve hiyerarşik ağ adresi atama mekanizması gibi ZigBee protokolüne özel yetenekleri kullanmaktadır. ZigBee teknolojisine özel algoritmamızı geçerlemek ve değerlendirmek amacıyla ilk olarak ZigBee protokolünün ihtiyaç duyduğumuz kısımlarını, daha sonra da önerdiğimiz algoritmayı ns-2 simülasyon ortamında gerçekledik. Hazırladığımız ns-2 simülasyon ortamını kullanarak elde ettiğimiz sonuçlar, ZigBee tanımında yer alan yol atama yöntemi ile karşılaştırıldığında, önerdiğimiz algoritmanın yol uzunluklarını arttırmadan ağ yaşam süresini arttırabildiğini göstermiştir.

Anahtar sözcükler: kablosuz algılayıcı ağlar, ağ yaşam süresi, yol atama, heterojen ağlar, omurga, güç kaynağı bilinçli, şebeke elektriği ile beslenen, ZigBee, enerji verimi.

Acknowledgement

First of all, I would like to express my sincere gratitudes to my supervisor Assoc. Prof. Dr. İbrahim Körpeoğlu. I am grateful for his invaluable support for all these long years, which made this thesis come true. His encouraging and positive attitude has always kept me walking.

I would like to thank to the thesis committee members Prof. Dr. Özgür Ulusoy and Dr. Defne Aktaş for their valuable comments for the past six years. I would also like to thank to the thesis jury members Prof. Dr. Tolga Mete Duman, Assoc. Prof. Dr. Uğur Güdükbay, and Assoc. Prof. Dr. Ahmet Coşar for kindly accepting to spend their valuable time and to evaluate this work.

I would like to express my appreciation to ASELSAN A.Ş. for the understanding and support during my academic studies. I also want to thank to people I work with for the joy they bring to my life making my work life together with academic life a pleasureful experience.

I would like to thank to my parents and grandparents for raising me with all their love. I would not be the person who I am without their never-ending support. I would also like to thank to my brother Sezgin. Despite the physical distance between us throughout our lives, he always cheers me up.

And most of all, my beloved wife Tuçe who has lived every stage of this long journey with me. Thank you for bearing with me for all this time. I cannot express how valuable your support has been to me, I love you. And my son Dorukhan Aral. I can only imagine being son of a daytime software developer and nighttime PhD candidate. I apologize for the time I have stolen from you for the last five years. And the newcomer of the family, my daughter Pelin. You are already five months old without a proper father. I promise to be a better father and husband from now on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A wireless sensor network (WSN) is a collection of nodes, which are capable of sensing, data processing, and wireless communication, distributed over an area of interest to accomplish a certain task. Nodes in a WSN monitor environmental and physical conditions and send the collected data to a special node, called sink or base station, usually in a multi-hop fashion. Therefore, the main purpose of a WSN is basically information gathering and delivery.

Typically, a sensor node consists of a processing unit, a wireless radio, and at least one sensor. Sensing capabilities of the nodes vary depending on the application, and different types of sensors are able to sense different physical phenomenon such as temperature, moisture, smoke, vibration, pressure, light, and sound. Continuous advances in wireless technology, sensing devices and low power electronics make WSNs a feasible solution for a wide range of applications. These applications include, but not limited to, habitat and environment monitoring, air conditioning control in buildings, collecting vital statistics of patients, controlling moisture level of soil in agriculture, industrial process monitoring and control, intrusion detection, target tracking, home automation, building evacuation in emergency conditions, and traffic control. Depending on the application type and coverage, number of nodes in a WSN can vary from a few to several thousands. While some of the applications require careful manual placement of nodes, in the others it might be possible to throw the nodes from air vehicles in large quantities.

Batteries have been the natural choice of power source in WSNs to facilitate ease of node deployment. On the other hand, limited energy of batteries restricts WSN lifetime. In a WSN composed of battery-powered nodes, replacing or recharging batteries is vital for the continuity of the operation. But most of the time, it is a costly process considering the high number of sensor nodes and worse, it might be impossible due to reasons such as safety in military applications or harsh environmental conditions. Historically, majority of the studies on WSNs assume that the nodes are battery-powered and extending the network lifetime has been one of the most studied subjects on WSNs. In general, the studies attack to a specific layer (e.g., physical, data link, network, transport) on the protocol stack, or they follow a cross-layer approach. Some of the studies try to increase the lifetime of individual nodes, and others try to increase the network lifetime as a whole.

Although most of the studies assume that sensor nodes are battery-powered, there are several studies showing that alternative energy sources exist [3][4][5]. Capacitors, fuel cells, heat engines, and beta voltaic systems are listed as energy sources with different capacities, that can be used to power sensor nodes, besides regular batteries. Although some of these energy sources last longer than the others, their energy depletes eventually as well. There are also alternatives to power sensor nodes continuously. One such alternative is to wirelessly transmit energy to the sensor nodes from a nearby energy-rich power source using radio frequency (RF) signals. A more promising perpetual power source alternative is energy harvesting, which basically is the process of converting ambient energy into usable electrical energy. Energy harvesting by itself is not a new research area, but its use in WSNs has gained popularity recently, as a result of the advances in the field enabling more efficient, smaller, and cheaper energy harvesting systems.

Mains electricity (power line) is another possible continuous power source alternative for sensor nodes. Although it seems to be in contradiction with the nature of wireless sensor nodes and networks, there is no reason for at least some of the sensor nodes not to benefit from the continuous energy source of the facility if they are already next to power lines. This is especially the case for indoor applications, where the nodes are deployed in houses, offices, buildings, factories,

etc. In such a scenario, mains-powered nodes would be preferred wherever possible to reduce maintenance costs, and battery-powered nodes would be used where installing power lines is costly or impractical.

In this thesis, our aim is to increase the lifetime of WSNs composed of nodes with heterogeneous power sources. We especially focus on WSNs composed of battery- and mains-powered nodes. But our study can easily be extended to WSNs in which some of the nodes have finite energy and the others are fed by continuous power sources, like energy harvesting systems. More details on possible continuous power sources are provided in Chapter 2. In the rest of the thesis, terms battery-powered and mains-powered are going to be used to refer class of nodes with finite and continuous power sources, respectively.

Our basic approach to increase the lifetime of WSNs is to make related algorithms power-source-aware in which energy consuming tasks, such as routing, coordination, and data processing, are mostly carried out by mains-powered nodes. Battery-powered nodes, on the other hand, are mainly responsible for sensing and sending their own data. We assume that all nodes have fixed and identical communication range without loss of generality. We also assume that the nodes are mostly stationary as expected in a WSN.

Firstly, we propose a framework for a class of routing algorithms. The reason we call it a framework is that some parts of it have several alternatives, therefore depending on the choice, different algorithms can be obtained. Our framework aims to form a backbone topology consisting of all mains-powered nodes, including the sinks. The backbone may also include battery-powered nodes, as required, if the mains-powered nodes do not form a connected topology. The resulting backbone is actually a tree rooted at the sink if there is a single sink, or a forest if there are multiple sinks. The remaining sensor nodes are connected to this backbone structure, either directly or through multiple-hops. The data packets produced by the sensor nodes are routed through this backbone structure towards the sinks.

Given a visibility (reachability) graph representing a WSN, where sensor nodes are the vertices and wireless links are the edges, the proposed framework first

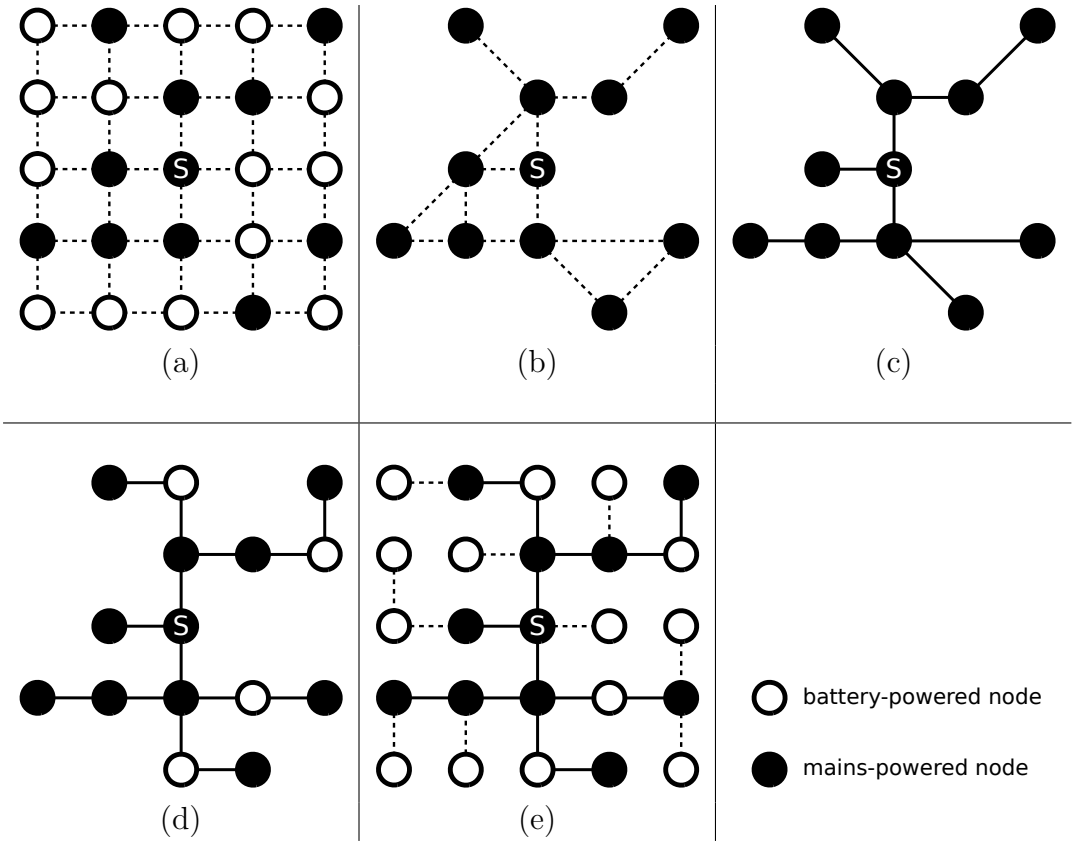

Figure 1.1: (a) Visibility graph, (b) secondary graph, (c) spanning tree on the secondary graph, (d) mapping to the original graph, and (e) routing tree.

reduces the original graph to a secondary graph: vertices representing the mains-powered nodes in the original graph constitutes the vertices of the secondary graph and there is an edge between the vertices of the secondary graph if the length of the shortest path between the corresponding vertices of the original graph is below a threshold. Then a spanning tree is found on the secondary graph. Using this spanning tree, the backbone on the original graph is determined, by replacing the edges in the spanning tree by the corresponding paths in the original graph. Finally, the rest of the nodes connect to the backbone to form a routing tree.

Figure 1.1 shows the graphs related to a sample network at different steps of the algorithm framework during its execution. Figure 1.1 (a) is the visibility

graph of the sample network in which the battery-powered nodes are represented by rings whereas the mains-powered nodes are represented by black circles. The sink is marked with an S. Figure 1.1 (b) is the secondary graph extracted by the algorithm framework. In this sample case, we assume the maximum hop distance between the mains-powered nodes used while extracting the secondary graph is 2. Figure 1.1 (c) depicts a spanning tree on the secondary graph and Figure 1.1 (d) shows the backbone obtained by mapping the spanning tree on the secondary graph to a tree on the original graph. Finally, Figure 1.1 (e) presents the routing tree obtained by connecting the rest of the nodes to the backbone.

In order to show the effectiveness of our approach and framework, we carried out extensive simulation experiments. In the experiments, we assumed that a central node (possibly the sink) has the global visibility graph, hence the backbone can be computed centrally. Our experiment results show that the centralized algorithms based on our proposed framework are able to increase the network lifetime by up to more than a factor of two, compared to the case in which battery- and mains-powered nodes are not differentiated.

Secondly, we present a power-source-aware backbone-based distributed routing algorithm (PSABR) based on the framework we propose. We give the detailed design of our distributed algorithm including all control messages and node behaviors in the form of pseudo-codes and finite state machines. We implemented the algorithm in ns-2 [6] network simulation environment completely, and run simulations to validate its correctness and robustness, as well as to measure its effectiveness and efficiency.

So far, our proposed methods are technology independent: any wireless technology providing one-hop wireless communication and an addressing mechanism to distinguish nodes can use our methods. But there are already several specific wireless communication technologies targeting especially WSNs such as 6LoWPAN [7], ZigBee [8], and Bluetooth Low Energy [9]. Algorithms designed for a specific wireless communication technology can benefit from the features the technology provides, but the algorithm should also comply with the restrictions and constraints the technology imposes, such as maximum number of neighbors,

types and roles of devices defined, address assignment mechanism and scheme, routing method, etc.

Thirdly, besides our wireless technology independent solutions, we also focused on and worked with a specific wireless technology, namely ZigBee, and applied our basic approach of distinguishing nodes based on their power sources while creating logical topologies and routing structures. The ZigBee standard is built on the IEEE 802.15.4 standard [10], which shares similar goals. ZigBee defines the application layer and the network layer, whereas IEEE 802.15.4 defines the medium access control layer and the physical layer. Both tree and mesh topologies are possible in a ZigBee network. The ZigBee standard defines a distributed and hierarchical address assignment mechanism and a routing method that depends on this addressing mechanism.

In our study related to ZigBee networks, we concentrate on tree topologies with the distributed address assignment mechanism enabled. We propose a distributed, dynamic, and traffic-load-aware routing algorithm to be used for sensor and ad hoc networks. The algorithm is fully distributed and forwards the packets according to the tree-based hierarchical addresses of ZigBee nodes. It considers traffic load in the network and in this way tries to use less number of battery-powered nodes on the paths between heavily communicating nodes. Each node in the network monitors the traffic it forwards and reconfigures the current topology if the data forwarded by battery-powered nodes would be less in an alternative topology compared to the current topology. Thanks to the distributed address assignment mechanism of the ZigBee standard, we can compute the addresses of the intermediate nodes given addresses of the source and the destination nodes, hence the nodes can compare the alternative topologies with minimal communication overhead.

In Figure 1.2, we present a sample reconfiguration scenario. Given a network as in Figure 1.2 (a), where the rings, circles, and edges represent battery-powered nodes, mains-powered nodes and the wireless links between the nodes, respectively and the PAN (personal area network) coordinator is marked with a C; assume that the initial routing tree is as shown in Figure 1.2 (b) where the edges

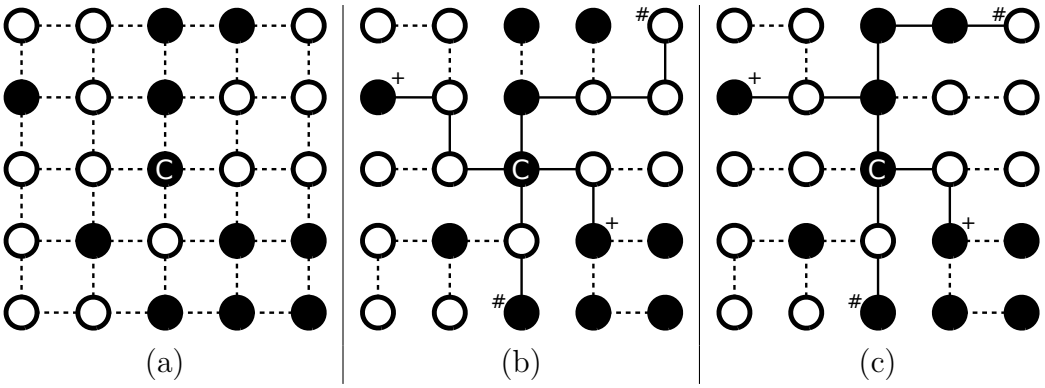

Figure 1.2: (a) Visibility graph, (b) initial routing tree, and (c) routing tree after reconfiguration.

represent the logical links of the routing tree. Also assume that the node pairs marked with + and # communicate with each other, hence the edges shown with solid lines represent the active links on the routing tree. In Figure 1.2 (b), number of battery-powered nodes on the routing paths is six. If we can reconfigure the routing tree as shown in Figure 1.2 (c), we can reduce the number of battery-powered nodes on the routing paths to three.

Our study on ZigBee networks differs from our previous studies in three respects. One, it is technology specific. Two, it assumes point-to-point communication is possible. Three, we do not assume a regular traffic flow between the nodes, that is, rate and end points of the data traffic can change over time. Due to these last two aspects, besides WSNs where traffic is usually many-to-one, it is also suitable for wireless ad hoc networks where traffic is peer-to-peer. It can also be used in wireless sensor and actuator networks (WSANs) where traffic can be both many-to-one and peer-to-peer. In WSANs, data may need to be gathered from all sensor nodes to the sink node and also may need to be transported between sensors and actuators. Hence, our ZigBee specific topology control and routing solution can be used for a broad range of ZigBee applications.

We evaluated the proposed ZigBee routing solution again using the ns-2 simulator. For that, we first implemented the ZigBee protocol in ns-2, on top of

the IEEE 802.15.4 implementation which is already available. Then, we embedded our protocol to ns-2 and performed extensive experiments. Our results show our routing protocol extends network lifetime compared to the default routing algorithm of ZigBee, which is not traffic-aware and which does not distinguish battery- and mains-powered nodes.

1.1 Contributions

The followings are the contributions of this thesis:

- We propose a backbone-based routing approach to increase WSN lifetime by distinguishing the sensor nodes according to their power sources.
- By following this approach, we propose an algorithm framework for WSNs in which battery- and mains-powered nodes coexist. The framework allows an energy-efficient, backbone-based routing algorithm to be obtained by selecting alternative sub-algorithms, depending on the application requirements.
- We propose a set of centralized backbone-based routing algorithms obtained using our framework. We showed via simulations that network lifetime can be increased by a factor of two using our algorithms, compared to using a basic algorithm that does not distinguish between battery- and mains-powered nodes.
- We also propose a distributed algorithm for constructing and maintaining a backbone-based routing structure that routes packets from sensor nodes to the sink in an energy-efficient manner and prolongs the network lifetime in this way. We describe the protocol messages used and also provide battery- and mains-powered node behaviors in detail and rigorously by means of pseudo-codes and finite state machines.
- To validate and evaluate our distributed algorithm, we developed a simulation environment based on ns-2 and a tool to visualize network topology

and routing relationships among the nodes. This tool can be used in similar studies with slight modifications. The simulation results show that our power-source-aware distributed routing algorithm can enhance network lifetime significantly compared to a basic shortest-path algorithm that does not distinguish battery- and mains-powered nodes.

- In addition to our wireless technology independent studies, we also propose a novel tree-based routing topology construction and maintenance algorithm for ZigBee wireless networks. Besides describing our algorithm in every detail, we also show a method to compute the ZigBee network addresses of intermediate nodes between any given source-destination pair, that depends on the distributed address assignment mechanism of ZigBee. Our routing algorithm can utilize mains-powered nodes to reduce packet and energy load on battery nodes. It is also adaptive to changes in traffic patterns.

- To evaluate the performance of our tree-based ZigBee routing algorithm, we again implemented an ns-2 simulation. To do this, we partially implemented the network (NWK) layer of ZigBee into ns-2 environment. Hence, we contributed a new module to ns-2.

- The set of solutions we provide in this thesis are analyzed considering different aspects such as:

  - centralized vs. distributed algorithms,
  - single vs. multiple sinks,
  - generic vs. technology-specific approaches,
  - continuous dissemination vs. event-driven communication,
  - sensor to sink vs. point-to-point communication.

1.2 Outline of the Thesis

Organization of the thesis is as follows. In the next chapter we give the related studies in the literature. In Chapter 3, we present a framework for a class of algorithms that basically forms a backbone, mainly composed of mains-powered nodes, to route the data packets. In Chapter 4, we describe a distributed algorithm based on this framework in detail. We propose a distributed algorithm specific to ZigBee networks, which shapes the network topology in order to reduce the data forwarded by the battery-powered devices in Chapter 5. Finally, in Chapter 6, we summarize accomplishments of the thesis along with possible future research directions.

Chapter 2

Related Work

In this chapter, we describe the previous work related to our study on power-source-aware routing in heterogeneous WSNs. We first give studies on energy conservation in WSNs since our aim is to increase the network lifetime by careful use of limited energy of battery-powered nodes. Next, we present studies related to routing in general and backbone routing in particular in wireless ad hoc and sensor networks, since our studies focus on routing and employ a backbone-based routing approach. We also describe some of the studies related to power-source types in WSNs including energy harvesting and discuss studies exploiting node heterogeneity similar to our study. Finally, we present studies related to our ZigBee-specific adaptive routing method that distinguishes nodes with respect to their power-source types to increase network lifetime.

2.1 Energy Conservation

Due to the limited energy resources and mostly unattended nature of WSNs, efficient use of energy to increase the network lifetime has been one of the most studied subjects related to WSNs [11].

In a recent study, Anastasi et al. [12] provide a taxonomy for energy conservation techniques in WSNs and present a survey on the related studies based on this taxonomy. At the highest level, the authors classify studies into one of the following categories: approaches employing duty-cycling, data-driven approaches, and mobility-based approaches. The duty-cycling based studies are further classified into two subcategories: topology-control based studies and connection-driven studies. In studies belonging to the topology control subcategory, node redundancy is exploited to cover the area as in [13] and [14] or provide the network connectivity using a subset of the nodes as in [15], [16], and [17]. Our study falls into the connection-driven subcategory, since we exploit the redundancy of the nodes in the network to route data packets in an energy-efficient manner. In [12], data-driven approaches are further divided into data reduction [18][19][20] and energy-efficient data acquisition [21][22][23] subcategories. Although we assume that data aggregation (which is a data reduction technique) is possible in our simulations, it is not an integral part of our proposed solutions and our schemes can be used in environments with or without data aggregation.

In another study, Gupta et al. [24] also investigate energy efficiency in WSNs. Authors first discuss methods to reduce energy consumption that can be employed at the MAC (media access control) layer, such as avoiding collisions and overhearing, but the main focus of the study is network layer protocols. Effective routes and efficient route setup and maintenance are given as means of energy efficient routing in WSNs. In our study, we aim at constructing effective routes from the viewpoint of battery-powered nodes, but we also consider efficiency of the route construction.

Yet in another work dealing with energy efficient strategies in WSNs [25], related studies are grouped into four: energy efficient routing [26][27][28], scheduling the nodes' sleeping state [29][30], topology control by tuning node transmission power [31][32], and reducing the volume of information transferred [33][34]. In some respects [25] resembles similarities with the grouping given in [12], but differently it considers routing in a separate category. We discuss routing in WSNs in a broader sense in Section 2.2, but here we focus on the energy efficiency as far as routing in WSNs concerned. Multipath routing is given as one of the strategies

to prolong lifetime and employed in studies like [26], [35], [36], and [37]. In our study we take multipath routing into account not as end-to-end multipath routing but rather as multipath routing between mains-powered nodes. By using this method we aim at balancing energy usage of battery-powered nodes connecting mains-powered nodes.

Adaptive hop-by-hop routing is given in [25] as another strategy for energy efficient routing in WSNs. Studies like [27] and [38] try to select paths consuming minimum energy whereas studies like [28] try to use paths with nodes having the highest residual energy. There are also studies which apply a hybrid of these two methods such as [39]. Favoring paths with minimum energy requirement has the disadvantage of depleting energy of common nodes residing on multiple paths. On the other hand, using residual energy while deciding the routing paths requires additional information exchange and can increase the amount of control messages. In some of the studies applying cluster-based routing, cluster heads are elected according to residual energy of the nodes either by one-hop message exchange or by probabilistic methods [40][41][42], to minimize or eliminate message exchange. But they are prone to connectivity problems unless additional precaution is taken, such as increasing node density.

2.2 Routing

Al-Karaki and Kamal [43] investigate different aspects of routing in WSNs and present a survey on the related studies. Authors categorize the routing protocols into three groups according to the network structure. In flat routing, sensor nodes take equal role in routing in the network as in [44][45][26]; in hierarchical routing, a group of the nodes take special role, such as cluster-heads, and coordinate communication between the regular nodes and the sink as in [46][47][48][49]; and finally in location based routing, nodes are addressed based on their locations and data packets are routed accordingly as in [15][50][51]. Hierarchical routing methods facilitate scalability and energy efficiency and have better potential to exploit node heterogeneity. Our proposed method based on backbone routing

best fits into the hierarchical routing category and has two levels of hierarchy: nodes forming the backbone, which are mostly mains-powered, and rest of the nodes, which are battery-powered.

As Simplot-Ryl et al. enumerate in [52], backbone-based approaches for data dissemination and gathering are rather well-studied. As in other related studies, in [52], backbone is considered to be either neighbor- or area-dominating set of a network. In the former, all nodes are either part of the backbone or in one-hop distance of it, and in the latter, the whole area is in 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 or distributed heuristics. Although centralized algorithms can provide bounds on the size of CDS, such as in [53], they require global information, increasing the messaging overhead.

Localized backbone-based approaches, in which only a limited neighborhood information is shared, are based on either deterministic or probabilistic algorithms. Span, presented in [15], is an example of probabilistic algorithms. In Span, 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 [54], Boukerche et al. try to find a spanning tree with many leaf nodes. In EAD, nodes with higher residual energy have a higher chance of not being a leaf-node. As another distributed algorithm, in ASCENT [16], nodes participate in sensing and routing tasks according to the packet-losses due to lack of relay nodes and packet-losses due to 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 different studies including [13] and [55]. 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 these studies is that they need to know the locations of the nodes. In the studies mentioned so far, the aim is to find a CDS. Differently in [56], the authors present different protocols that ensure k-connectedness of dominating sets, for the sake of fault tolerance. In a different study that take fault tolerance into account,

Kashyap et al. [57] add relay nodes to a WSN in order to provide a k-connected backbone.

In our study, different from the previous studies based on backbone construction, we assume that the sensor nodes are heterogeneous as far as their power sources are considered. Although studies, such as [53] and [15], that take residual energy into account, can be applied for this case, prior knowledge of different power-source types enables specialized solutions, since the energy of the nodes change in time but their power-source types do not. In this study, we also adapt the definition of backbone: in our case, a backbone consists of a connected set of mains-powered nodes, compared to the CDS of all nodes. As mentioned earlier, there are both centralized and localized algorithms for backbone construction. Both centralized and distributed algorithms, based on the approach presented in this thesis, are possible. We propose several centralized algorithms in Chapter 3, and we propose a distributed algorithm in Chapter 4. In any case, our solutions fall into the deterministic category. That means, if there is a connected backbone, our proposed approach is able to construct it, whereas in randomized algorithms backbone connectivity is highly affected by the node density. Our proposed approach can also take fault tolerance into account similar to [56] and [57], but different from them our proposed algorithms try to increase the number of vertex disjoint paths between a pair of mains-powered nodes on the backbone, rather than trying to achieve k-connectedness of the whole backbone. As another difference with [57], we assume that the locations of the sensor nodes are fixed.

2.3 Power Sources

In most of the studies concerning WSNs, nodes are assumed to be battery-powered. But there are several studies showing that alternative energy sources exist. Fuel-cells, heat engines, energy harvesting methods as well as power distribution techniques (e.g., through use of radio frequencies, acoustics, light, etc.) are discussed in [3], [4], and [5], besides batteries and power lines. Some of these power-source types provide energy for a limited time similar to batteries, whereas

others, such as energy harvesting methods, have potential to provide a continuous source of energy. Therefore, although studies presented in this thesis are originally designed for networks consisting of battery- and mains-powered nodes, they can be used in similar heterogeneous deployment cases as far as the energy sources are considered. Using a technique similar to the one presented in [58], nodes employing our approach 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., [59], [60], [61], and [62]), since such methods have great potential to decrease maintenance costs due to battery replacement and to greatly extend network lifetime for cases where replacing batteries is practically impossible. In these studies, different renewable energy sources, such as sun, wind, vibration, heat, electromagnetics, are considered for energy harvesting. In general, harvesting provides intermittent energy, due to the unreliable nature of the sources and the effectiveness of the harvesting. But different approaches can be employed to increase the reliability of this method. One approach is to use multiple sources of energy as in [63] and [64]. Another approach is to harvest energy from relatively more reliable ambient energy sources such as fluorescent lamps in hospitals or factories, where the lights are always on, as in [65], or air flow near ventilation exhausts, as in [66]. As long as some of the nodes can be powered by such reliable methods, we can make use of those special nodes to increase the network lifetime.

2.4 Heterogeneity

Basically, our approach exploits the nodes' heterogeneous power sources in a WSN. There are other studies that make use of superior nodes to increase WSN lifetime. Yarvis et al. [67] show that even a modest number of mains-powered nodes has 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

the nodes can increase the network lifetime, our study does not rely on such arrangements. In [68], Ma et al. present a cluster based topology formation and update protocol which takes the energy resources of the nodes into account. Unlike this study, we assume that all nodes have the same communication range; that means in our solutions all nodes, whether ordinary or superior, can use the same wireless communication technology. In [69] and [70], authors consider the heterogeneity of the nodes as far as their energy harvesting capabilities are considered. Kansal et al. [69] propose a routing method that makes use of nodes with higher harvesting potential. Voight et al. [70], on the other hand, describe a modified directed diffusion approach in which solar powered nodes are taken into account.

2.5 ZigBee

In this thesis we also propose a power-source-aware routing algorithm for ZigBee networks. Here, we first give some energy conservation related studies specific to ZigBee, then we present some studies on topology control and routing in ZigBee networks.

There are several studies on increasing energy efficiency, hence the lifetime, of ZigBee networks. [71] provides a survey of ZigBee networks as sensor networks and includes a section on energy efficiency. As presented in [71], energy-efficiency related approaches for ZigBee networks are realized in different layers of the protocol stack.

Suarez et al. in [72] replace the MAC protocol of ZigBee with X-MAC. Cho et al. in [73] adapt the beacon interval dynamically based on the arrival rate of packets in order to increase the sleep time of the nodes. In a similar study, Kim et al. [74] present the impact of adaptive superframe duration as well as beacon interval. Li et al. in [75] exploit multiple sleep/wake-up schedules as opposed to the single beacon interval of ZigBee, to conserve energy.

Piccunelli et al. in [76] present a strategy to build a routing tree based on

a cross-layer cost function incorporating remaining energy, channel quality, and number of hops. Similarly, Boughanmi et al. in [77] use a cost function in order to satisfy the energy and delay constraints of the paths to be used. This modified function is used at the path discovery phase of ZigBee. Unlike studies in [76] and [77], Peng et al. in [78] use the two ZigBee routing methods presented in the ZigBee standard as they are, but choose one of them according to the data service that requires routing functionality. In some studies, such as [79] and [80], multi-path routing is exploited in order to prolong the network lifetime.

In some studies, topology control is also applied in ZigBee networks to increase network lifetime. Ma et al. in [81], for example, propose an algorithm to construct network topologies with a small number of coordinators while still maintaining network connectivity. The average duty cycle is reduced and the battery life is prolonged by reducing the number of coordinators.

There are two different routing mechanisms, i.e., hierarchical tree routing and modified ad hoc on-demand distance vector routing (AODV), specified in the ZigBee standard and there are several studies analyzing and comparing these two mechanisms (e.g., [82], [83], and [84]). Cuomo et al. in [82] show that hierarchical tree routing performs better than AODV in terms of packet loss, energy consumption, and delay. Hierarchical tree routing exploits the information exchanged during the topology formation to achieve its superior performance. Although hierarchical tree routing is superior in certain scenarios, AODV provides a more generic routing solution, especially in relatively dynamic networks. Furthermore, hierarchical tree routing uses relatively longer paths compared to AODV. Studies in [85] and [86] make use of neighboring nodes, which are neither parents nor children of the current node, to enhance hierarchical tree routing performance by shortening the paths.

The main difference between the studies related to ZigBee mentioned so far and our ZigBee specific study presented in this thesis is that our study distinguishes between mains- and battery-powered devices in order to modify the network topology. Unlike residual energy, power-source type information does not

change over time. Hence, messaging required to share the energy levels is eliminated. Furthermore, studies that propose ZigBee routing strategies are generally for mesh topology networks whereas our algorithm focuses on tree topology networks, which is natural to consider for WSNs. Hence, our algorithm makes use of advantages provided by hierarchical tree routing while eliminating the inefficient battery usage due to relatively longer paths.

Chapter 3

An Algorithm Framework for Power-Source-Aware Routing in Wireless Sensor Networks

Wireless Sensor Networks (WSNs) are used to monitor physical and environmental conditions in a wide range of civilian and military applications, as stated before. We believe in many WSN applications at least a portion of the nodes can be mains-powered. This is especially the case for indoor applications. On the other hand, in some applications, clever use of energy-harvesting methods can provide continuous and reliable energy to some of the sensor nodes in a WSN, as discussed in Chapter 2.

In this chapter, we present an energy efficient routing approach that increases the lifetime of heterogeneous WSNs in which nodes with different power-source types coexist. We make the following assumptions about a sensor network:

- a node is either battery- or mains-powered,
- all sensor nodes have the same communication range,
- all sensor nodes have periodic data to send,

- data flow is from the sensor nodes to the sink,
- nodes are stationary,
- nodes are located randomly.

The basic idea behind our proposed approach is to form a backbone structure consisting of mains-powered nodes and the sink to relay the packets from sensor nodes to the sink. However, the sink and the mains-powered sensor nodes might not always form a connected topology. Therefore, some battery-powered nodes can also be used to form a connected backbone for the rest of the network. The proposed approach is presented as an algorithm framework defining a class of routing algorithms.

The remainder of this chapter is organized as follows. We describe our proposed approach and framework in Section 3.1. We present four sample centralized algorithms based on our framework in Section 3.2. In Section 3.3, we give the simulation results presenting the performance of our centralized algorithms. Finally, in Section 3.4, we conclude the chapter.

3.1 Our Routing Algorithm Framework

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 mains-powered sensor nodes (which are accessible from the sink) and some of the battery-powered nodes to interconnect mains-powered nodes, if required. The remaining battery-powered nodes are connected to this backbone structure. Then this backbone structure is used to route packets from all sensor nodes, battery- or mains-powered, to the sink node.

Figure 3.1 shows a sample network to explain the proposed approach. This

network consists of 500 sensor nodes, 100 of which, including the sink, are mains-powered. The nodes are distributed to a square-shaped area with an edge length of 10 units. Communication range of each node is 1 unit. The sink is located at the center of the area. In the figures, the battery-powered nodes are denoted by small circles and the mains-powered nodes are shown as larger circles. Figure 3.1 (a) shows the visibility graph of the network; if a node is in the communication range of another, there is an edge between the vertices representing these two nodes. We assume links are symmetric. Given such a visibility graph, our approach can extract a backbone similar to the one shown in Figure 3.1 (c). Connectivity information of the mains-powered nodes, reduced to a spanning tree as in Figure 3.1 (b), is used to form the backbone, which is explained later in this section. Please note that all mains-powered nodes take part in the backbone, and in some cases battery-powered nodes are used to interconnect them. Finally, Figure 3.1 (d) 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 $p = (v_1, v_2, ..., v_n)$ between v_i and v_j in G s.t. $v_1, v_2, ..., v_n$ 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.

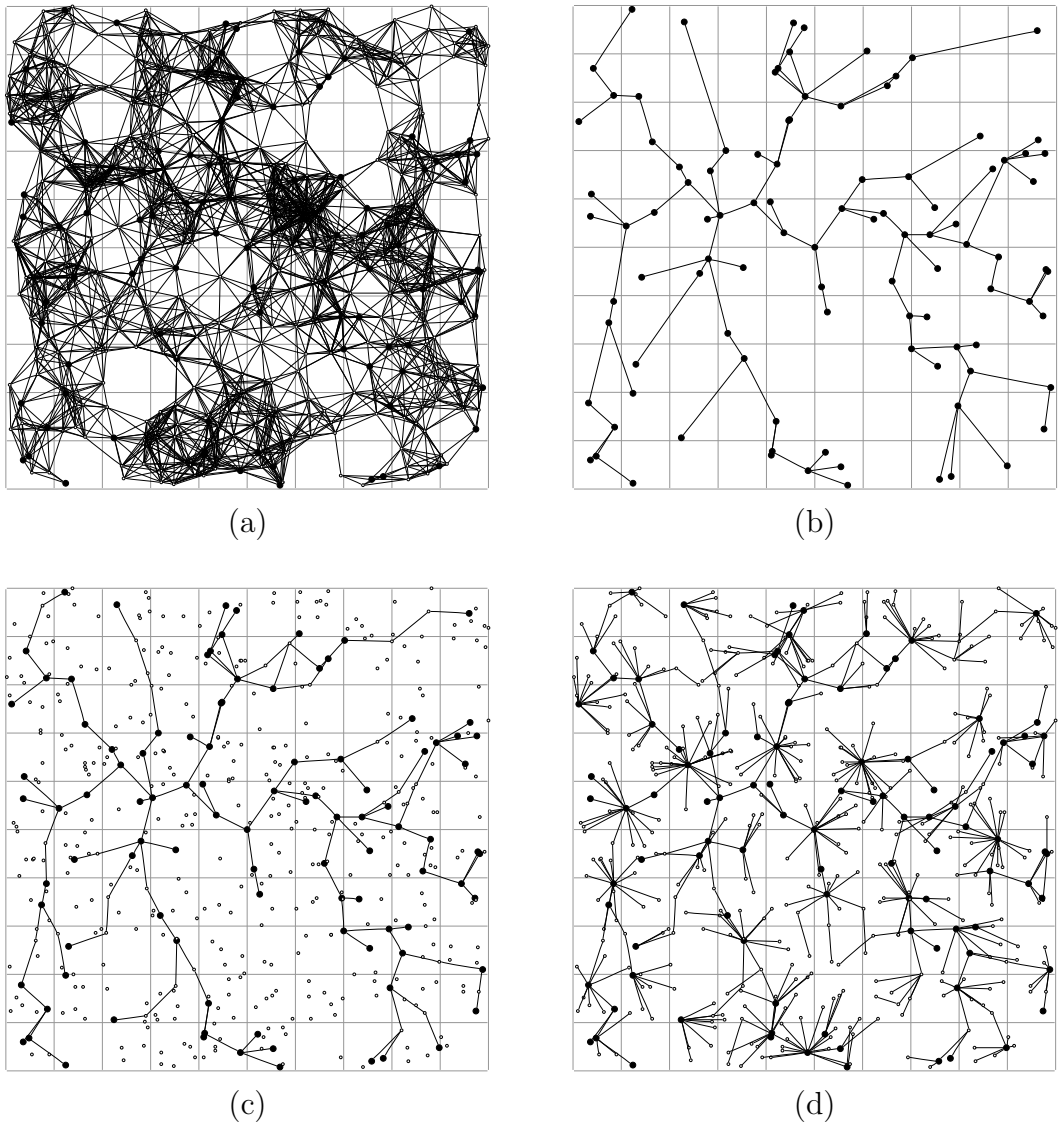

Figure 3.1: (a) Visibility graph, (b) a spanning tree on the mains-powered node connectivity graph, (c) backbone, and (d) routing tree graph (reprinted from Fig. 1 of [1] © 2010 IEEE).

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 Step 1, the original network visibility graph is reduced to a secondary graph in which the vertices are the mains-powered nodes (Step 1-a) 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 simple path between them shorter than or equal to a threshold T and consisting of only battery-powered nodes (Step 1-b).

In Step 1-c, cost values are assigned to the edges of the secondary graph to be used in Step 2 of the procedure. 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 between the two mains-powered nodes and 2) A value inversely proportional to the number of vertex disjoint paths (shorter than or equal to a threshold, T, and consisting of only battery-powered nodes) between the two mains-powered nodes. The first method is expected to reduce the amount of energy consumed by the battery-powered nodes, whereas the second method 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. Instead, the alternative paths between the mains-powered nodes can also be used for load balancing by sending each packet through a different alternative path.

In Step 2 of the procedure, the backbone is formed. First a spanning tree on the secondary graph is found (Step 2-a), similar to the one in Figure 3.1 (b). 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) rooted at the sink are considered as alternatives. Although an MST is expected to give better network-wide results than an SPT, especially when data is aggregated, 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 original

graph (Step 2-b), which corresponds to mapping each edge of the spanning tree on the secondary graph to a path on the original graph. End points of the paths are the vertices representing mains-powered nodes and nodes on the paths are the battery-powered nodes connecting the corresponding mains-powered nodes. In the algorithms given in Section 3.2, the paths are chosen to be the shortest simple paths connecting mains-powered nodes and there might be no battery-powered nodes on the paths if mains-powered nodes are immediate neighbors. The mapping can be seen in Figure 3.1 (b) and (c). After these first two steps, the backbone is formed.

Finally, in the last step of the procedure, the remaining battery-powered nodes, i.e., the nodes are not connected to the backbone yet, are connected to the nodes that are part of the backbone (Step 3), either directly or over multiple-hops. Although there might be other alternatives, in the algorithms presented in Section 3.2, mains-powered nodes are chosen to have precedence over battery-powered nodes to be parents of the connecting nodes. The final routing tree is similar to the one in Figure 3.1 (d).

So far the algorithm framework and properties of possible concrete algorithms are described but their implementations are not explained. Centralized implementation is one of the alternatives. In the centralized implementation, each node sends its neighbor list to the sink and in this way the sink obtains the complete network topology information. The sink then executes the centralized algorithm and sends back the final connectivity information to the nodes according to the algorithm results. Whenever neighborhood information of a node changes, the sink is informed about the situation and the topology is restructured according to the current visibility of the nodes, if required. More detail on centralized implementation is provided in Section 3.2.

Distributed implementation is another alternative. Although both MST and SPT have distributed implementations available in the literature [87][88], finding an MST of a graph in a distributed manner is more complex than finding an SPT. In a distributed implementation of SPT case, local information, which includes neighboring mains-powered nodes (other mains-powered nodes connected directly

or through a limited number of battery-powered nodes) and the path alternatives to them, can be collected at mains-powered nodes. Hence each mains-powered node can have a partial view of the visibility graph. In order to construct an SPT, distance to the sink can be shared between connected mains-powered nodes. Once a mains-powered node determines its parent, which has the minimum distance to the sink among its mains-powered neighbors, it may also decide the path to its parent among the alternatives. This corresponds to a partial mapping of the secondary graph to the original graph handling part of the step 2-b of the procedure. Nodes that are not declared as part of the backbone by the mains-powered nodes can be connected to the backbone as in the last step of the procedure. More detail on distributed implementation is provided in Chapter 4.

3.2 Sample Centralized Algorithms Based on Our Framework

In this chapter, we mainly focus on the centralized implementation of the algorithms based on our proposed framework. Before going into the details of the centralized algorithms, let us first clarify some of the terms that we use in the algorithm descriptions. Two nodes are *neighbors* if they are within communication range. The *peer* of a mains-powered node is another mains-powered node (possibly the sink) that is reachable through less than T battery-powered nodes, where $T > 0$. 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.

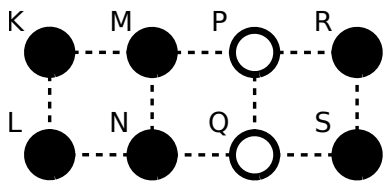

Figure 3.2: A portion of a sample wireless sensor network.

In Figure 3.2, we give a graph representing a portion of a sample WSN where

ring-shaped vertices denote battery-powered nodes, solid-circle-shaped vertices denote the mains-powered nodes, and the edges denote the wireless links between the nodes. In the sample network, neighbors of node M are K, N, and P. Assuming the threshold T is 2, peers of M are K, N, and R: K and N are immediate neighbors of M and there is a path between R and M whose length is less than or equal to T (i.e., 2) and consisting of only battery-powered nodes. M and S are not peers since shortest path between these two nodes consisting of only battery-powered nodes is 3, which is greater than T. Similarly M and L are not peers: although they are 2 hops away, there is no path between them consisting of only battery-powered nodes. Assuming the sink is S and M sends its data to the sink through P and R, parent of M is R.

Based on our algorithm framework, we propose four different algorithms that aim at constructing a backbone structure to route the data traffic through, as listed below.

1. Algorithm "# of BP-Nodes + MST",
2. Algorithm "# of Disjoint Paths + MST",
3. Algorithm "# of BP-Nodes + SPT",
4. Algorithm "# of Disjoint Paths + SPT".

The first two algorithms construct the spanning tree over the secondary graph as a minimum spanning tree (MST) and the latter two algorithms construct the spanning tree as a shortest path tree (SPT). Furthermore, the first and the third algorithms use the minimum number of battery-powered nodes connecting peers (i.e., mains-powered node pairs) as the edge costs on the secondary graph, and the second and the fourth algorithms make use of a value inversely proportional with the number of vertex disjoint paths connecting peers to assign costs to edges. More algorithms can be derived from our algorithm framework by considering different cost functions or different tree formation algorithms. We propose and analyze these four specific algorithms in this chapter.

Algorithm 3.1 Construct the backbone

function ConstructBackbone($G(V, E)$, ps)

1: {find peers and all possible paths between them}
2: $paths \leftarrow$ FindPeers($G(V, E)$, ps)
3: {reduce G to G' and compute the edge costs of G'}
4: $G'(V', E', cost) \leftarrow$ ReduceGraph($G(V, E)$, ps, $paths$)
5: {given G' and $cost$, find a spanning tree, that is G''}
6: $G''(V'', E'') \leftarrow$ FindMST($G', cost$)
7: {map G'' back to a tree on G to obtain $\bar{G}(\bar{V}, \bar{E})$}
8: **for all** edge $(u, v) \in E''$ **do**
9: $\quad \bar{V} \leftarrow \bar{V} \cup u \cup v$
10: $\quad path \leftarrow argmin_l(|l|, \forall\ l \in paths_v^u)$
11: $\quad last \leftarrow u$
12: $\quad$ **for all** node n on path l **do**
13: $\quad\quad \bar{V} \leftarrow \bar{V} \cup n$
14: $\quad\quad \bar{E} \leftarrow \bar{E} \cup (last, n)$
15: $\quad\quad last \leftarrow n$
16: $\quad$ **end for**
17: $\quad \bar{E} \leftarrow \bar{E} \cup (last, v)$
18: **end for**
19: **return** $\bar{G}(\bar{V}, \bar{E})$

In Algorithms 3.1, 3.2, and 3.3, we give the pseudocodes for the centralized implementation of one of our four algorithms, the "# of BP-Nodes + MST" algorithm. The pseudocodes assume that the visibility graph $G(V, E)$ of the network and the power-source types of the nodes ps are already available at the sink. Note that, these pseudocodes can easily be modified to implement the other algorithms, besides "# of BP-Nodes + MST" algorithm.

In Algorithm 3.1, which constructs the routing backbone, first peers (mains-powered nodes) and possible paths between them are computed. The details of this step are given in Algorithm 3.2. In the algorithms, $paths$ is a data structure such that $paths_v^u$ stores the list of all possible paths between mains-powered nodes u and v, and each path between u and v is a sequence of battery-powered nodes, excluding the mains-powered end points (i.e., u and v). In Algorithm 3.2, a queue of node sequences (i.e., paths), Q, is used to explore all possible paths between the mains-powered nodes. For each mains-powered node m, first a sequence containing only the mains-powered node itself is pushed to the queue Q. Then,

until Q is empty, sequences are popped and according to the power-source types of each newly explored node, either a new path is discovered to a mains-powered node and added to *paths* (line 11) or a new sequence is obtained (by adding the new node to the current sequence) and pushed to the queue (line 8).

Algorithm 3.2 Find peers and paths between them

function FindPeers($G(V, E)$, ps)

```
 1: for all mains-powered node m ∈ V do
 2:    push(Q, [m])
 3:    while Q ≠ ∅ do
 4:       L ←pop(Q)
 5:       for all v adjacent to front(L) in G do
 6:          if ps_v = battery-powered then
 7:             if (|L| < T) ∧ (v ∉ L) then
 8:                push(Q, L + v)
 9:             end if
10:          else if v ≠ m then
11:             paths_v^m ← paths_v^m ∪ (L − m)
12:          end if
13:       end for
14:    end while
15: end for
16: return paths
```

Next in Algorithm 3.1, the original visibility graph is reduced to a secondary graph, whose details are given in Algorithm 3.3. In Algorithm 3.3, vertices are added to the secondary graph for all mains-powered nodes (line 3), and edges are added for all mains-powered node pairs if there is path between them stored in *paths* (line 7). For each edge added, a cost value is also computed. Here, the length of the shortest path between the nodes in the original graph is chosen as the cost metric (line 8), but other alternatives can easily be adapted.

Then, in Algorithm 3.1, a spanning tree on the secondary graph is computed using the minimum spanning tree algorithm (line 6, details are not provided). Note that, here the minimum spanning tree algorithm can be replaced by a spanning tree algorithm of choice. In the last part of Algorithm 3.1 (lines 8-18), the spanning tree on the secondary graph G', that is G'', is mapped back to a tree on the original graph G, and therefore the backbone, $\bar{G}$, is obtained. While

Algorithm 3.3 Reduce graph

function ReduceGraph($G(V, E)$, ps, $paths$)

```
1: for all u ∈ V do
2:    if ps_u = mains-powered then
3:       V' ← V' ∪ u
4:       for all v ∈ V do
5:          if ps_v = mains-powered then
6:             if paths_v^u ≠ ∅ then
7:                E' ← E' ∪ (u, v)
8:                cost_(u,v) ← |argmin_l(|l|, ∀ l ∈ paths_v^u)|
9:             end if
10:         end if
11:      end for
12:   end if
13: end for
14: return G'(V', E', cost)
```

mapping the edges of G'' to the paths on G, shortest possible paths between the mains-powered nodes are chosen (line 10). Note that the mains-powered nodes (line 9) are added to the backbone as well as the battery-powered nodes (line 13) connecting these mains-powered nodes.

3.3 Evaluation of the Proposed Approach

In order to evaluate the effectiveness and the performance of our approach and its alternative algorithms obtained by selecting different sub-algorithms and cost functions, described in Sections 3.1 and 3.2, we implemented a custom simulator in C++ and run a set of simulation experiments. In our simulations, our power-source-aware backbone-based routing approach is compared with a basic shortest path tree routing algorithm, in which battery- and mains-powered nodes are not distinguished and each node is connected to the sink via the shortest possible path. For our backbone-based routing approach, either the minimum number of battery-powered nodes or $1/d$, where d is the number of vertex-disjoint paths bounded by a threshold T, is chosen to be the cost value assigned to the edges

of the secondary graph (referred as *# of BP-Nodes* and *# of Disjoint Paths*, respectively). T is chosen to be 4, and although polynomial time algorithms exist to find the maximum number of vertex-disjoint paths for $T \leq 4$, a greedy approach is followed to find a lower bound on the number of vertex-disjoint paths. On the other hand, either a minimum spanning tree (MST) or a shortest path tree (SPT) algorithm is applied to find a spanning tree on the secondary graph.

In the simulations, the network size is chosen to be 500 sensor nodes, a certain ratio of which is mains-powered. The sensor nodes are distributed over an area of 500 m by 500 m and the communication range of a node is set to be 50 m, independent of its power-source type. The sink is located at the center of the area and it is mains-powered. The simulation results are averaged over 20 runs, in each of which the locations of the sensor nodes are determined pseudo-randomly as described in [89]. A sensor node is assumed to be reachable if it is alive and there is a path between the node and the sink. Each simulation run is stopped when more than half of the sensor-nodes (i.e., 250 sensor nodes) become unreachable. In the simulations, we assume that each sensor node sends data to the sink periodically. At each round of data gathering, data is collected once from each sensor node. We also assume that the data is aggregated at each node.

A simple energy consumption model is applied in the simulations. In [90], the energy consumption ratio of a wireless device is measured as 1:1.05:1.4 for idle, receive, and send periods, respectively. The energy consumption of the sensor nodes is determined in accordance with this measurement. Hence, receiving a data packet is assumed to consume 1.05 units of energy, whereas sending a data packet is assumed to consume 1.4 units of energy. Idle periods of sensor nodes are ruled out, since total idle period of a node is almost equal for any approach compared in the simulations. Furthermore, the energy consumed during data aggregation is assumed to be negligible. In each simulation run, all the battery-powered nodes start running with 1000 units of energy.

Figure 3.3 depicts the number of reachable nodes with respect to number of rounds since the start of data gathering. We express the time and network lifetime in number of rounds. As the figure shows, our backbone-based routing approach,

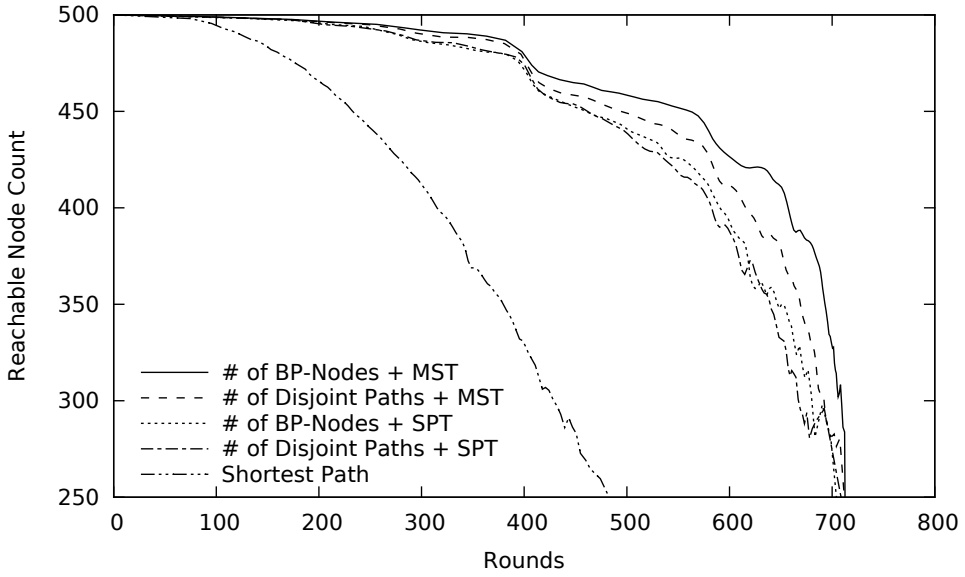

Figure 3.3: Number of rounds passed vs. number of nodes reachable from the sink (mains-powered node ratio: 20%) (reprinted from Fig. 2 of [1] © 2010 IEEE).

with its four different versions, exhibits a significant improvement over the basic shortest path routing algorithm. Note that if the network lifetime is defined as the number of rounds until 20% of the nodes (i.e., 100 nodes) become unreachable, our algorithms provide network lifetime more than twice as long as the shortest path algorithm. Also note that, MST performs better than SPT as the spanning tree algorithm and *# of BP-Nodes* performs better than *# of Disjoint Paths* as the cost function, as far as the network lifetime is concerned.

Figure 3.4 presents the average energy consumption of a battery-powered node (averaged over all battery-powered nodes) during the lifetime of the network. Our backbone-based approach has an average energy consumption of around 1.5 units per node per round. Considering that each node must transmit exactly once at each round, thanks to data aggregation, and one packet transmission consumes 1.4 units of energy, this means that packets are mostly relayed by mains-powered nodes. As the number of rounds increases, the average energy consumption per node decreases for the basic shortest path algorithm. The reason for this is the increase in the ratio of mains-powered nodes among the reachable nodes while the battery-powered nodes deplete their energy, which in turn decreases the burden

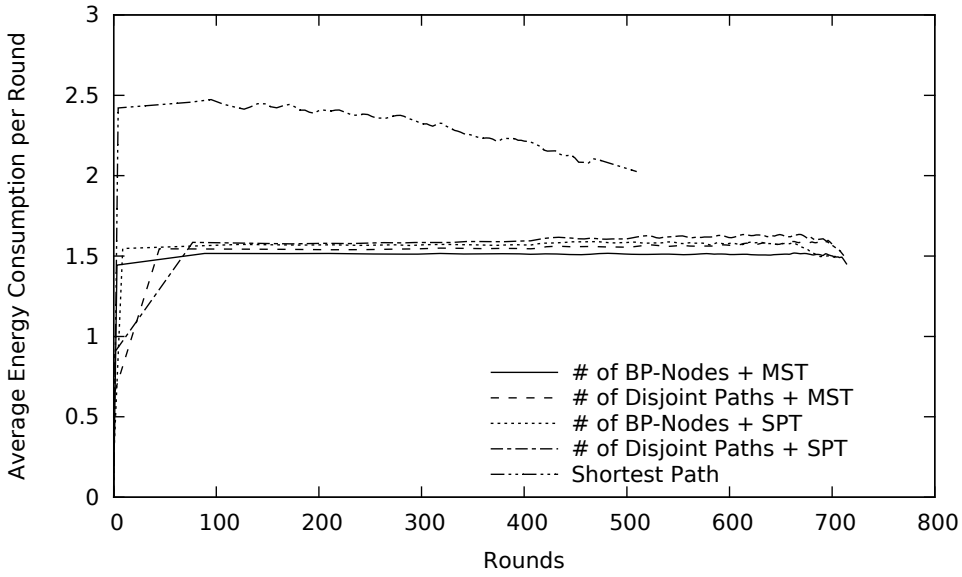

Figure 3.4: Number of rounds passed vs. average energy consumption per round (mains-powered node ratio: 20%) (reprinted from Fig. 3 of [1] © 2010 IEEE).

on the remaining battery-powered nodes.

In our proposed approach, since packets are relayed through a backbone with certain restrictions such as including all mains-powered nodes, suboptimal results are expected as far as the average path length from a node to the sink is concerned. As depicted in Figure 3.5, the average path lengths are twice as long on the average for MST versions of our algorithms compared to the basic shortest path algorithm, which ensures that each node is connected to the sink via the shortest possible path. On the other hand, for the SPT versions of our algorithms, the average path length is at most 40% longer compared to the shortest path algorithm. Note that for the shortest path algorithm after around 250 rounds, which corresponds to about 50 (i.e., 10%) unreachable nodes, the average path length exhibits a noticeable increase and eventually exceeds the SPT versions of our algorithms. This is probably due to the battery-powered node failures at critical locations, which lead to longer paths.

Finally in Figure 3.6, the effect of mains-powered node ratio on the effectiveness of our algorithms is presented. As the figure shows, our backbone-based

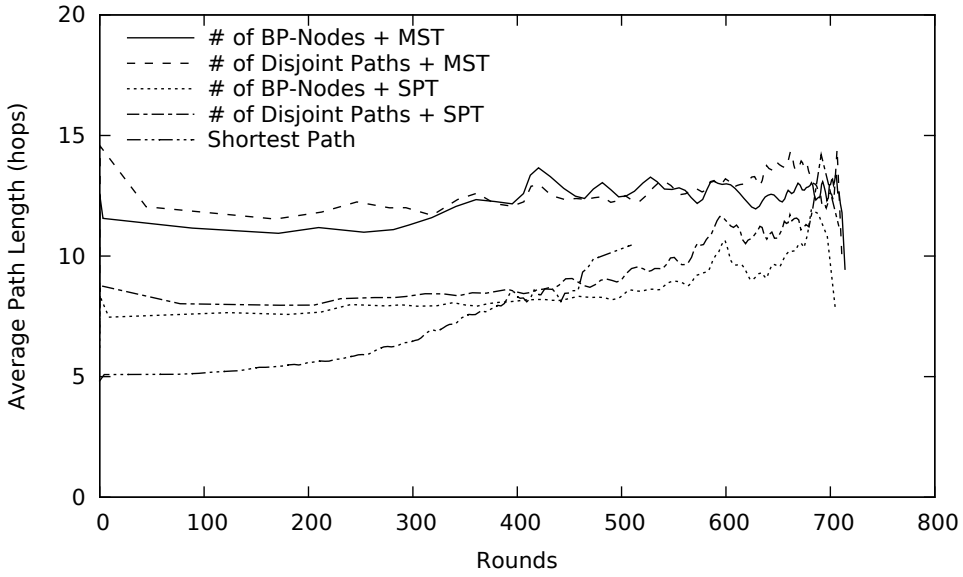

Figure 3.5: Number of rounds passed vs. average path length to the sink (mains-powered node ratio: 20%) (reprinted from Fig. 4 of [1] © 2010 IEEE).

routing approach outperforms the basic shortest path algorithm in terms of network lifetime and stabilizes at around 715 rounds, which is the theoretical upper bound for network lifetime considering battery-powered nodes initially have 1000 units of energy and each transmission consumes 1.4 units of it. For the 40% mains-powered node ratio, all algorithms based on the proposed approach have exactly 715 rounds of lifetime, meaning that the backbone is completely composed of mains-powered nodes, for the node density preferred in the simulations (500 nodes in an area of $500\,\mathrm{m}\times500\,\mathrm{m}$). As the mains-powered node ratio increases, the shortest path algorithm exhibits longer lifetime, which is the result of coincidental benefit obtained from mains-powered nodes.

3.4 Conclusions

In this chapter, we proposed a routing approach together with an algorithm framework, and a set of centralized routing algorithms based on this framework that can effectively increase the lifetime of WSNs with heterogeneous power sources. In order to achieve this, a backbone is formed to relay the data packets. The

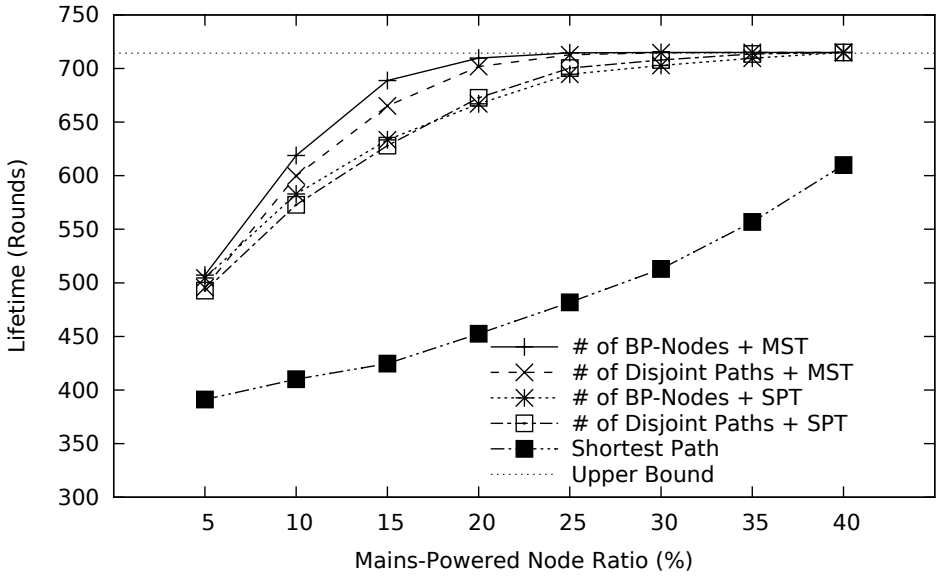

Figure 3.6: Mains-powered node ratio vs. network lifetime (reprinted from Fig. 5 of [1] © 2010 IEEE).

backbone consists of mains-powered nodes which are assumed to coexist with battery-powered nodes.

Although the MST versions of our algorithms achieve longer lifetimes in our simulations, SPT versions have a very close performance with much better average shortest path length values. Since SPT also has a less complex distributed implementation, it is further investigated in this thesis. Other algorithms, with different primary purposes, based on the proposed framework can also be studied as a future work. In the proposed algorithms, nodes are kept in idle mode, unless they are receiving or transmitting, and the simulation results are obtained accordingly. Putting non-backbone nodes into sleep mode can be considered as a way to extend the network lifetime and benefits of such a scheme can be analyzed.

Chapter 4

A Distributed Algorithm for Power-Source-Aware Routing in Wireless Sensor Networks

An algorithm framework, which aims at increasing the lifetime of heterogeneous wireless sensor networks (WSNs), is presented in Chapter 3. In the same chapter, the effectiveness of the centralized algorithms based on this framework is also shown by means of simulation results.

In this chapter we propose and present the detailed description of a distributed power-source-aware backbone-based routing algorithm, called PSABR. PSABR is a distributed version of one of the centralized algorithms given in the previous chapter, Algorithm "# of BP-Nodes + SPT". PSABR assumes battery- and mains-powered nodes coexist in the network and form a backbone using mostly the mains-powered nodes to relay data packets from the sensor nodes to the sink similar to the proposed framework. But different than the algorithms based on the framework presented in the previous section, PSABR achieves this in a fully distributed manner, without requiring any centralized control and without requiring global topology information to be available at a center.

The remainder of this chapter is organized as follows. In Section 4.1, we

present a detailed description of our distributed algorithm PSABR. We give the simulation results showing the performance of PSABR later in Section 4.2. Finally, in Section 4.3, we conclude the chapter with some discussions.

4.1 Our Distributed Routing Algorithm: PSABR

In this section, we describe and detail our distributed power-source aware routing algorithm, PSABR, which is based on the approach presented in Chapter 3. Please refer to Section 3.2 for the terms *neighbor*, *peer*, and *parent* used in the algorithm descriptions.

In our distributed algorithm, 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. 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.

The proposed algorithm 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 our algorithm are transferred by either broadcast or source routing. The route information in a source-routed packet is extracted from the partial visibility graph maintained by the nodes. We assume all nodes have periodic data to send to the sink and that data aggregation takes place. Parent nodes aggregate their own data with the data received from their child nodes and send the aggregated message to their parents. 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. As the battery-powered nodes on the backbone forward data messages, they aggregate their own data with the forwarded. Battery-powered nodes that are not on the backbone send the data packets to their parents which are one hop away.

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 the next subsection, we present the control messages used to gather information and form the parent-child relations (i.e., the backbone routing tree). In Section 4.1.2, we give a sample scenario to explain how the messages are used during backbone construction. In Section 4.1.3, we describe the node behavior at different events, including when receiving the control messages. Finally, in Section 4.1.4, we discuss the messaging overhead of the distributed algorithm.

4.1.1 Messages

A summary of the messages used by PSABR are given in Table 4.1. In the table, the transfer type (T) of the messages are given either as broadcast (B) or unicast (U). Power-source types of the senders are also provided as either battery-powered node (BP) or mains-powered node (MP). Note that some messages can be sent by both battery- and mains-powered nodes. The message sent as a response to the current message is given in the last column of the table. Next we describe the messages in more detail.

Table 4.1: Summary of the messages.

Abbr.	**Name**	**T**	**Sender**	**Replied by**
MDM	MP-Node Discovery Message	B	BP-MP	MIM
MIM	MP-Node Information Message	U	BP-MP	-
MUM	MP-Node Update Message	U	MP	-
BCM	Backbone Construction Message	U	MP	BCMACK
LFM	Link Failure Message	U	BP	-
NDM	Neighbor Discovery Message	B	BP	NIM
NIM	Neighbor Information Message	B-U	BP-MP	-

MDM (MP-Node Discovery Message) – An MDM is initiated by a battery- or mains-powered node 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 accumulated path r, which consists of battery-powered nodes, visited 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 generated from messages routed either by source routing or table-driven routing. The LFM's routing method matches the routing method of the message that caused it. Hence, an LFM$_{tdr}$ 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$_{tdr}(d, un, up)$ 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$_{tdr}$ 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 a response to an NDM or as the cost of the node changes. An NIM is transferred either by broadcast or 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.

4.1.2 Sample Backbone Construction

In this section, we explain how the messages described in Section 4.1.1 are used to construct a backbone using a sample scenario shown in Figures 4.1 and 4.2.

In Figures 4.1 and 4.2, 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 4.1 (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 Figure 4.1 (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 ($\varnothing$). 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 Figure 4.1 (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 Figure 4.1 (d). 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 Figure 4.1 (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.

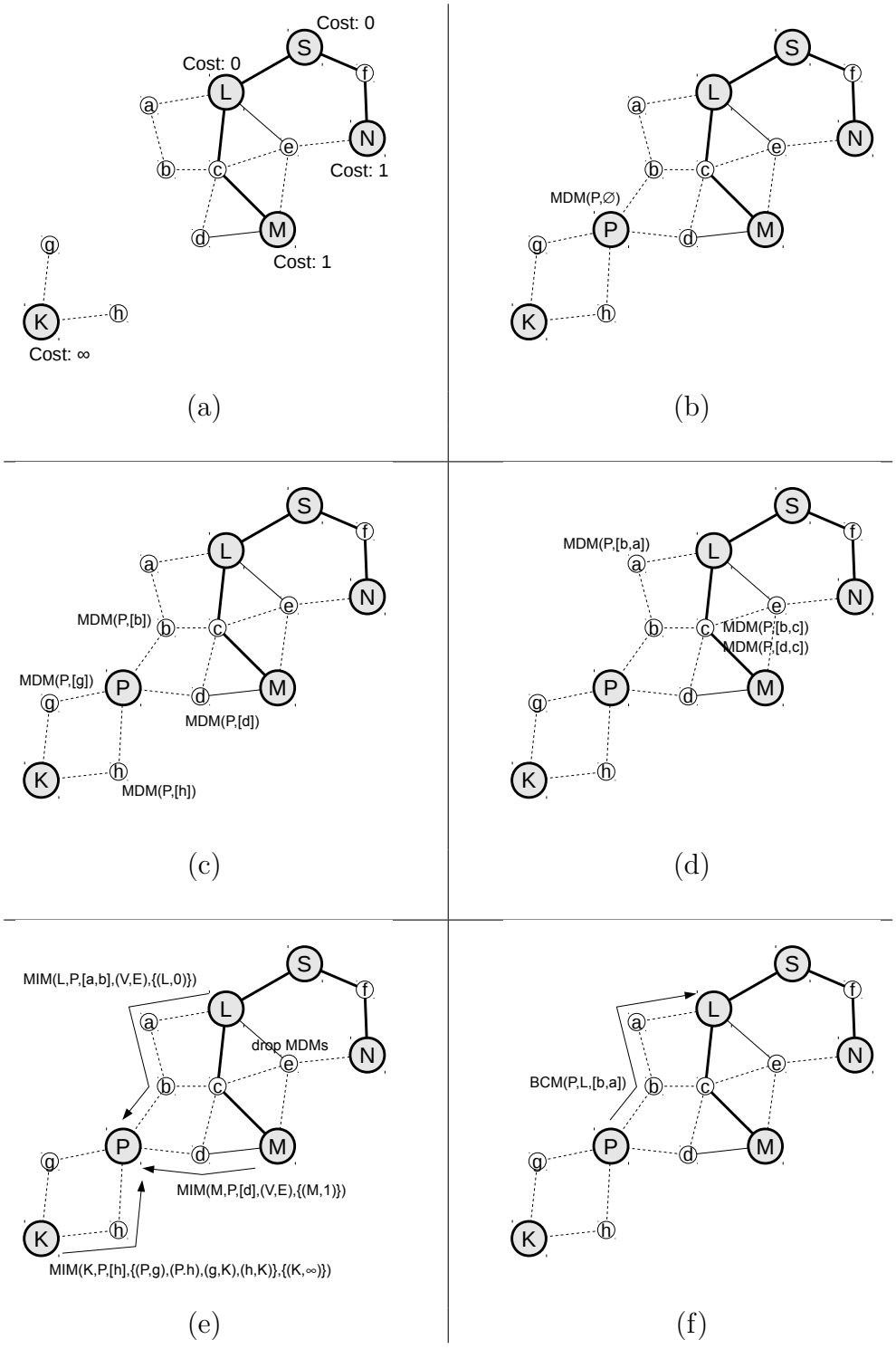

Figure 4.1: Backbone construction (1 of 2).

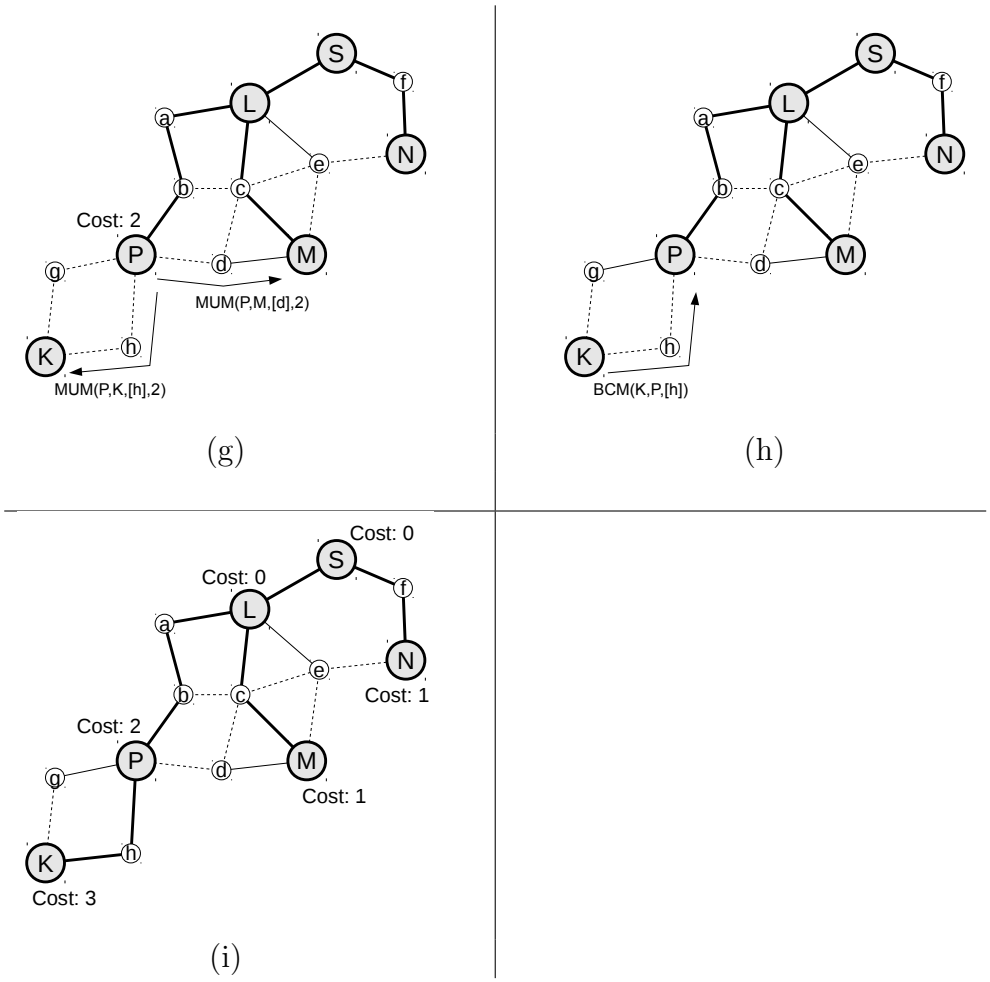

Figure 4.2: Backbone construction (2 of 2).

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 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 Figure 4.1 (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 Figure 4.1 (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 Figure 4.2 (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 (Figure 4.2 (h)). The final topology, after P and K exchange BCM and BCMACK, is depicted in Figure 4.2 (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.

4.1.3 Behavior

Node behaviors, described in the form of finite state machines (FSM) are depicted in Figures 4.3 and 4.4, and the algorithms are presented in Algorithms 4.1 to 4.11.

Table 4.2: Variables and expressions.

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

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 4.2.

The FSM of a mains-powered node is depicted in Figure 4.3. Initially, a mains-powered node is in the *Idle* state. With the *Start* event, it transits into the *WaitMIMTimeout* state. *Start* is fired when the node is powered up, as shown in Algorithm 4.1. At the transition from *Idle* to *WaitMIMTimeout*, a mains-powered node broadcasts an MDM and starts a timer, t_{mim}, for corresponding

Algorithm 4.1 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

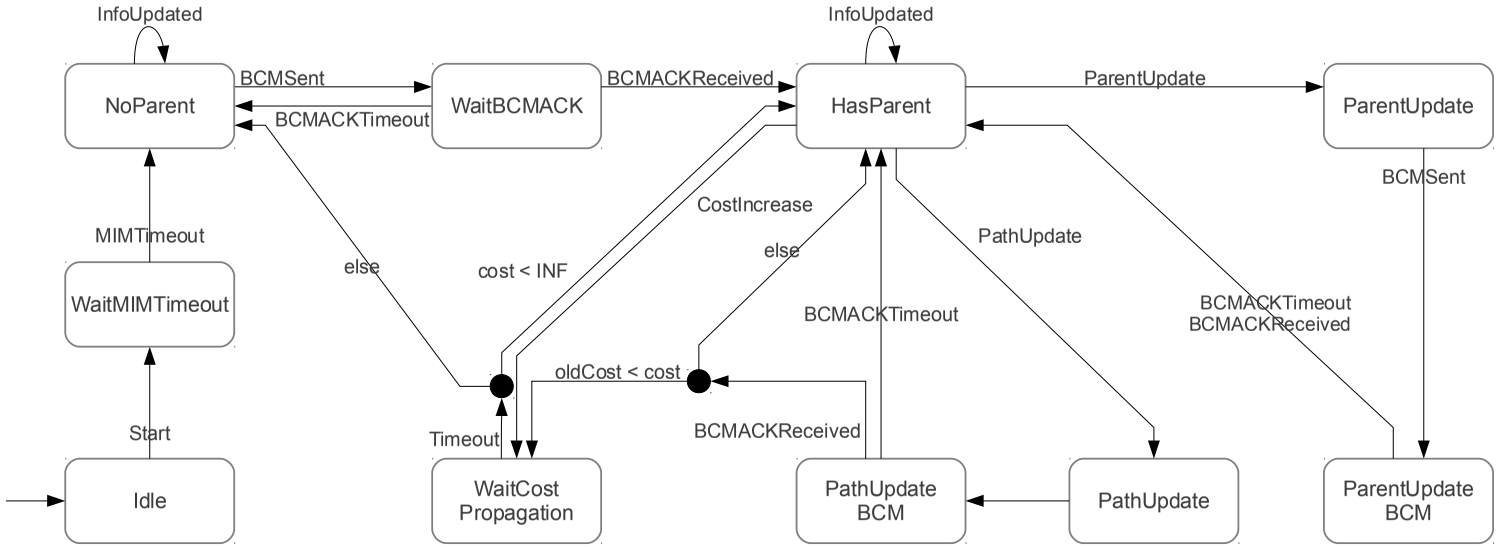

Figure 4.3: Finite state machine for mains-powered nodes.

Algorithm 4.2 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**

MIMs. Each MIM received restarts the timer, as shown in Algorithm 4.5, 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.

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 4.10. 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 4.4. 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. The third method to discover new peers or new paths to known peers is achieved by the help of newly joined battery-powered

nodes, which is presented later in this section.

Algorithm 4.3 On Entry of HasParent (MP-Node)

```
1: if cost < ∞ then
2:    if parent ∈ peers then
3:       if peers[parent].cost > cost then
4:          if peers[parent].cost < inf then
5:             cost ← peers[parent].cost
6:          else
7:             parent ←⊥, cost ← ∞
8:          end if
9:          fire CostIncrease
10:      else
11:         if ∃p ∈ peers s.t. peers[p].cost < peers[parent].cost then
12:            fire ParentUpdate
13:         else
14:            if pathToParent still exists then
15:               if cost ≠ peers[parent].cost then
16:                  cost ← 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 ←⊥, cost ← ∞
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 4.2.

Algorithm 4.4 On Message Receive (MP-Node) - MDM

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 $n_1, n_2, ..., n_k$ 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**

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 4.11) 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 4.3.

If a better parent candidate is found in the *HasParent* state, a *ParentUpdate* event is fired (Algorithm 4.3, line 12) 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 4.5 On Message Receive (MP-Node) - MIM, MUM

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**

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

If the path to the current parent needs updating in the *HasParent* state, a *PathUpdate* event is fired (Algorithm 4.3, 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.

Algorithm 4.6 On Message Receive (MP-Node) - LFM

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: $peers[p].cost \leftarrow peers[p].cost_{peer} |\mathrm{SP}(self, p, V, E)|$
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

The node cost increases in the following two cases: the parent cost increases (Algorithm 4.3, line 3) or the current parent becomes unreachable (Algorithm 4.3, 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

Algorithm 4.7 On Message Receive (MP-Node) - BCM, BCMACK

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$, cost)$

wait for the information to disseminate before attempting to find a better-cost parent; otherwise routing 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 then 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 4.4, 4.5, and 4.6 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 4.2) or the validity of the current parent (Algorithm 4.3), respectively.

The FSM of a battery-powered node is depicted in Figure 4.4. Initially, a battery-powered node is in the *Idle* state. With the *Start* event, it transits into the *WaitMIMTimeout* state. *Start* event is fired when the node is powered up, as shown in Algorithm 4.8. At the transition from *Idle* to *WaitMIMTimeout*, a battery-powered node broadcasts MDM and 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.

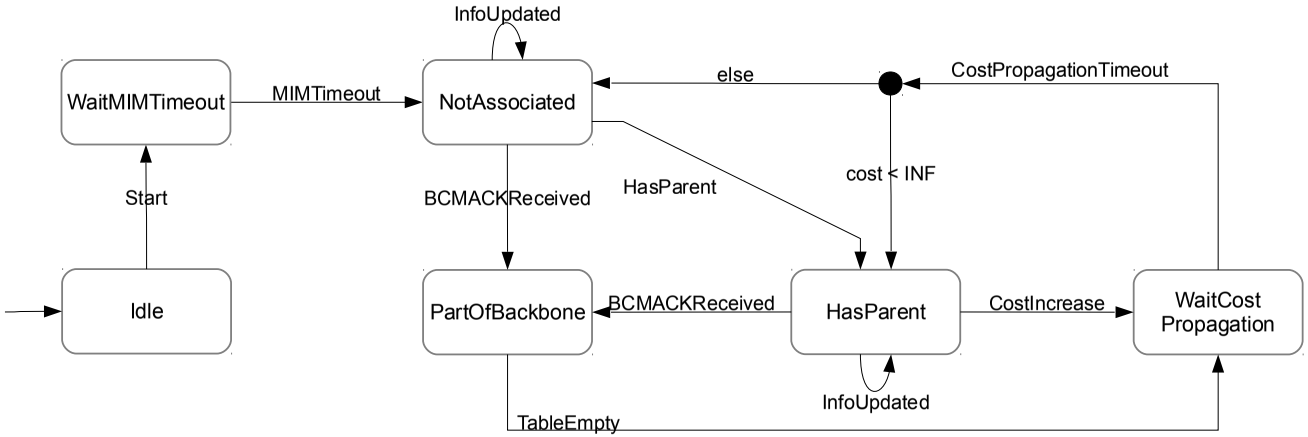

Figure 4.4: Finite state machine for battery-powered nodes.

Algorithm 4.8 On Power-up (BP-Node)

```
1: neighbors ← ∅
2: parent ←⊥, cost ← ∞
3: V ← {self}, E ← ∅, I ← ∅
4: fire Start
```

A battery-powered node discovers nearby mains-powered nodes by broadcasting an MDM and receiving corresponding MIMs (Algorithm 4.10, 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 4.10, 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.

Algorithm 4.9 On Entry of HasParent (BP-Node)

```
1: if parent ∈ neighbors then
2:    if neighbors[parent].cost > cost then
3:       if neighbors[parent].cost = ∞ then
4:          parent ←⊥, cost ← ∞
5:       else
6:          cost ← neighbors[parent].cost + 1
7:       end if
8:       broadcast NIM(self, ∗, BP, cost)
9:       fire CostIncrease
10:   else
11:      if ∃n ∈ neighbors s.t. neighbors[n].cost < (cost − 1) then
12:         parent ← n, cost ← neighbors[n].cost + 1
13:         broadcast NIM(self, ∗, BP, cost)
14:      end if
15:   end if
16: else
17:   parent ←⊥, cost ← ∞
18:   broadcast NIM(self, ∗, BP, cost)
19:   fire CostIncrease
20: end if
```

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 4.10 On Message Receive (BP-Node) - MDM, MIM, MUM, LFM

When MDM(s, r, ps) arrives:
1a: **if** $(|r| < (T-1))$ **and** $(self \notin r)$ **then**
2a: $r' \leftarrow r + self$
3a: **broadcast** MDM(s, r', ps)
4a: **end if**
When MIM(s, d, r, V', E', I') arrives:
1b: **if** $s \neq self$ **then**
2b: $i \leftarrow$ index of $self$ in list r
3b: **send** 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 MUM(s, d, r, c) arrives:
1c: $i \leftarrow$ index of $self$ in list r
2c: **send** MUM(s, d, r, c) to $r[i+1]$
When LFM$_{tdr}(d, un, up)$ arrives:
1d: **send** LFM$_{tdr}(d, un, up)$ to *next-hop*$[d]$
When LFM$_{sr}(d, r, un, up)$ arrives:
1e: $i \leftarrow$ index of $self$ in list r
2e: **send** LFM$_{sr}(d, r, un, up)$ to $r[i+1]$

On entry to the *HasParent* state, the node checks whether it can still access its current parent (Algorithm 4.9, line 1), whether the cost of the current parent has changed (Algorithm 4.9, line 2), and whether there is a parent candidate with a better cost (Algorithm 4.9, 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 4.11, line 3g) and the node transits into the *PartOfBackbone* state. As shown in Algorithm 4.11, 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, therefore, when a battery-powered

node is not part of the backbone its routing table becomes empty.

Algorithm 4.11 On Message Receive (BP-Node) - BCM, BCMACK, NDM, NIM

When BCM(s, d, r) arrives:
1f: $i \leftarrow$ index of $self$ in list r
2f: **if** $i = 0$ **then**
3f: $\quad next\text{-}hop[s] \leftarrow s$
4f: **else**
5f: $\quad next\text{-}hop[s] \leftarrow r[i-1]$
6f: **end if**
7f: **if** $i = (|r| - 1)$ **then**
8f: $\quad next\text{-}hop[d] \leftarrow d$
9f: **else**
10f: $\quad next\text{-}hop[d] \leftarrow r[i+1]$
11f: **end if**
12f: **send** BCMACK(s, d, r) to $next\text{-}hop[d]$
When BCMACK(s, d, r) arrives:
1g: *same as the lines [1f,11f]*
2g: **send** BCMACK(s, d, r) to $next\text{-}hop[d]$
3g: **fire** BCMACKReceived
When NDM(s) arrives:
1h: **send** NIM$(self, s,$ BP$, cost)$
When NIM(s, d, ps, c) arrives:
1i: $neighbors \leftarrow neighbors \cup \{s\}$
2i: $neighbors[s].cost \leftarrow c$
3i: **fire** InfoUpdated

If, when in the *PartOfBackbone* state, a 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.

Note that PSABR can work with multiple sinks. Each sink advertises its cost as zero and each mains-powered node chooses a parent minimizing its own cost, which is the sum of its parent's cost and the cost to reach its parent. Therefore, the backbone is constructed minimizing cost of mains-powered nodes by choosing the appropriate parents and in turn the appropriate sink.

4.1.4 Analysis

This section analyzes PSABR's 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 exactly t-hop neighbors, k_t, of a node is given in Equation (4.1). 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.

$$k_t = n\left(\frac{\pi[(tr)^2 - ((t-1)r)^2]}{\pi R^2}\right) = n(2t-1)\left(\frac{r}{R}\right)^2 \tag{4.1}$$

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 Equation (4.2). 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 of transmissions 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. Equation (4.2) 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.

$$C_{discover} \leq \sum_{t=1}^{T-1}[k_1(1-m)]^t + \sum_{t=1}^{T} k_t m t \tag{4.2}$$

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 Section 4.2. In the same section, we also compare Equation (4.2) 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 Equation (4.3), and is basically the number of transmission required to send MUM 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, cost change of a mains-powered node causes cost change 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.

$$C_{advertise} = \sum_{t=1}^{T} k_t mt \tag{4.3}$$

4.2 Performance Evaluation

This section presents our simulation results illustrating the performance of PSABR. First, in Section 4.2.1, we give the simulation implementation details. Then in Section 4.2.2, we present the method used to visualize the algorithm behavior. We explain the algorithm parameters in Section 4.2.3. Finally in Section 4.2.4, we give the detailed simulation results and their interpretations.

4.2.1 Simulation Implementation Details

Proposed routing algorithm is implemented as a network layer protocol in ns-2 [6] (version 2.34) simulation environment. ns-2 models and simulates a mobile node as depicted in Figure 4.5. LL has the link layer implementation. It uses ARP to handle IP address to MAC address conversions. Outgoing packets are handed down to link layer by the routing agent, and incoming packets are handed by the MAC layer, directly to the link layer. IFq is a priority queue implementation and gives precedence to the routing control packets over other packets. MAC has the media access control protocol implementation. There are currently different alternatives readily available in ns-2 for wireless MAC. NetIF corresponds to the network interfaces of the mobile node. Similar to MAC, there are different interface implementations. It basically stamps outgoing packets with metadata containing transmission power, wavelength, etc., hence the propagation model can use. In turn, packet collisions and corruptions can be simulated. As shown

in Figure 4.5, above these components, two demultiplexers are located. They forward packets to other components according to their address and port information. Hence, the packets can be consumed within the node or forwarded to a neighbor. The RTagent component provides routing functionality so that the packets can be sent to the appropriate nodes.

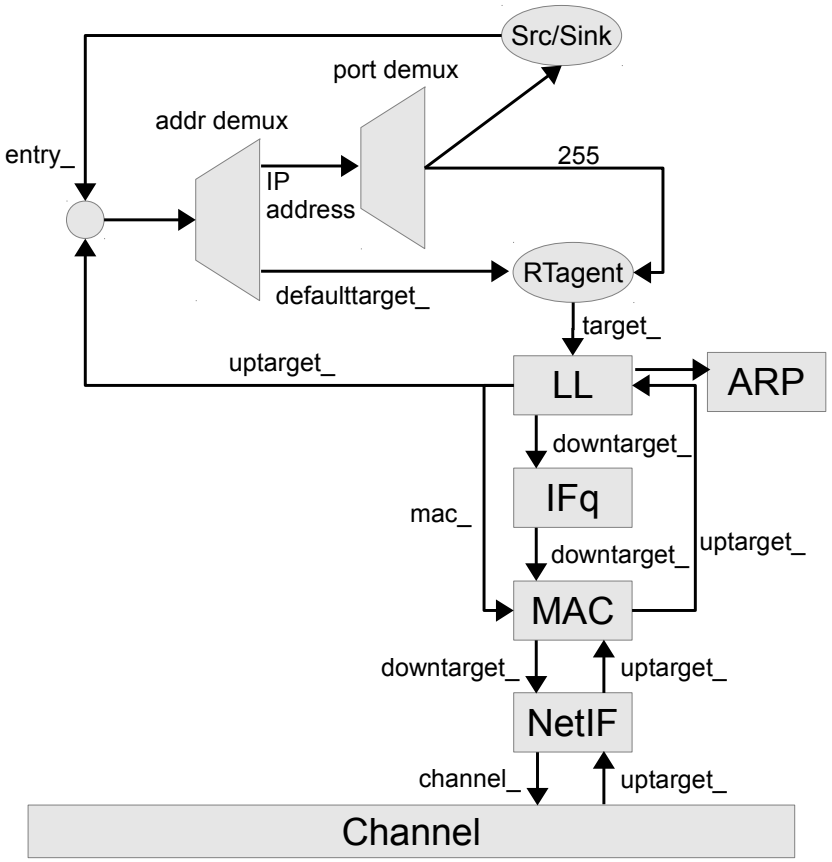

Figure 4.5: Mobile node architecture in ns-2.

In order to simulate our algorithm, we provided our own RTagent implementation. As seen in Section 4.1, battery- and mains-powered nodes have different behavior. Therefore, two different routing agents (one for the battery-powered nodes and another for the mains-powered nodes), which can cooperate, are implemented. Besides exchanging routing packets, they maintain a routing table according to the information obtained from these messages. We assume that the nodes do not use separate network addresses and use their device addresses as their network addresses as well. Therefore, we do not need ARP. To adapt ns-2 to

this condition, we gave the same identifier as both network and device addresses. Next we modified ARP to return the same address that it is provided, without any actual address resolution.

IEEE 802.15.4 [10], which is already implemented in ns-2, is used as the underlying MAC and physical layer protocol. To compare PSABR's performance, we use a shortest-path routing implementation. In the shortest-path routing, packets are forwarded to the sink through the path with the minimum-hop distance among all possible paths.

In the simulations, we assume that each sensor node sends data to the sink periodically. We also assume that data can be aggregated. To run the simulation in accordance with these assumptions we partly used infrastructure already available in ns-2, but we also implemented certain functionality. In order to generate periodic data packets we used the traffic generators, to be specific, constant bit rate (CBR) traffic generator of ns-2. The CBR traffic generator allows to change the frequency and size of the data packets. Once a packet is received at the network layer of a node, where the proposed algorithm is implemented, either from the upper layer (i.e., from CBR traffic generator) or from the lower layers (i.e., from another node), the packet is either dropped, forwarded (i.e., routed), or "stored" to be aggregated at a later time. Battery- and mains-powered nodes have different algorithms to decide the action to be taken, which are specified in Algorithms 4.12 and 4.13.

Algorithm 4.12 Data packet processing on mains-powered nodes

```
1: if data is originating from this node then
2:     if node has parent then
3:         Forward to the next hop according to the routing table
4:     else
5:         Drop
6:     end if
7: else
8:     if node has parent then
9:         Store
10:    else
11:        Drop
12:    end if
13: end if
```

Algorithm 4.13 Data packet processing on battery-powered nodes

```
1: if data is originating from this node then
2:    if node is part of the backbone then
3:       Store
4:    else
5:       if node has parent then
6:          Forward to the parent
7:       else
8:          Drop
9:       end if
10:   end if
11: else
12:   if data is originating from a battery-powered node then
13:      if node is part of the backbone or it has a parent then
14:         Store
15:      else
16:         Drop
17:      end if
18:   else
19:      if there is a routing table entry for the destination then
20:         Forward to the next hop according to the routing table
21:      else
22:         Drop
23:      end if
24:   end if
25: end if
```

In general, if a node cannot reach sink (i.e., does not have a parent, or is not part of the backbone), it drops the data packets. On the other hand, if a node decides to aggregate a packet at a later time, it "stores" the packet. Storing a packet in the simulation implementation is virtual, since there is no actual data, hence no actual aggregation. Dropping and storing a data packet have the same external behavior (i.e., no packet transmission). Therefore, in the simulation implementation, if a node decides to store a packet, rather than to forward or to drop it, the node drops it with a special label. In this way, we simulate the aggregation, since the resulting over-the-air traffic is equivalent to our assumptions.

We consider a node is *reachable* if it is alive and 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 lifetime is the time passed until half of the nodes become unreachable, and we run the simulations accordingly, unless otherwise noted. To simulate this behavior, we monitor the changes in the parent-child relations and the paths between the parent and child if they are both mains-powered. At every change, we compute number of nodes reachable from the sink and if it is below half the number of nodes for a period of time, we stop the simulation.

4.2.2 Visualization of Simulations

As we implemented the proposed algorithm in ns-2 environment, we used small and hand crafted networks to check whether the algorithm works as expected. We tried to choose networks to test some odd cases. But it is not easy to foresee all the possible cases and guess the algorithm behavior as the network gets larger. In the simulations, we started to use as high as 300 nodes whose locations are determined randomly. In such a chaotic environment, we required a visual mechanism to track the algorithm behavior.

As mentioned in Section 4.2.1, we monitor the changes in the parent-child relations and the paths between the parents and their children. At every change, having this information, we draw the graph representing logical links. The current logical links are represented using the DOT graph description language [91] and this textual representation is converted into an image using the neato tool found in graphviz [92], which is an open-source graph visualization toolset. Such an approach helped us identify several major and minor problems about the algorithm, and achieve a robust algorithm after several iterations.

Some sample images are depicted in Figure 4.6. In the images, nodes that consider themselves as connected are shown with solid lines and others with dashed lines; mains-powered nodes are shown as red and battery-powered nodes as black; backbone paths are shown with red edges and parent-child relation for

battery-powered nodes with black edges.

4.2.3 Simulation Parameters

Table 4.3: Simulation parameters.

Abbr.	**Parameter**	**Value**
n	Node count	*variable*
m	Mains-powered node ratio	*variable*
ρ	Node density	*variable*
σ	Sink count	*variable*
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

We use several parameters in the simulations to observe the impact of different conditions on the algorithm's performance (Table 4.3). 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 [89]. Node arrival times are also determined randomly, keeping all the aforementioned parameters intact.

The energy model in the simulations is based on the values in [93]. According to this model, 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. As stated before, 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 destined to the

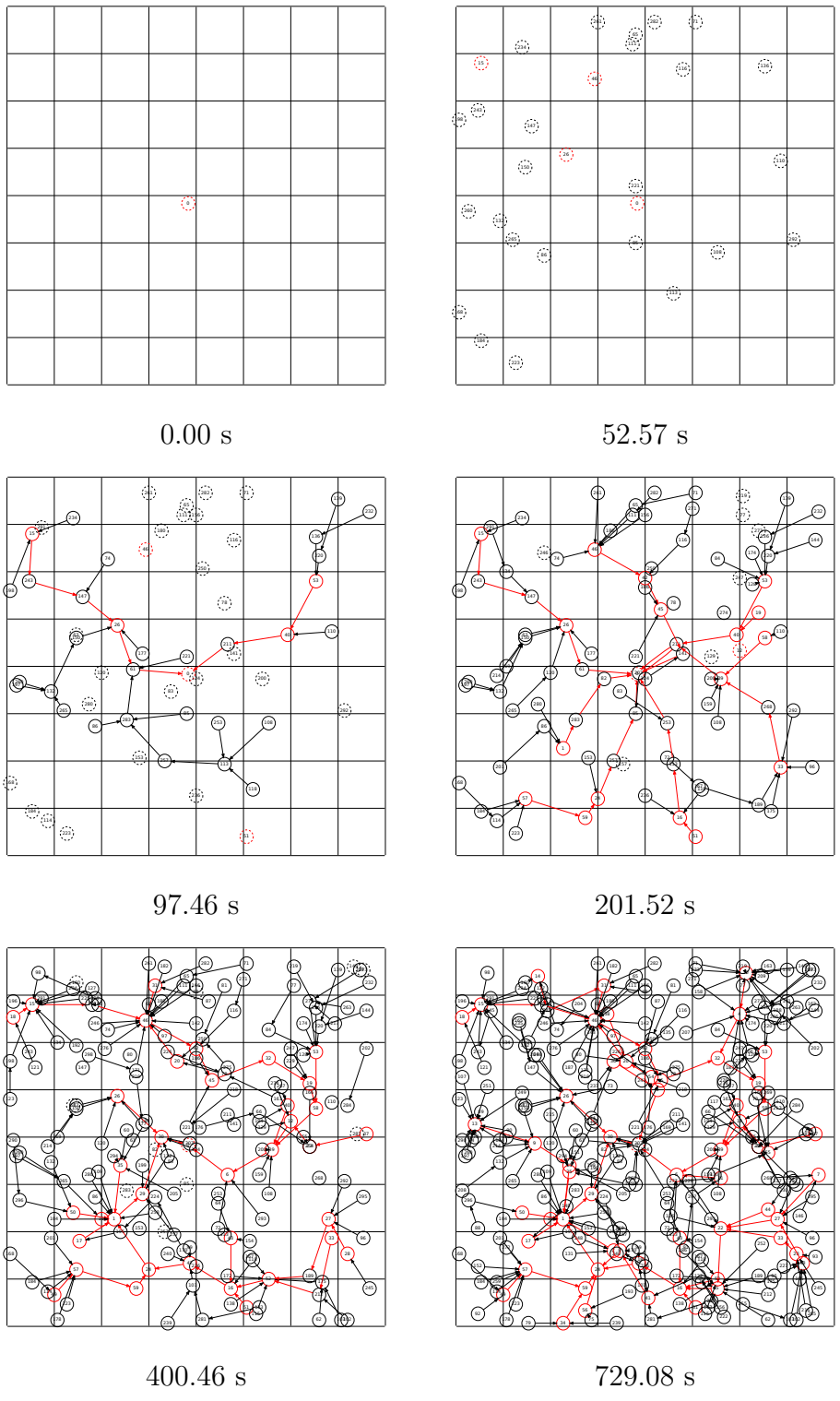

Figure 4.6: Visual output of a sample simulation run at different points in time.

sink with a 32-byte payload every 60 seconds.

4.2.4 Simulation Results

In the first part of this section, we present the simulation results mostly related to PSABR's behavior, evaluating its efficiency. Later, we give results that help evaluate the performance of the algorithm, i.e., its effectiveness.

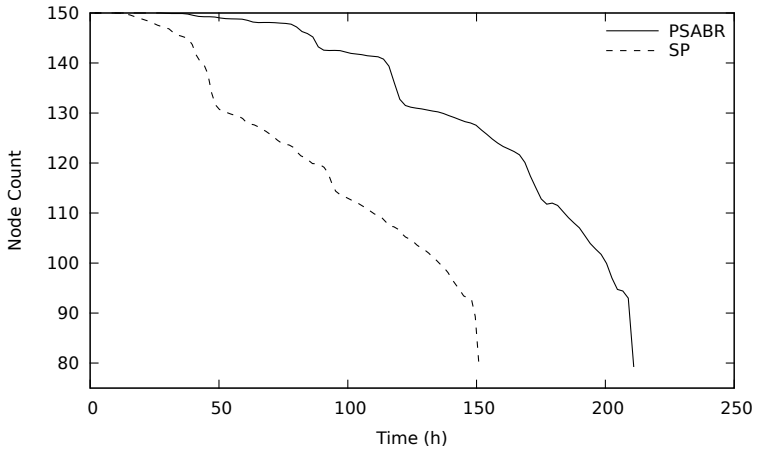

Figure 4.7: Number of reachable nodes over time [$n = 150$, $m = 20\%$].

Figure 4.7 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 drops, whereas in the shortest-path routing, it decreases almost linearly over time.

Figure 4.8 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 4.8 (a) shows how the average in-degree of battery-powered nodes changes

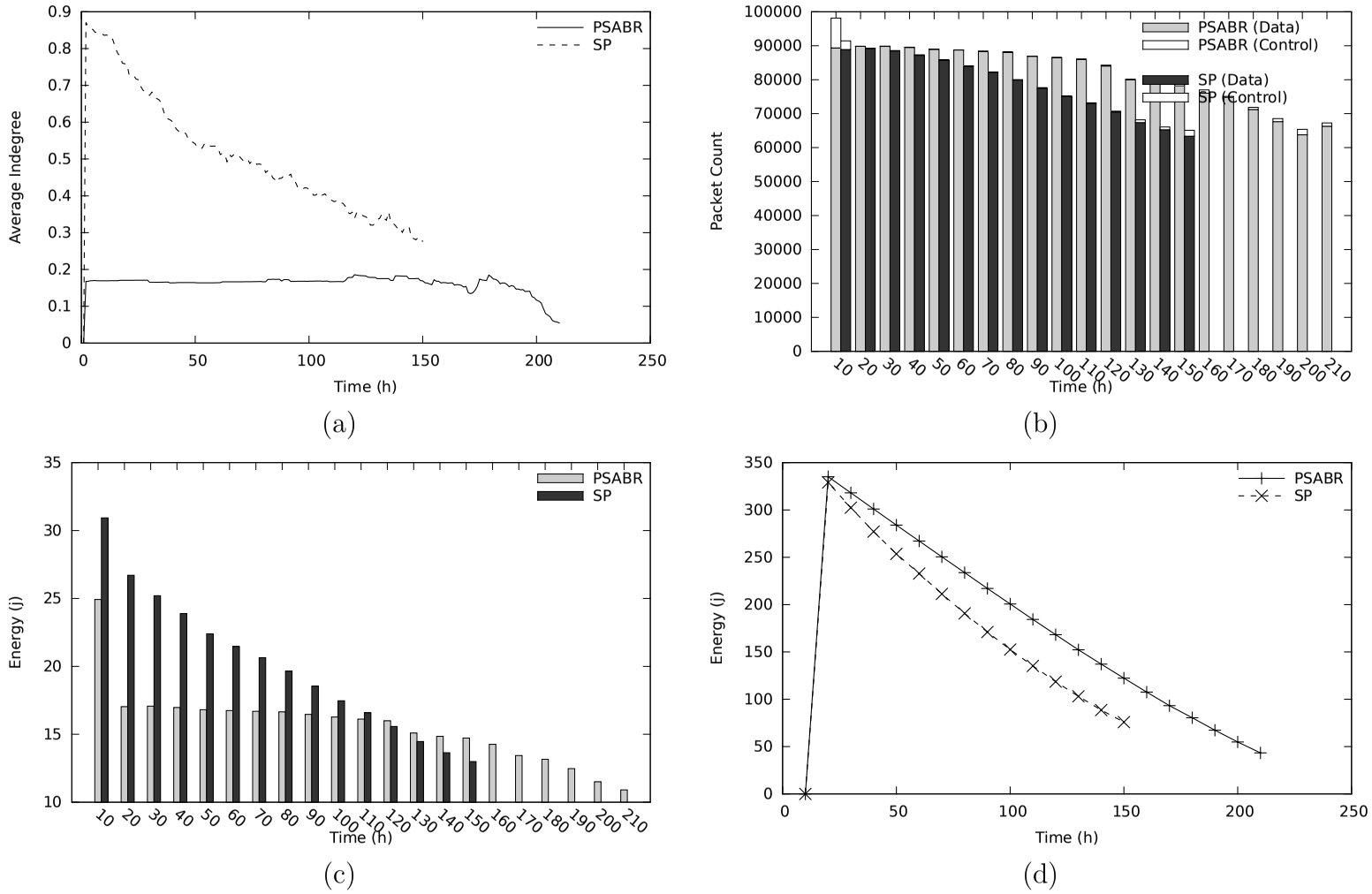

Figure 4.8: (a) Average indegree, (b) number of packets transmitted over time, (c) energy consumption of battery-powered nodes, and (d) total residual energy of battery-powered nodes [$n = 150$, $m = 20\%$].

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 result is primarily due to a decrease in the battery-powered node ratio because of battery depletion.

The number of packets transmitted in the network is shown in Figure 4.8 (b), 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.

In Figures 4.8 (c) and (d), we show the experiment results related to energy usage. Figure 4.8 (c) 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 4.8 (d) 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.

Figure 4.9 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 (straight 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

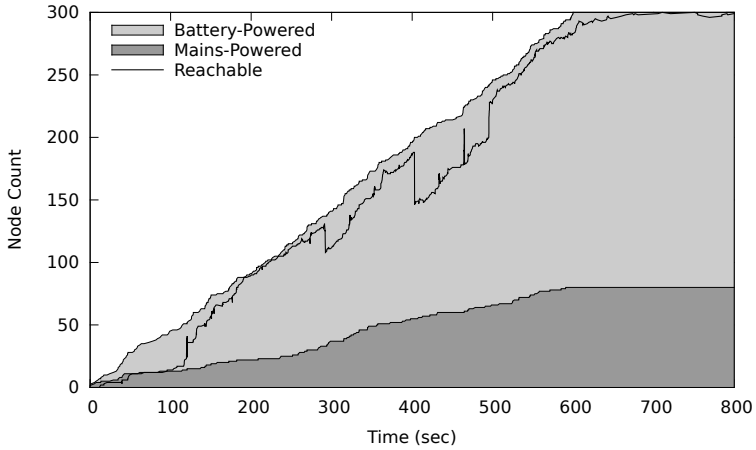

Figure 4.9: Node counts as new nodes arrive and the network is constructed [$n = 300$, $m = 25\%$].

algorithm's reactions in more detail. 300 nodes arrive in about 600 seconds, 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.

We now present our results regarding our algorithm's performance (effectiveness). Figure 4.10 depicts network lifetime under different conditions, with Figure 4.10 (a) 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 Figure 4.10 (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 4.10 (c) shows the effect of node density on network lifetime. In the other experiments, node density is around 1 node per 44 unit^2 (note that the communication range is around 20 units). Here, the lifetime is given for different density values relative to the usual case. 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

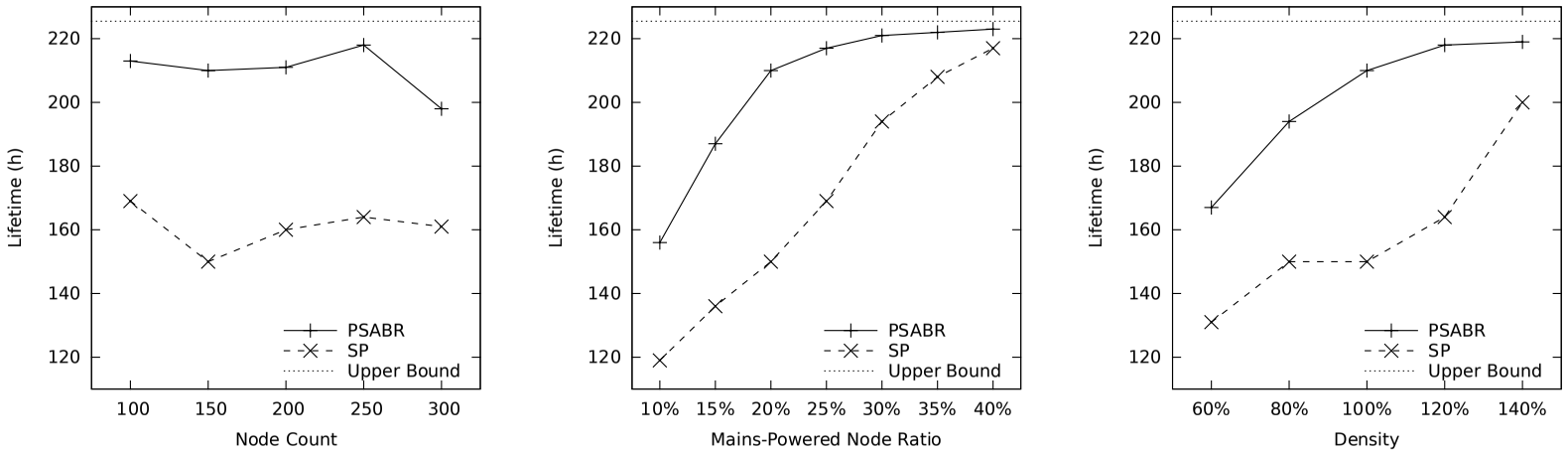

Figure 4.10: Network lifetime (assuming lifetime is the time passed until half of the nodes become unreachable from the sink) depending on (a) node count $[m = 20\%, \rho = 100\%]$, (b) mains-powered node ratio $[n = 150, \rho = 100\%]$, and (c) density $[n = 150, m = 20\%]$.

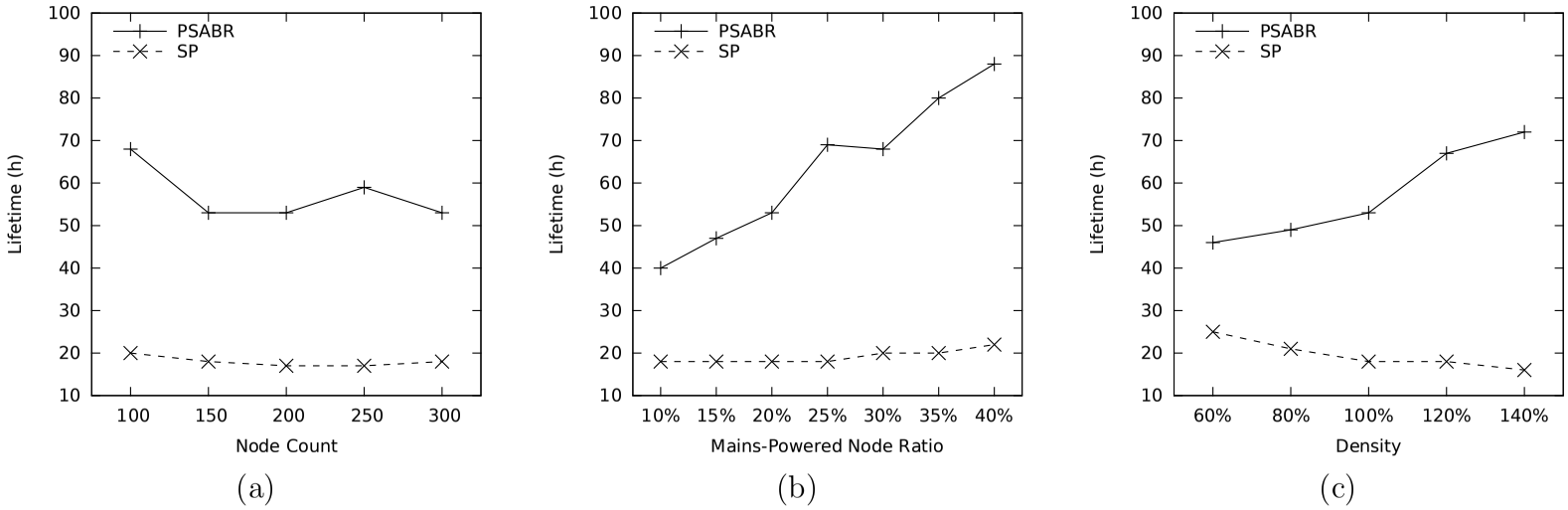

Figure 4.11: Network lifetime (assuming lifetime is the time passed until the first node death) depending on (a) node count $[m = 20\%, \rho = 100\%]$, (b) mains-powered node ratio $[n = 150, \rho = 100\%]$, and (c) density $[n = 150, m = 20\%]$.

the sink.

Figure 4.10 also shows the experimental upper bound. Given the simulation parameters (i.e., 32-byte packets transmitted every 60 seconds, 0.0807 W transmission energy, and 3 J initial energy), the bound is the time required until the energy of a battery-powered node drains completely, assuming that it does not exchange control packets or forward data but only transmits its own data packets. Note that the bound is not tight for cases where battery-powered node are required to forward data packets, such as with low node density or a low mains-powered node ratio.

For the simulation results given in Figure 4.10, we assume the lifetime is the time passed until the half of the nodes become unreachable as mentioned earlier. In Figure 4.11, 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. As shown in the figures, in first node death case, PSABR performs much better than the shortest-path routing, proportionally.

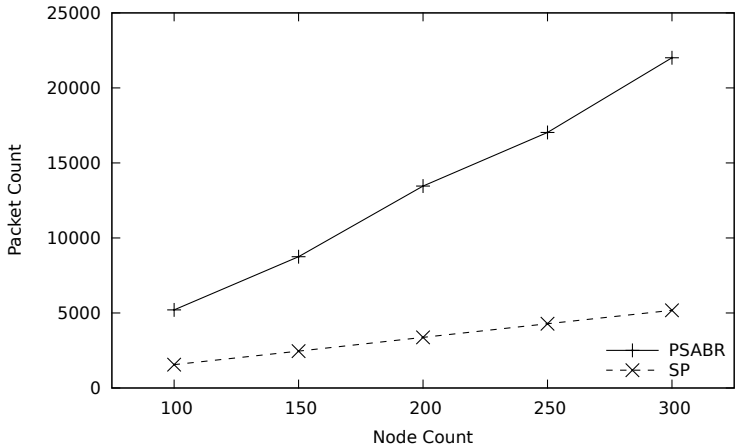

Figure 4.12: Number of control packets required to construct the network [$m = 20\%$].

The number of control packets required to construct the initial routing tree with respect to network size is presented in Figure 4.12. As shown in the figure, PSABR requires three to five times more control packets for construction. This 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, although it requires more control packets. 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.

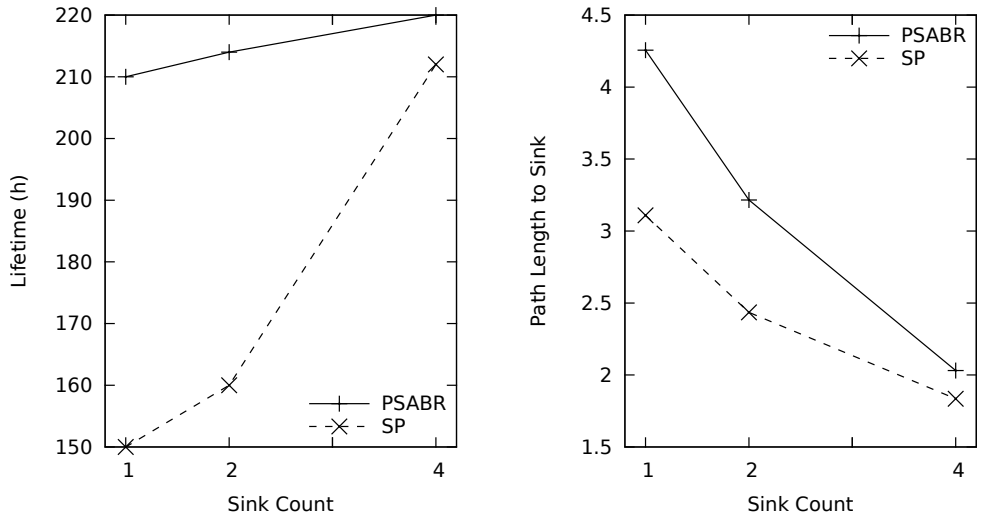

Figure 4.13: Lifetime and average path length to sink for different sink counts [$n = 150$, $m = 20\%$].

The proposed algorithm is designed to run with an arbitrary number of sinks, and Figure 4.13 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 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.

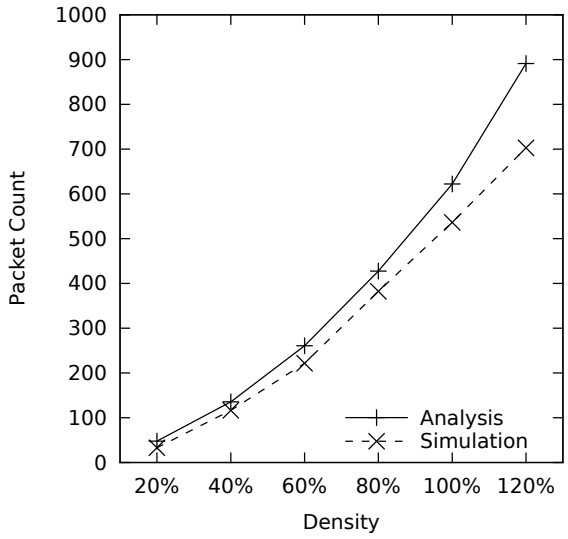

Figure 4.14: Number of messages (MDM and MIM) required to discover a peer [$n = 300$, $m = 20\%$].

In Figure 4.14, we compare the number of messages required for a mains-powered node to discover all of its peers found by analysis as given in Section 4.1.4, 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 Equation (4.2) 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 for the case all nodes are ready 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 total number of messages required for mains-powered node discovery would be much less than $nC_{discover}$, given in Section 4.1.4.

4.3 Conclusions

In this chapter, we proposed a distributed routing algorithm (PSABR) based on our approach presented in Chapter 3, which is able to significantly increase the lifetime of WSNs where different power-source types for nodes exist. Our PSABR algorithm 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. Although PSABR is especially designed for WSNs with battery- and mains-powered nodes, we explained in Chapter 2 that it can be used in WSNs where nodes have heterogeneous power sources.

In addition to message complexity analysis of PSABR, we also presented simulation results to better explain the algorithm behavior and to evaluate its performance. As our results show, distinguishing between sensor nodes according to their power sources 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 conventional routing 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.

Chapter 5

Power-Source-Aware Routing in ZigBee Networks

In the previous chapters, we provided power-source aware routing algorithms mostly independent of the underlying wireless technology. In this chapter, we propose a power-source-aware routing algorithm designed for a specific wireless technology, that is, ZigBee [8].

ZigBee is a short-range wireless networking technology that targets low-data rate as well as low-duty cycle applications. Such applications include a wide range of control and monitoring applications such as building automation, industrial control, and sensor networks. A typical deployment site is likely to have battery- and mains-powered devices coexisting.

Both tree and mesh topologies are possible in a ZigBee network. The ZigBee standard defines different address assignments and routing mechanisms for these topologies. Two different routing schemes are specified in the ZigBee standard: hierarchical tree routing and a modified version of ad hoc on-demand distance vector routing (AODV). In hierarchical tree routing, packets are routed according to the parent-child relationships established during ZigBee topology formation and distributed address assignment.

We propose a power-source-aware routing algorithm, PSAR, for tree topology ZigBee networks. PSAR is based on hierarchical tree routing and simply aims at reducing the power consumption of battery-powered devices and consequently increasing network lifetime. The basic approach to achieve this is to route network traffic through mains-powered devices instead of battery-powered devices as much as possible. When routing in tree topology networks, because there is a single path between any two devices, the only way to reduce the burden on battery-powered devices is to modify the network topology, by disconnecting and reconnecting some devices, to reduce traffic flow through the battery-powered devices.

PSAR requires only minor modifications to the current ZigBee protocol specification and minimal additional messaging, which keeps the overhead of the algorithm at a minimum. Our simulation results shows that the average traffic on battery-powered devices can be reduced by up to 50%, without a significant increase in the average path length between devices (hence neither in the total traffic load of the network) due to the topology changes.

The remainder of this chapter is organized as follows: In Section 5.1, we give some information on ZigBee standard including a brief overview and the distributed address assignment scheme, which is important for hierarchical routing. In Section 5.2, we present a detailed description of our proposed routing scheme, PSAR. Next, in Section 5.3, we give the simulation results. Finally, in Section 5.4, we conclude the chapter.

5.1 The ZigBee Standard

5.1.1 A Brief Summary

The ZigBee standard [8] defines a low-data rate wireless networking solution for interconnection of devices in a wireless personal area network (WPAN). The low-data rate requirement enables reduced complexity and very low power consumption, which are also the primary goals of ZigBee. The ZigBee standard is built on

the IEEE 802.15.4 standard [10], which shares similar goals. ZigBee defines the application layer (APL) and the network layer (NWK), whereas IEEE 802.15.4 defines the medium access control layer (MAC) and the physical layer (PHY), as depicted in the protocol stack of Figure 5.1. In this chapter, we use ZigBee to refer the ZigBee and IEEE 802.15.4 standards as a whole, unless otherwise specified.

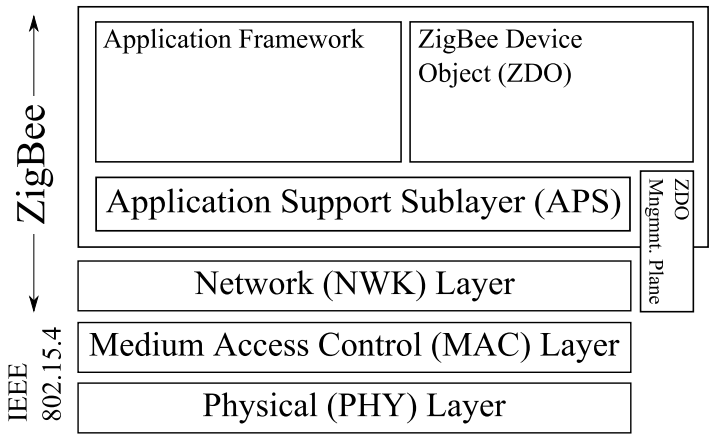

Figure 5.1: ZigBee protocol stack (reprinted from Fig. 1 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

The PHY layer defines 16 channels in the 2450 MHz band, 30 channels in the 915 MHz band, and three channels in the 868 MHz band [10]. Depending on the band, the devices can communicate with data rates of 250 kbps, 100 kbps, 40 kbps, and 20 kbps. The MAC layer controls access to the radio channel using the carrier sense multiple access with collision avoidance (CSMA/CA) mechanism. An optional superframe structure can be used to coordinate the channel access. A superframe, which is bounded by network beacons, can possibly include contention and contention-free access periods (CAP and CFA) as well as an inactive period. CFA periods can be assigned to time- or bandwidth-critical applications. On the other hand, inactive periods can be exploited to reduce power consumption by switching off the radio transmitters.

The NWK layer enables data transfer between devices that are not in the

communication range of each other through the use of intermediate devices, hence making multi-hop communication possible. Responsibilities of the NWK layer include starting a network, coordinating joining and leaving a network, routing, discovering one-hop neighbors, and storing neighbor information. Three types of devices are possible in a ZigBee network: Coordinator, router and end devices. Routers are capable of forwarding data on behalf of others and a coordinator is a router that starts the network and chooses key network parameters. Any device can connect to a router in the network, whereas, devices cannot connect to an end device, as the name implies. A ZigBee device is called a full functional device (FFD) if it can have a router role in the network and be a reduced functional device (RFD) otherwise. An RFD is usually limited in terms of its energy source (e.g., battery-powered), processing power, and memory capacity. Each ZigBee device has a universal 64-bit address and a 16-bit short address assigned when it connects to a network. As stated before, both tree and mesh topologies are possible in a ZigBee network, having different address assignments and routing mechanisms.

The APL layer of ZigBee consists of the application support sub-layer (APS) and the application framework. Responsibilities of the APS include maintaining tables used to bind devices according to the services provided and needed, forwarding between bound devices, fragmentation, reassembly, and reliable data transport. The application framework contains the ZigBee device object (ZDO) and manufacturer-defined application objects. ZDO defines the role of the device in the network, such as coordinator or end device, discovers application services, and manages service bindings.

5.1.2 ZigBee Address Assignment

There are different mechanisms for address assignment depending on the topology (i.e., tree or mesh) of the ZigBee networks. In tree topology ZigBee networks, there are two alternatives for the network address assignment. In one of the alternatives, address assignment is left to the next higher layer. In the other alternative, the specification defines a distributed address assignment mechanism.

According to the distributed address assignment mechanism, every potential parent is assigned a finite block of network addresses. Each parent later assigns one (if the child is an end device) or more (if the child is router-capable, and therefore a potential parent) of these addresses to the devices connected to it. The coordinator of a network determines the maximum number of children that a parent can have, which is denoted by Cm. Of these children only Rm of them can be router-capable. Every device has a depth, d, which is the minimum number of hops to the ZigBee coordinator (i.e., the root of the tree). Maximum depth, Lm, of a tree network is also determined by the coordinator of that network. Given these values, the function $C_{skip}(d)$, which is actually the size of the address sub-block assigned to a router-capable device at depth $d+1$, is computed as in Equation (5.1) [8].

$$C_{skip}(d) = \begin{cases} 1 + Cm \cdot (Lm - d - 1), & \text{if } Rm = 1 \\ \dfrac{1 + Cm - Rm - Cm \cdot Rm^{Lm-d-1}}{1 - Rm}, & \text{otherwise} \end{cases} \tag{5.1}$$

A parent device assigns an address one greater than its own to its first router-capable child and the address of each such child is separated by $C_{skip}(d)$. The address of the n^{th} end device, A_n, is computed as $A_n = A_{parent} + C_{skip}(d) \cdot Rm + n$, where $1 \leq n \leq (Cm - Rm)$ and A_{parent} is the address of the parent. Figure 5.2 depicts how the address space is used and redistributed at depth d.

Such a systematic address assignment mechanism enables a simple routing strategy. Any routing-capable device receiving a packet destined to an address A knows whether any of its children has address A or if A falls into the address sub-block of any of its children, in which case the packet is forwarded to the corresponding child. If no such child exists, then the packet is forwarded to the parent device. This routing strategy is called hierarchical routing and is applied in ZigBee tree topology. Although distributed address assignment eases the routing, one of its drawbacks is that whenever a device changes its parent, its and all of its descendants' network addresses need to change.

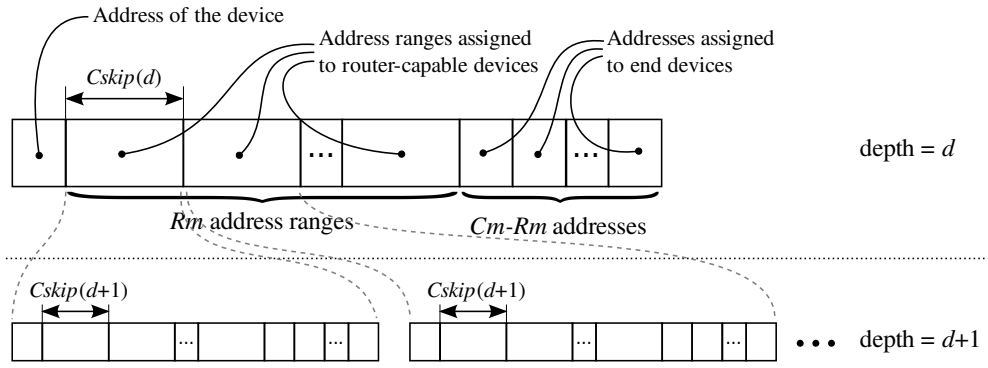

Figure 5.2: ZigBee distributed address assignment (reprinted from Fig. 2 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

5.2 Our Power-Source-Aware Routing Algorithm: PSAR

The basic strategy for our power-source-aware routing (PSAR) algorithm is to route the traffic through mains-powered ZigBee devices rather than battery-powered devices as much as possible. In a tree topology network there is a single simple path, hence, only one meaningful route between any two nodes. This means, once a topology is determined, no alternative route can be found to reduce the traffic routed by a battery-powered device. One possible solution is to modify the tree-based network topology dynamically depending on traffic demand so that the burden on the battery-powered devices is reduced.

Consider the ZigBee network given in Figure 5.3, where C is the coordinator of the network, the nodes from R_1 to R_9 are the routers and Es are the end devices. In the figure, mains-powered routers are shown with solid lines, whereas battery-powered routers are shown with dashed lines. Assume that R_6 and its children have communication with R_3, R_4, and R_5. For the topology given in Figure 5.3 (a), R_6-R_3, R_6-R_4, and R_6-R_5 communications must follow paths R_2-C, R_2-C, and R_2-C-R_1, respectively, meaning that battery-powered devices R_1 and R_2 are used as relay nodes. If R_6 is disconnected from R_2 and connected to R_7, as shown in Figure 5.3 (b), R_2 is eliminated from the communication paths,

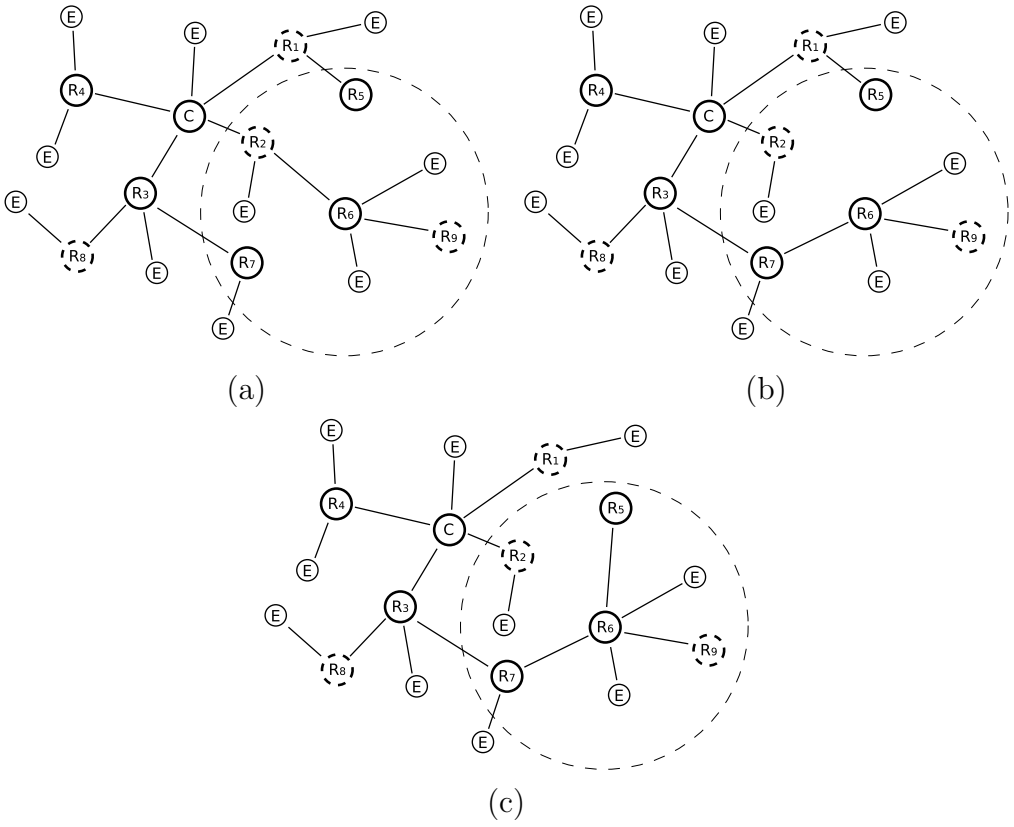

Figure 5.3: Modifying the ZigBee tree topology to change communication paths (reprinted from Fig. 3 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

hence, the amount of traffic load on battery-powered devices is reduced. The next modification would probably be to disconnect R_5 from R_1 and connect it to R_6, as shown in Figure 5.3 (c), leaving all the communication paths free of battery-powered devices. In general, it is not always possible to eliminate all the battery-powered devices on the communication paths due to constraints such as communication range, but following a strategy like the one described certainly reduces the amount of load on them. If a device knows

- the amount of traffic it forwards,
- the paths for each such traffic for a given topology,
- the battery-powered devices on the paths, and

- the alternative devices that it can connect to,

it is possible to follow such a strategy. Fortunately, a router-capable device in a ZigBee network can obtain all this information with minimal or no overhead in terms of network communication. The next section describes how to obtain and use this information to modify the topology.

5.2.1 The Algorithm Details

As explained earlier, in ZigBee tree networks, a router-capable device relays all the traffic between its descendants and the rest of the network. Hence, for such a device it is possible to monitor the source, the destination, and the rate of each flow it forwards.

Algorithm 5.1 Path between two ZigBee devices

function PathBetween(A_s, A_d)

1: $path_s \leftarrow$ PathFromRoot(A_s)
2: $path_d \leftarrow$ PathFromRoot(A_d)
3: $lcp \leftarrow$ longest common prefix of $path_s$ and $path_d$
4: $path \leftarrow A_s$
+ reverse of ($path_s - lcp$)
+ last address in lcp
+ ($path_d - lcp$)
+ A_d
5: **return** $path$

Thanks to the distributed address assignment mechanism of ZigBee, it is possible just by local computation to find the path between any two devices whose addresses are known, meaning that no communication overhead such as route discovery is required. Although not described in the ZigBee specification, we provide a way to compute the path from a device with address A_s to another device with address A_d in Algorithms 5.1 and 5.2. Algorithm 5.2 is used by Algorithm 5.1 to compute the path from the network coordinator to an arbitrary ZigBee node A in the network. Line 5 of Algorithm 5.2 computes the child

of parent p, having node A as a descendant, where $C_{skip}(d)$ is calculated as in Equation (5.1). Hence, starting from the root (lines 1 and 2), at each iteration of the while-loop, the ancestor of node A at depth $d+1$ is found. It is easy to compute the path between any two nodes if the paths between those nodes and the root are known. As described in Algorithm 5.1, to compute the path between A_s and A_d, the common prefix, except for the closest common ancestor, is removed from the paths. Then one of the paths is reversed and concatenated to the other.

Algorithm 5.2 Path from the coordinator

function PathFromRoot(A)

1: $d \leftarrow 0$
2: $p \leftarrow$ coordinator
3: **while** $A \neq p$ **do**
4: $\quad path \leftarrow path + p$
5: $\quad p \leftarrow p + 1 + \lfloor (A - (p+1)) \div C_{skip}(d) \rfloor \times C_{skip}(d)$
6: $\quad d \leftarrow d + 1$
7: **end while**
8: **return** *path*

Unlike the path computation, determining the types of power sources of the devices requires additional communication, as this information does not generally exhibit a predictable pattern. One obvious way to collect this information is to let each battery-powered device register itself to the coordinator and let any device query this information whenever required.

Algorithm 5.3 Load on battery-powered devices

function LoadOnBPNodes(T)

1: $load \leftarrow 0$
2: **for all** forwarded traffic T **do**
3: $\quad path \leftarrow$ PathBetween(T_{source}, $T_{destination}$)
4: $\quad n \leftarrow$ number of battery-powered devices on *path*
5: $\quad load \leftarrow load + n \times T_{rate}$
6: **end for**
7: **return** *load*

Having this information, a router-capable device can now compute the total

load on the battery-powered devices of the network due to communication between its descendants and rest of the network, as in Algorithm 5.3. Note that the battery-powered devices are implicitly obtained from the coordinator on line 4 of the algorithm. In the current form, the battery-powered devices are queried independently for each path. A more efficient approach would be to have a single query for the union of all paths, as they probably contain many common devices, but for the sake of simplicity, the algorithm is described in this way.

The method for finding the load on the battery-powered devices from the descendants of a router-capable device for the current topology, is described so far. In order to modify the topology, it is required that the node learns about alternative neighboring devices that can be connected to and computes the possible loads on the battery-powered devices in the alternative topologies. ZigBee provides neighbor discovery mechanisms, making it possible to determine whether other devices are in the communication range. Due to the distributed address assignment mechanism of ZigBee, it is also possible to compute the paths, obtain the battery-powered devices on the paths, and find out the load on them if a certain alternative router-capable device is chosen as the new parent, in the same way described previously.

As described in Algorithm 5.4, a router-capable device can decide the best parent in terms of load on the battery-powered devices and change its parent if the best case differs from the current one. As a restriction, the new parent should have equal or less depth than the current one (line 5). Otherwise some descendants of the device might not get a network address since the address range assigned to a device decreases as its depth increases. Please note that this is a conservative approach. Even if the depth of a parent candidate is greater, it is possible to obtain a network address for all the descendants of the device if the depth of the deepest device is still less then or equal to Lm (i.e., maximum allowed depth of the tree) after the reconfiguration. Since the depth information of some descendants might be unknown for a device without additional communication, this conservative approach is preferred. On the other hand, even if the parent candidate satisfies the restriction on the depth, it might already have Rm router-capable children, meaning that it cannot accept any other router-capable

Algorithm 5.4 Reconfiguration of the topology

```
1: load ← load in the current configuration
2: n ← 0
3: for all router-capable neighbor N do
4:     c_n.neighbor ← N
5:     if depth_N < depth then
6:         c_n.load ← load if the device connects to N
7:     else
8:         c_n.load ← ∞
9:     end if
10:    n ← n + 1
11: end for
12: sort c in ascending order w.r.t. load values
13: i ← 0
14: while i < n and not connected to a new parent do
15:    if c_i.load < load and router-capable child count of c_i.neighbor < Rm then
16:        connect to c_i.neighbor
17:    end if
18:    i ← i + 1
19: end while
```

child. Hence a device requires explicit permission of the parent candidate before reconfiguration (line 15).

Note that the term *connection* mentioned in the 6^{th} and the 16^{th} lines of Algorithm 5.4 refers to the connection of the device with all of its descendants as a subtree. As mentioned in Section 5.1.2, whenever a device changes its parent, its address and the addresses of all its descendants have to be changed as well. The descendant devices are informed about the old and new addresses of the device starting the reconfiguration (i.e., root of the subtree) and each descendant utilizes this information to compute the updated addresses of its parent, children, and itself as described in Algorithm 5.5. Next, updated addresses are used to re-join to the network by orphaning mechanism described in the ZigBee specification.

Algorithm 5.5 takes old (R_o) and new (R_n) addresses of a device R, and old address (D_o) of another device D as input and returns either the new address of D, if it is a descendant of R or $\perp$, otherwise. In line 2, the algorithm checks whether D is a descendant of R. If so, in the while-loop between lines 5 and 11, the location of D in the subtree is traced starting from the root address R_o and

Algorithm 5.5 Obtaining updated address of a device

function ObtainUpdatedAddress(R_o, R_n, D_o)

```
1: D_n ←⊥
2: if R_o < D_o and D_o < (R_o + C_skip(depth_{R_o} − 1)) then
3:     D_n ← R_n
4:     A_o ← R_o
5:     while A_o ≠ D_o do
6:         skip_o ← C_skip(depth_{A_o})
7:         skip_n ← C_skip(depth_{D_n})
8:         index ← ⌊(D_o − (A_o + 1)) ÷ skip_o⌋
9:         A_o ← A_o + 1 + index × skip_o
10:        D_n ← D_n + 1 + index × skip_n
11:    end while
12: end if
13: return D_n
```

using its old address D_o and at each iteration of the loop, new address D_n of D is updated according to this location information given that the new root address is R_n.

Since the network addresses of the devices change, the rest of the network should be informed about these address changes to route the data packets to the correct devices. The new addresses are also required by the devices to update their power source information caches in which power source information is stored along with the network addresses to compute the load on the battery-powered devices whenever required. To disseminate this information only the address change of the device starting the reconfiguration is broadcast in the network. Receiving the old and new addresses of the root of the subtree, which consists of devices whose addresses are updated, a device checks all the addresses it is interested in to see whether they belonged to the subtree and if so updates them accordingly using the function given in Algorithm 5.5. Since successful dissemination of address updates has vital importance for the ongoing communications and power source information, which is required for later reconfigurations, a reliable method for broadcast should be chosen. In our current implementation, we transmit broadcast messages at most three times and try to limit the number of retransmissions applying passive acknowledgement mechanism as the ZigBee

specification suggests. Other methods such as the one presented in [94] can also be utilized.

As long as a network has a stable traffic characteristic and the devices it is composed of remain the same, the network is expected to converge to a topology in which battery-powered devices are avoided as much as possible. Because at each reconfiguration, topology is modified in a way that the total load on the battery-powered devices are reduced and reconfigurations occur as long as better topologies are found in terms of load on the battery-powered devices. But the algorithm can handle changes in the traffic characteristics (e.g., communicating pairs, bandwidth requirements, etc.) and members of the network since the routers constantly monitor the packets they forward and react accordingly. Therefore, new reconfigurations may take place to adapt to new situations.

5.2.2 Implementation

We implemented the PSAR algorithms presented in Section 5.2.1 fully and efficiently in ns-2 (version 2.31) simulation environment [6], on top of the IEEE 802.15.4 and ZigBee protocol stacks. A module implementing 802.15.4 was already in ns-2, and we utilized that with slight modifications done wherever required (e.g., to support network addresses in addition to device addresses). However, at the time we did our simulations, there was no publicly available ZigBee module for ns-2, hence, we implemented the required parts of the ZigBee standard (that is, address assignment, routing, broadcast, rejoining, etc.) and integrated them with ns-2.

During a reconfiguration, a subtree of devices disconnects from a parent and connects to another as a whole, preserving the topology within the subtree. Therefore the load on the battery-powered devices of the subtree remain the same before and after the reconfiguration. Hence, in our PSAR implementation, as an efficiency measure the 4^{th} line of Algorithm 5.3, and in turn, Algorithms 5.1 and 5.2, is implemented to avoid paths from the current node (i.e., the node executing the algorithm locally) to its descendants. Such an implementation choice

reduces the bandwidth required to obtain the power sources of the nodes on those paths. On the other hand, to preserve the consistency of the network, simultaneous topology reconfigurations are not allowed. Otherwise, nodes may try to connect to nodes which are actually in the middle of an independent reconfiguration. In the current implementation, permission of the coordinator is obtained to begin a reconfiguration and the coordinator allows only one reconfiguration at a time.

Table 5.1: 802.15.4 and ZigBee commands utilized in the implementation of the proposed algorithm.

Command	**Specified In**	**Purpose**
`MLME-SCAN`	802.15.4	Used for discovering other devices in the communication range.
`NLME-LEAVE`	ZigBee	Used for disconnecting a subtree from the network.
`NLME-JOIN`	ZigBee	Used for reconnecting all nodes in the subtree. Since preserving the topology of the subtree is desired, rejoin through orphaning procedure is applied.
`NLME-DIRECT-JOIN`	ZigBee	Used for preparing the candidate parent for the new child node.
`Power_Desc_store_req` & `Power_Desc_store_rsp`	ZigBee	Used for storing the power source information of the devices in the coordinator.
`Power_Desc_req` & `Power_Desc_rsp`	ZigBee	Used for retrieving the power source information of the devices from the coordinator.

One of the advantages of the described scheme is that it can be implemented mostly using existing commands of the IEEE 802.15.4 and ZigBee specifications. Table 5.1 lists the commands directly used in the implementation of PSAR. These commands in turn uses other commands such as `NLDE-DATA`. The scheme, which makes use of the listed commands, is implemented as a ZDO in the APL layer. Since a ZDO cannot access certain functionalities such as monitoring forwarded packets at the NWK layer, minor modifications are required to make necessary

information available to the algorithm. On the other hand, there are cases in which the algorithm residing at the APL requires direct access to some of the functionalities provided by the lower layers, skipping the NWK. As an example, `MLME-SCAN` is used by the NWK only if the device is currently not connected to a network, whereas the algorithm needs to discover nearby devices even if it is already part of a network. Therefore our solution also utilized cross-layering approach.

5.2.3 Analysis

One of the aims of the ZigBee specification is to make the design and production of low-cost devices possible. Reducing the complexity of the hardware is an important part of this goal and software running on such reduced hardware is required to have low memory and processing power demands. First part of this section presents an asymptotic analysis of memory and processing power requirements of PSAR and its possible implementations. In the second part of the section, message complexity of PSAR is discussed.

First of all, the algorithm requires monitoring and keeping track of the forwarded traffic. For each communicating pair whose traffic is forwarded (in other words, for each traffic flow forwarded), the network addresses and the associated bandwidth requirements should be stored and updated. There are several options for storing this data, each having advantages and disadvantages in terms of memory (i.e., space complexity) and processing power (i.e., time complexity) requirements. The most straightforward implementation is to use a list, whose space complexity is $O(m)$ where m is the number of communicating pairs to be tracked. On the other hand, for each data packet forwarded, the list should be sequentially searched and the corresponding element should be updated, which has a time complexity of $O(m)$. An alternative, as preferred for the implementation of the simulation, is to use a binary search tree (BST). A BST has the same space complexity (i.e., $O(m)$) as the list implementation, and although it has a larger overhead per communicating pair to be tracked, the gain is in the $O(\log m)$ search time. Other options are also possible, all of which can be used for tracking

the forwarded traffic, such as a sorted array with $O(m)$ space and $O(\log m)$ time complexity but costly maintenance as new communicating pairs arise, or a hash map with amortized $O(1)$ time complexity but possibly higher space complexity.

Having the communicating pairs and associated bandwidth requirements, nodes can decide whether a reconfiguration is necessary. As shown in Algorithms 5.1, 5.2, and 5.3, for each communicating pair the path between them should be computed. This leads to an average time complexity of $O(m \log n)$, assuming the depth of the tree-shaped network is bounded by $O(\log n)$ on average, where m is the number of communicating pairs whose traffic is forwarded by the current node and n is the total number of nodes in the network. In the worst case, there can be at most $O(n^2)$ flows (communicating pairs) going over a node of the network. This is, however, quite a loose upper bound. The paths can be computed each time they are required, as in the current implementation of the simulation, or cached, which increases the memory overhead and is probably not preferable. Once the paths are known, each node should be tested against its power source to compute the load on the battery-powered devices (see Algorithm 5.3, lines 4 and 5). Assuming the power source information is cached after it is obtained from the coordinator, to prevent unnecessary communication, the power source of the device on the computed paths can be searched from this cache. The cache can be implemented as a BST or hash map to favor time complexity or as a simple list to favor space complexity. Considering the possible number of unique nodes on the paths, the power source cache is implemented as a BST in the simulations, meaning that in the current implementation load computation has a time complexity of $O(m \log^2 n)$.

Until now possible space and time complexities of PSAR depending on the implementation alternatives have been presented. PSAR also requires extra control messaging as described in the previous sections and this part analyzes this messaging overhead asymptotically. Let n be the total number of nodes in the network and k be the number of nodes in the subtree to be connected to another node in the network. There are two groups of messages: one is exchanged once in a lifetime of a network and the other is once per reconfiguration. Messaging required to register power source information of a device when it connects to a

network and to query this information to compute the load on the battery powered devices belongs to the first group. $O(n\log n)$ messages are exchanged for both registration and querying, assuming a tree depth of $O(\log n)$. Since the ZigBee networks are expected to have long lifetimes, overhead of this group of messaging is considered to be negligible. On the other hand in each successful reconfiguration attempt $O(\log n)$ messages are exchanged to inform PAN coordinator about the start and end of a reconfiguration, $O(\log n)$ messages are exchanged to have permission of the parent candidate, $O(k)$ messages are exchanged for network leave and network join operations, and finally $O(n)$ messages are exchanged to inform the network about the address change of the device which initiates the reconfiguration. Hence neglecting the initial one-time overheads and considering $k < n$, the algorithm requires $O(n)$ messages per reconfiguration.

5.3 Simulation Results

This section presents simulation results illustrating the performance of PSAR. In the simulations, several parameters are fixed. The communication band is set to 2450 MHz and Cm, Rm, and Lm values are 6, 6, and 6, respectively. Furthermore, the superframe structure is not applied and all the devices in the network are chosen to be FFD. On the other hand, several parameters are changed to observe the impact of different conditions on the performance of the algorithm. These parameters, along with their chosen values, are network size (10, 40, 70 devices), density (one device per 24, 16, and 8 m^2), battery-powered device ratio (10%, 20%, ..., 90%) and ratio of data flow count to total number of devices (10%, 30%, 50%). Note that we use node count, not the number of all possible pairs, while limiting the traffic flows. Otherwise, the number of flows (pairs) would be excessive. For each combination of aforementioned parameters, the results are averaged across 100 simulations (disconnected topologies due to communication range and the orphan problem [95] are eliminated), in each of which node locations, battery-powered devices, and communicating nodes are determined pseudo-randomly, as described in [89].

Traffic flows are constant bit rate (CBR) flows with two-seconds data generation interval and a random packet size between 2 and 50 bytes. In each simulation, 120 minutes of communication is simulated and each device is configured to check for a possible reconfiguration every 20 minutes with a randomization of $\pm$ 20 seconds, to prevent simultaneous reconfiguration attempts. Another alternative for triggering the algorithm on a device is to wait until a significant change occurs in the traffic observed by that device. In the majority of the simulation results presented in this section, the communicating pairs are fixed (i.e., static traffic) for a simulation run. But the results in which communicating pairs change during a simulation run (i.e., dynamic traffic) are also given to present how PSAR copes with the dynamic traffic scenarios. Please note that, the static traffic case can be interpreted as a stable portion of a longer and dynamic (in terms of communication demands and device arrivals and departures) traffic case.

Currently, to the best of our knowledge, there is not a similar study with PSAR in the literature that is adapting the tree topology dynamically with respect to the traffic demand to reduce the load on battery-powered devices. Therefore, simulations are repeated in the presence and the absence of PSAR (i.e., using ZigBee tree routing reported in the specification), keeping all the remaining parameters intact for both cases. Change (percent reduction) in the following metrics are measured to evaluate the performance of the algorithm: total amount of data forwarded by all the battery-powered devices, standard deviation of the forwarded data by the battery-powered devices, and average path lengths between communicating devices. Percent reduction is defined as in Equation (5.2). Apart from the above, packet drops and communication overhead due to PSAR are also measured.

$$\text{Reduction} = \frac{\text{Value w/o PSAR} - \text{Value w/ PSAR}}{\text{Value w/o PSAR}} \times 100 \tag{5.2}$$

Figures 5.4, 5.9, 5.10, 5.11, and 5.12 depict the reduction in total traffic load of the battery-powered devices, the reduction in the standard deviation of the traffic loads of the battery-powered devices, the reduction in average path lengths between communicating node pairs, the packet drop rates, and control packet ratio for the static traffic case, respectively. In these figures, the columns present

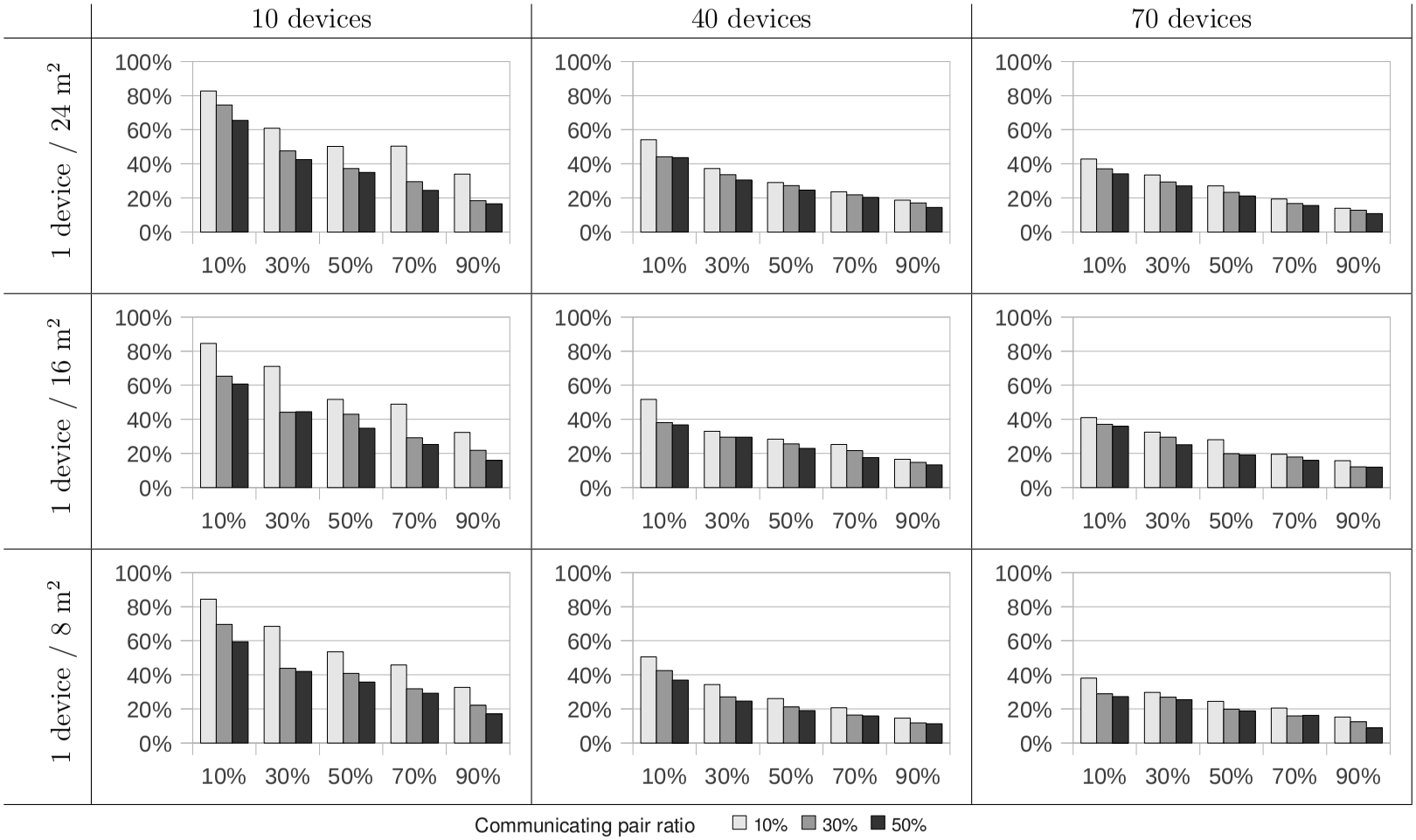

Figure 5.4: Percent reduction in traffic load on the battery-powered devices (x-axis: battery-powered device ratio, y-axis: percent reduction) (reprinted from Fig. 5 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

simulation results for the network sizes of 10, 40, and 70 devices and the rows present simulation results for the networks with densities of one device per $24\,m^2$, $16\,m^2$, and $8\,m^2$. The network size increases from left to right and the device density increases from top to bottom. In each graph, values for three different communicating pair ratio (i.e., ratio of communicating pair, or traffic flow, count to the total node count) cases are given. Figure 5.8 presents the reduction in total traffic load of the battery-powered devices for the dynamic traffic case.

Before analyzing the effect of different parameters on the performance of PSAR, let us show how PSAR helps increasing the lifetime of a network. In a network that is composed of both battery- and mains-powered devices, lifetime of the network directly depends on the lifetime of the battery-powered devices. As shown in Figure 5.4, if PSAR is applied, although additional control packets need to be forwarded, traffic forwarded by a battery-powered device is reduced on the average, which means each battery-powered device has a longer expected lifetime. This result does not necessarily mean that the lifetime of the network is increased, since some battery-powered devices might die much earlier than the case without PSAR (although on the average the battery-powered devices live longer), leaving some portions of the network unreachable. But Figure 5.9 shows that if PSAR is applied, standard deviation of the traffic forwarded by the battery-powered devices is also reduced, meaning that the lifetime of the battery-powered devices are distributed more evenly. Hence we claim that lifetime of the network increases, since the lifetime of each battery-powered device increases.

As far as the percent reduction in total traffic load on the battery-powered devices are concerned, values as high as 80%, 50% and 40% are observed for the network sizes of 10, 40 and 70, respectively (see Figure 5.4). But as the battery-powered device ratio increases, the percent reduction in total traffic load decreases to values as low as around 10%. There are two obvious reasons for this decrease in the traffic load percent reduction. First, an increase in the number of battery-powered devices does not correspond to an increase at the same rate in the number of battery-powered devices avoided from the paths between communicating pairs, because it gets harder to find paths without battery-powered devices. Second, even if some of the battery-powered devices are avoided in networks with

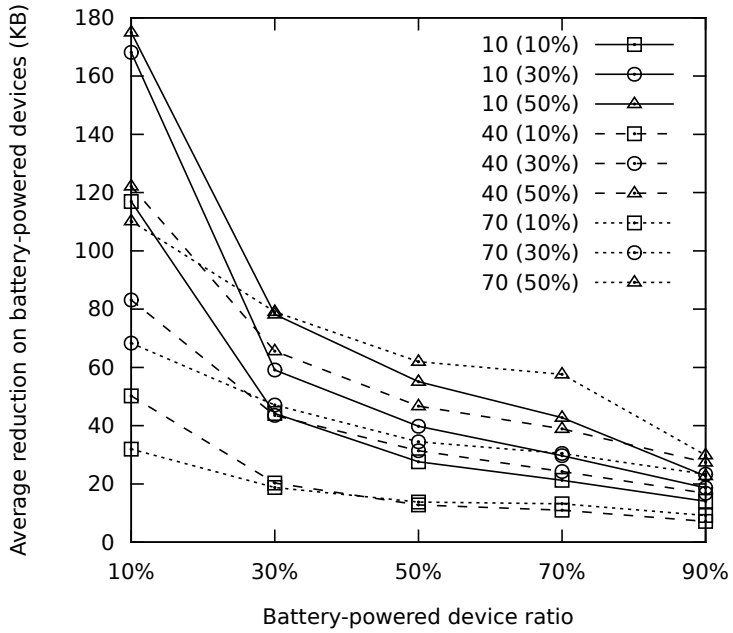

Figure 5.5: Average reduction in traffic load on the battery-powered devices for the network sizes of 10, 40, and 70 devices and communicating pair ratios of 10%, 30%, and 50% (reprinted from Fig. 6 (a) of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

a higher battery-powered device ratio, the total load on all battery-powered devices is so high that the reduced amount of load does not have a comparatively significant value. Another observation is that as the network size increases, the percent reduction in traffic load on the battery-powered devices decreases. The reason is similar to the previous argument, that is, although the amount of traffic load avoided from battery-powered devices does not change significantly, since the number of battery-powered devices (and therefore, the total load on them) increases, the significance of the avoided traffic load decreases.

Figures 5.5 and 5.6 support these arguments: as the battery-powered device ratio increases, average reduction per battery-powered device decreases while the total reduction on all the battery-powered devices increases. Furthermore, larger networks have better average and total reduction values in bytes although they have worse percent reduction values since the avoided traffic does not keep up with the increase in the number of battery-powered devices. In Table 5.2, total amount of traffic (i.e., all traffic sent from the source nodes and forwarded by the

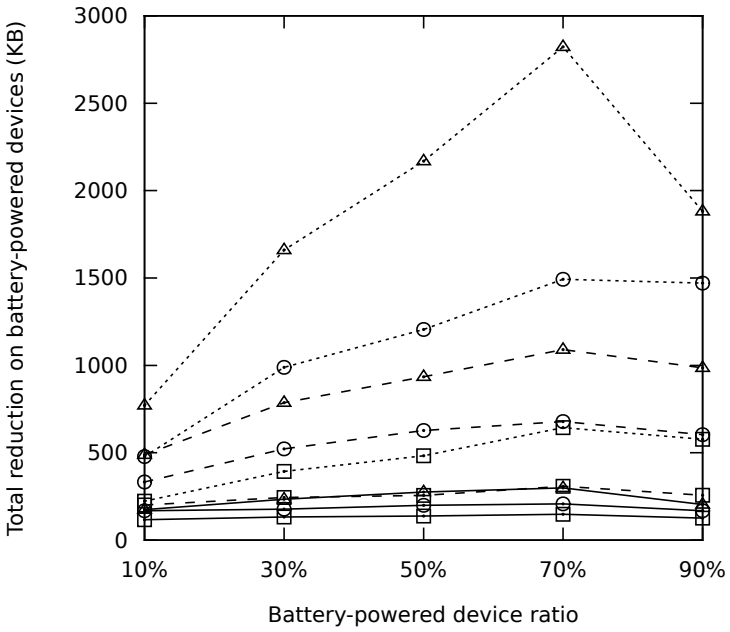

Figure 5.6: Total reduction in traffic load on the battery-powered devices for the network sizes of 10, 40, and 70 devices and communicating pair ratios of 10%, 30%, and 50% (reprinted from Fig. 6 (b) of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

intermediate nodes) for different network sizes and communicating pair ratios is given to compare with the values presented in Figures 5.5 and 5.6.

Table 5.2: Total traffic in KB for different network sizes and communicating pair ratios.

	10 devices	**40 devices**	**70 devices**
10%	523	2645	5592
30%	1210	7836	17448
50%	1959	13373	30741

In Figures 5.4, 5.5, and 5.6, reduction for the first two hours of different networks are given. Differently in Figure 5.7, percent reduction values until different time points from the beginning of the network is presented (e.g., y value which corresponds to the 60 in the x-axis is the percent reduction for 0-60 period). Figure 5.7 also depicts the number of successful reconfigurations for the last time period (e.g., y value which corresponds to the 60 in the x-axis is the number of

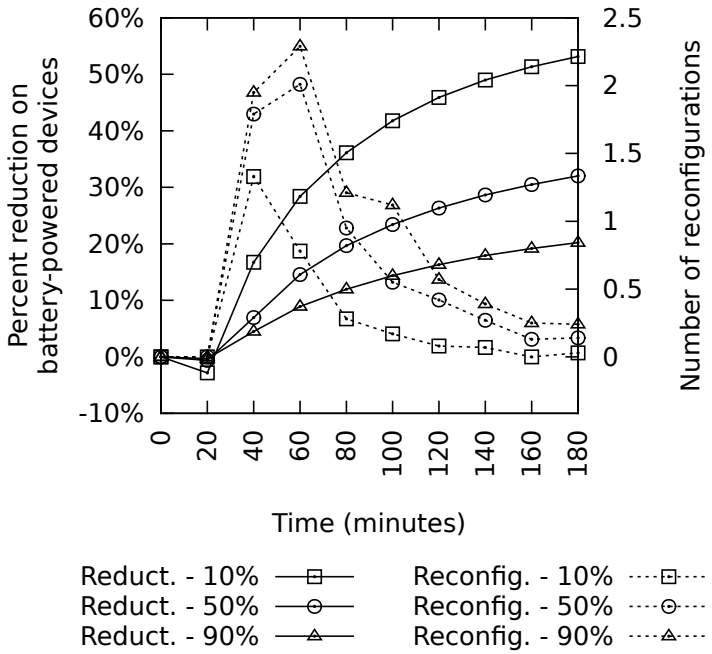

Figure 5.7: Percent reduction in traffic load on the battery-powered devices and number of configurations over time for the network size of 40 devices and communicating pair ratio of 30% (reprinted from Fig. 7 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

reconfigurations for 40-60 period). As it can be seen from the figure, reduction increases over time although its rate decreases and tends to converge to a certain value. Additionally, for the first 20 minutes of the network lifetime, negative reduction values are obtained, since there have been messaging overhead on the battery-powered devices due to PSAR although there has not been any reconfigurations to reduce the amount of traffic forwarded by the battery-powered devices. Note that the number of reconfigurations peaks early in the network lifetime and decreases rapidly, meaning that the network topology converges rather quick considering a node can attempt for a reconfiguration once in every 20 minutes and simultaneous reconfigurations are not allowed. Also note that in Figure 5.4 reduction values are given for the first 120 minutes of the network and Figure 5.7 shows that the percent reduction continues to increase after this period, as long as the network traffic stays the same.

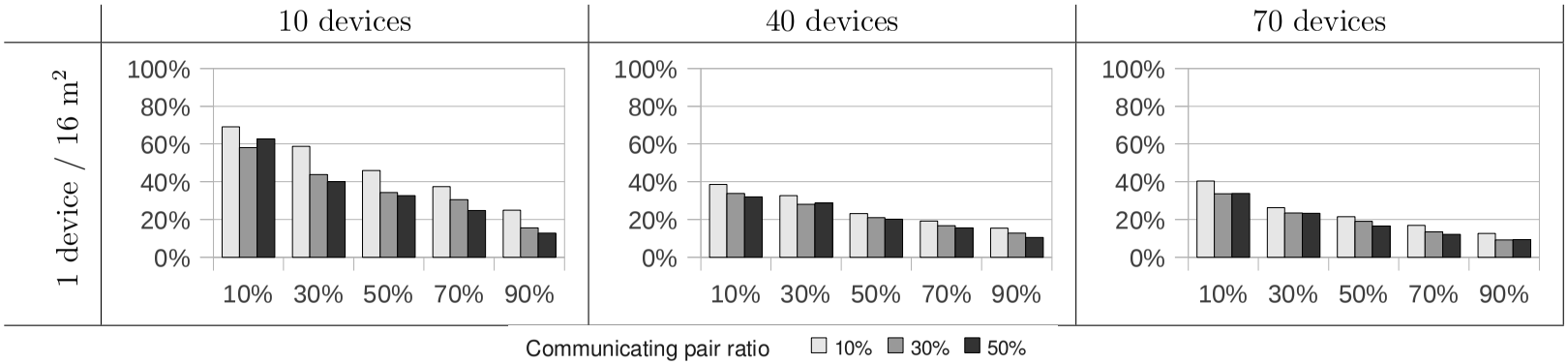

Figure 5.8: Percent reduction in traffic load on the battery-powered devices for the dynamic traffic case (x-axis: battery-powered device ratio, y-axis: percent reduction) (reprinted from Fig. 8 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

Figure 5.8 gives the similar set of results with the ones in Figure 5.4 for the dynamic traffic case. In the simulations with the dynamic traffic scenario, the communicating devices are changed in the middle of the simulation period (i.e., around 60^{th} minute). As it can be observed from the figures, in the dynamic traffic case, the percent reductions are slightly less compared to the static traffic case. This is expected since the benefit obtained due to reconfigurations has effect for less amount of time in the dynamic traffic case. As the traffic characteristics change, current topology, which is the result of previous reconfigurations, would probably not be the optimal one, as far as the load on the battery-powered devices is concerned. Since, recognizing the current traffic characteristics and adapting the topology accordingly take time, dynamic traffic patterns have negative impact on the performance of PSAR.

As shown in Figure 5.9, PSAR is also able to decrease the standard deviation of the traffic load on the battery-powered devices. This result means that the load on the battery-powered devices is not only reduced but also distributed more evenly, as stated earlier. The primary reason for the reduction in the standard deviation is that since the load on the most of the battery-powered devices are reduced or completely eliminated, the quantity of the differences is also reduced. The reduction in the standard deviation exhibits a similar characteristic with the reduction in the traffic itself, that is, the reduction in the standard deviation decreases from around 60% to below 20%. The reason for the decrease in the traffic load reduction described previously, largely applies to this case as well. The number of battery-powered devices avoided from the communication paths shows little change as the number of battery-powered devices increases, hence, the effect of the algorithm remains limited in the variation of the traffic load on them. As the network size increases, the reduction in standard deviation of traffic load on the battery-powered devices decreases, due to similar reasons given for the traffic load case.

Our experiments are designed to run on different device densities to observe the effect of device density on the effectiveness of PSAR. But as can be seen from the figures, neither the reduction in traffic load on the battery-powered devices nor the reduction in the standard deviation of the loads on the battery-powered

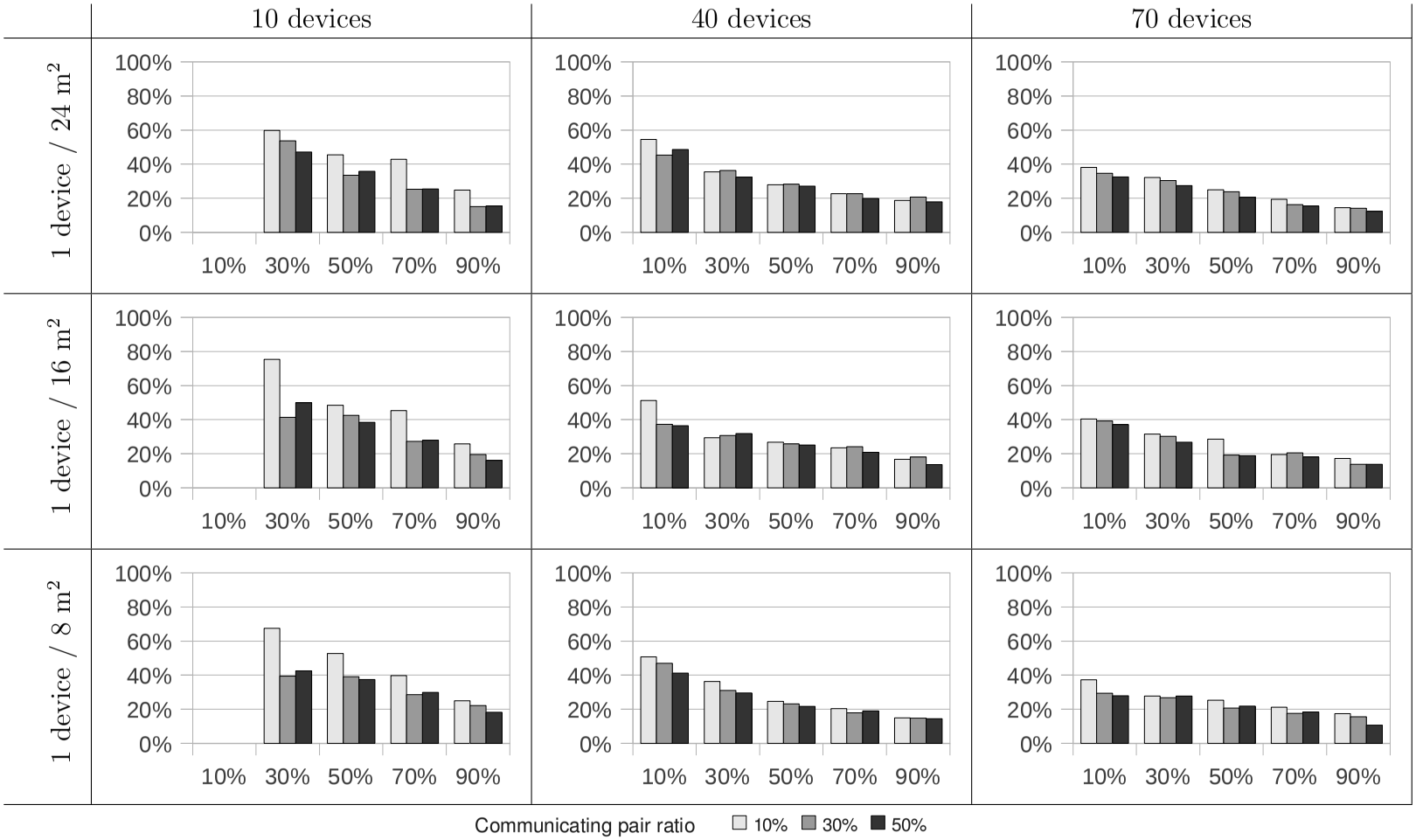

Figure 5.9: Percent reduction in standard deviation of traffic load on the battery-powered devices (x-axis: battery-powered device ratio, y-axis: percent reduction) (reprinted from Fig. 9 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

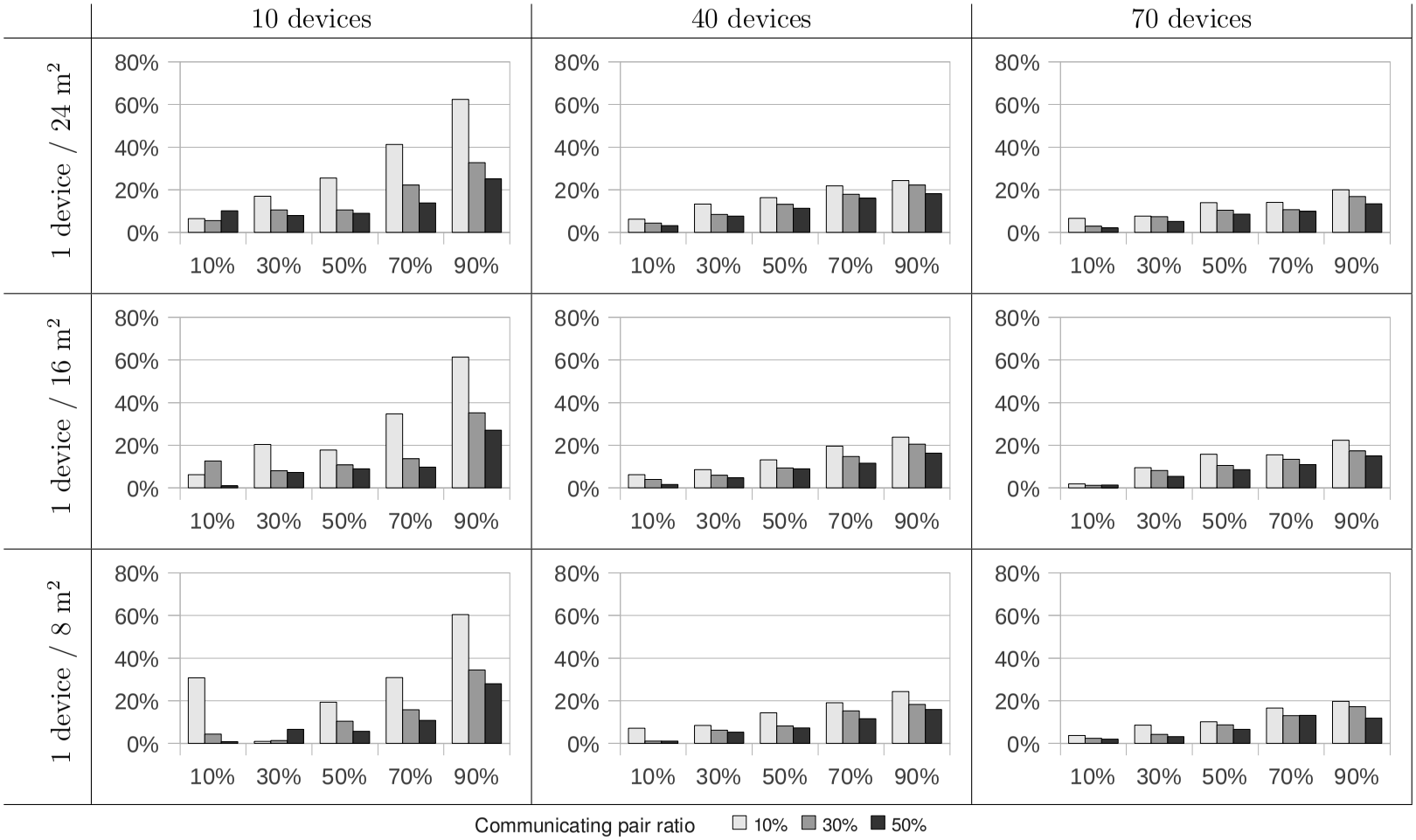

Figure 5.10: Percent reduction in average path lengths between communicating devices (x-axis: battery-powered device ratio, y-axis: percent reduction) (reprinted from Fig. 10 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

devices are affected by device density significantly.

Although the primary concern of reconfigurations in PSAR is to reduce the traffic load on the battery-powered devices, it also helps to reduce the average path lengths between communicating devices, as shown in Figure 5.10. There are two reasons for this side benefit. First, if there are more than one reconfiguration alternatives with equal reduction amounts, which is a rare case, then the one with a shorter path is preferred. Second, a node marks another node as a new parent candidate only if that node has equal or less depth value, reducing the average depth of the tree, hence its diameter.

As described in Section 5.2.1 once the network address of devices change due to reconfigurations, the address changes are advertised using a broadcast message. As a negative effect, between the time a destination changes its address and the corresponding source recognizes the change, the data packets are sent to the old address of the destination, which leads to packet drops. Hence, one of the aims of the experiments is to see the effect of the algorithm on the packet drops due to disconnections during the reconfigurations. As presented in Figure 5.11 as the network size increases from 10 to 70 devices the packet drop rate increases from below 0.01% to around 0.1%. These results correspond to less than 1, 5 and 20 packet drops on the average for network sizes of 10, 40 and 70 nodes, respectively, given that 2 hours of communication, CBR with two-seconds intervals and 50% communicating pair ratio (i.e., 5, 20 or 35 data flows).

The communication overhead of PSAR is also observed in the experiments and the results are given in Figures 5.12 and 5.13. The communication overhead of the algorithm is mainly due to registering the power source of the devices, query the power source of the devices, request connection from a parent candidate, inform the subtree about an upcoming reconfiguration, and inform the rest of the network about new network addresses after the reconfiguration. Not all the communication ends up with a successful reconfiguration, due to reasons such as not being a better alternative found after the power sources are queried or because a parent candidate does not accept the new connection. Hence, two approaches are applied to measure the control packet overhead traffic due to

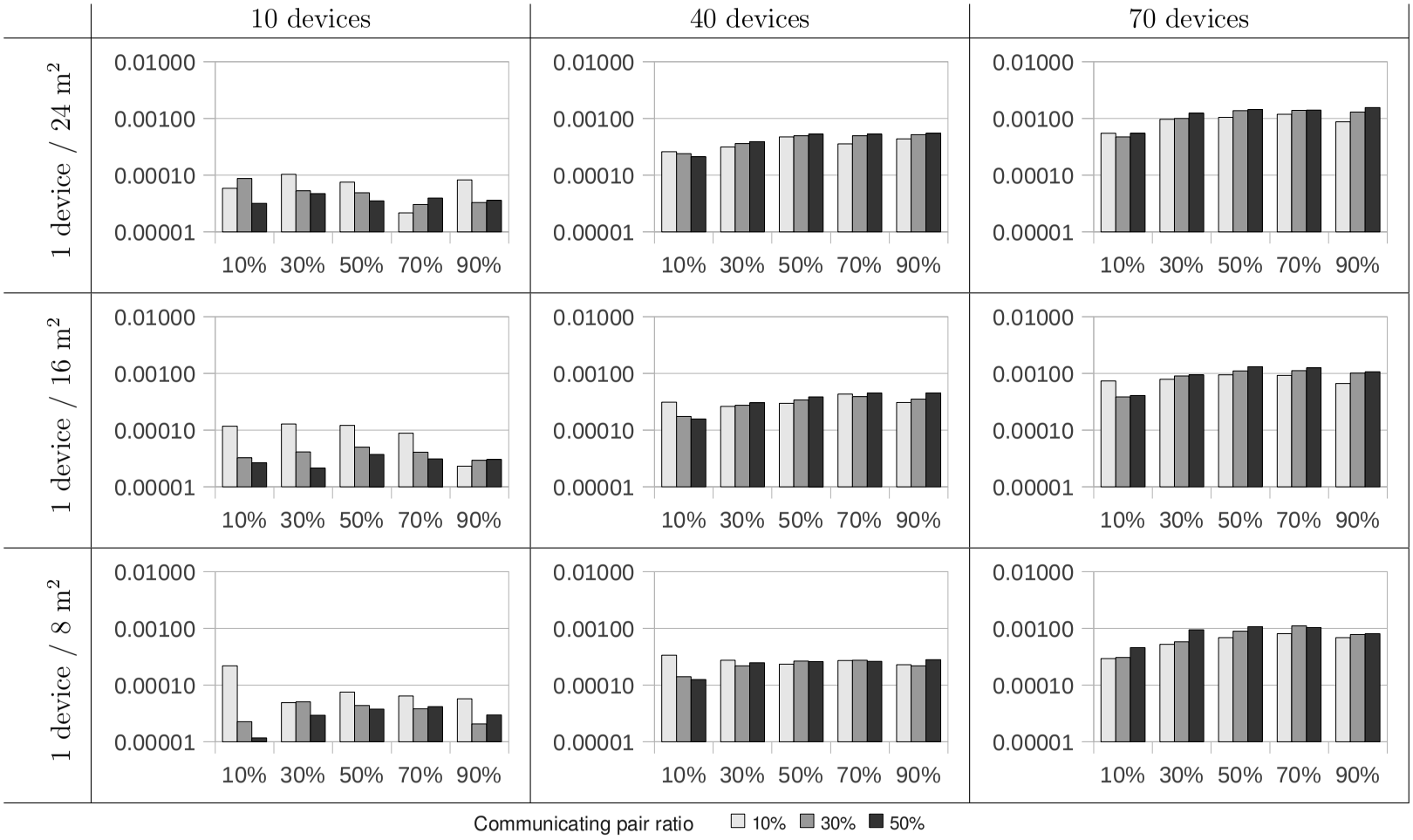

Figure 5.11: Packet drop rate (x-axis: battery-powered device ratio, y-axis: drop rate in packets per second) (reprinted from Fig. 11 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

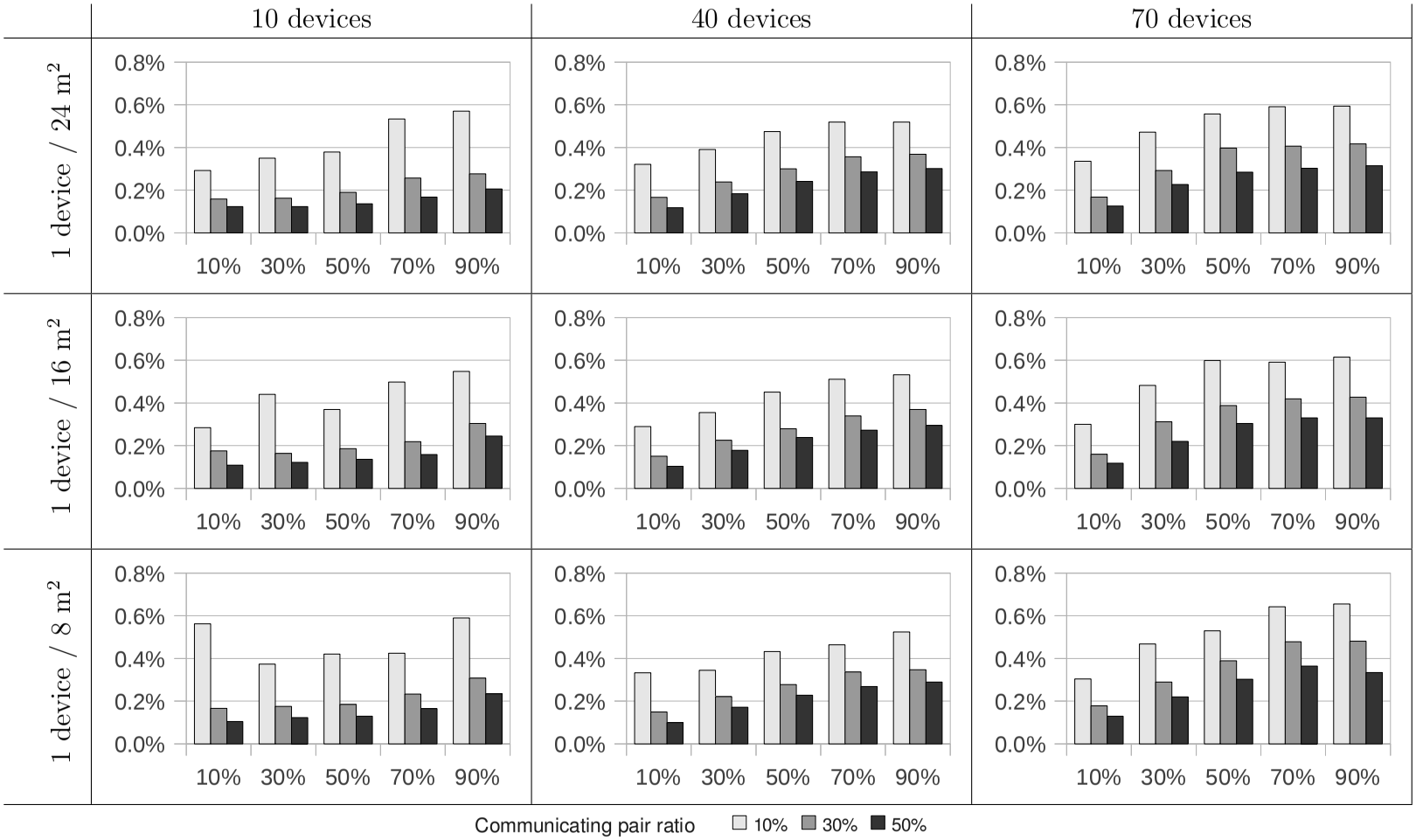

Figure 5.12: Ratio of control packet traffic to data traffic (x-axis: battery-powered device ratio, y-axis: control packet ratio) (reprinted from Fig. 12 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

PSAR. The first approach is to find out the ratio of control packet traffic to the actual data traffic (i.e., amount of control packet traffic divided by the amount of data traffic) and the second approach is to measure number of control packets per reconfiguration (i.e., total number of control packets divided by the total number of reconfigurations). As shown in Figure 5.12, the control packet ratio is always below 0.7%. Note that the network size does not have an observable effect on the ratio, because the control packet traffic and the data traffic increase at the same rate as the network size increases. The communication overhead is approximately 60, 170, and 260 packets per reconfiguration for the network sizes of 10, 40, and 70 devices respectively, as depicted in Figure 5.13. Hence the number of control packets per device per reconfiguration is around 6 for the network size of 10, while it is below 5 for the network sizes of 40 and 70. This is due to the less number of reconfigurations for the network size of 10 devices in which the effect of initial communication overhead per reconfiguration is higher.

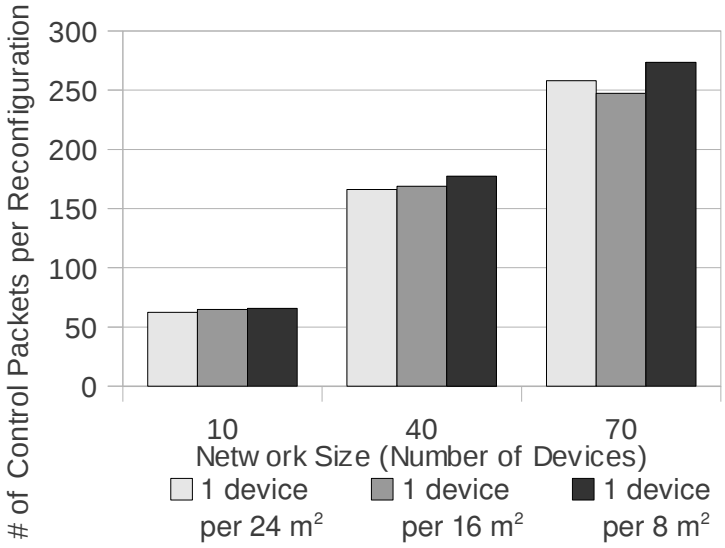

Figure 5.13: Change in control packet count per reconfiguration with respect to network size (reprinted from Fig. 13 of [2] © Springer Science+Business Media, LLC 2012, with kind permission from Springer Science and Business Media).

5.4 Conclusions

In this chapter we propose a distributed algorithm, PSAR, to reduce the traffic load on the battery-powered devices in tree topology ZigBee networks. The basic approach is to route the network traffic through mains-powered devices instead of battery-powered devices as much as possible. In order to achieve this, the topology must be modified as there is only a single path between any two nodes in a tree. New topology is decided by local computations with minimal communication to gather the required information. Simulation results show that the reduction in traffic routed via battery-powered devices is as high as 80% in some cases. Simulation results also show that PSAR reduces the standard deviation of the traffic load on the battery-powered devices so that the energy consumption is distributed more evenly among those devices. These benefits are obtained with insignificant communication overhead (due to control packets) and packet losses (due to disconnections during reconfigurations).

Chapter 6

Conclusions and Future Work

In this thesis, we propose several energy-efficient routing solutions for wireless ad hoc sensor networks consisting of nodes with different power sources to extend the network lifetime. We mainly focus on WSNs in which battery- and mains-powered nodes coexist, but our proposed solutions can also be applied in different settings such as where some of the nodes are powered by energy harvesting methods.

We first provide an approach for power-source aware routing, which basically tries to construct a backbone structure to route packets from all sensor nodes to one or more sinks. The constructed backbone is a tree (or a forest depending on the sink count) containing all the mains-powered nodes and the sinks. It may also include some of the battery-powered nodes, if the mains-powered nodes and the sinks do not form a connected topology. Rest of the battery-powered nodes are connected to this backbone. Since the data packets are forwarded over this backbone mainly by the mains-powered nodes, precious energy of the battery-powered nodes is preserved.

Next, based on this approach, we propose an algorithm framework that specifies the main steps of a procedure to construct such a backbone. The framework may have different alternative algorithms for some of its steps. Such a framework gives means to produce different algorithms with the same main behavior but different characteristics by altering the steps with different options. We describe four

such centralized algorithms and by means of simulations we evaluate their performance by comparing them with a basic algorithm, which does not distinguish between battery- and mains-powered nodes. Our simulation results reveal that favoring mains-powered nodes on the routing paths using our algorithms increases the network lifetime more than 20% even when the mains-powered node ratio is as low as 5% and close to theoretical upper-bound when the mains-powered node ratio is around 25%.

We also propose a distributed version of one of our centralized routing algorithms obtained by using our algorithm framework. We give a detailed design of our distributed algorithm, called PSABR, in which we include the description of the control messages as well as the node behaviors which are different for battery- and mains-powered nodes. We also provide extensive simulation results validating the correctness and robustness of the algorithm and measuring its effectiveness and efficiency. We show, using ns-2 simulations, that it is possible to increase network lifetime significantly by favoring mains-powered nodes as the relay nodes.

In our distributed algorithm, we assumed that the path-cost metric between two mains-powered nodes is the number of battery-powered nodes on the shortest path connecting these two mains-powered nodes. Such a cost metric is useful for reducing the number of battery-powered nodes on the routing paths. Another cost metric between two mains-powered nodes could be a value inversely proportional with the number of vertex disjoint paths consisting of battery-powered nodes. Such an approach can be useful for increasing the fault-tolerance of the routing tree by increasing the connectivity alternatives of mains-powered nodes in case a routing path between two mains-powered nodes fails. This cost metric can also be used to achieve a more balanced energy usage by sending each packet through a different vertex disjoint path from one mains-powered to another. This approach can be studied in a future work.

Finally, we propose a distributed routing topology construction and maintenance algorithm, called PSAR, for 802.15.4/ZigBee based wireless networks. In its basic approach, that is making use of mains-powered nodes and eliminating

battery-powered nodes on the routing paths, it is similar to our previous algorithms. On the other hand, it is different than our previous algorithms in some other aspects. First, it is designed for a specific wireless communication technology. Second, it can route peer-to-peer as well as many-to-one traffic. Third, it can adapt to changes in traffic patterns, not only to node failures. Therefore, it is suitable for wireless sensor and actuator networks as well, besides WSNs.

We performed extensive ns-2 simulations to investigate the performance of our PSAR algorithm with respect to different parameters, such as node density, node count, and mains-powered node ratio. Our simulation results show that our distributed algorithm can reduce the load on the battery-powered devices and maintain a more balance energy usage among battery-powered devices, effectively.

As we mention in Chapter 1, we also investigate different aspects of algorithms introduced in this thesis by analyzing their similarities and differences. In Chapter 4, we give a distributed version of one of our centralized routing algorithms proposed in Chapter 3. Implementation of a centralized algorithm is relatively simple, but it introduces additional communication overhead, since the required global topology information should be collected at a central location and after route computation, the nodes should be informed about the routing paths. A distributed algorithm, on the other hand, is more complex, because nodes should decide the routing paths with limited local information while trying to prevent anomalies, such as routing loops. But a distributed algorithm can repair local route failures easily with low overhead by exchanging relatively small amount of control messages.

We also analyzed the effect of sink count in Chapter 4. As expected, as the sink count increases, the average path length to the sinks decreases. Interestingly, however, we observed that sink count may not always have a significant effect on network lifetime. It depends on the algorithm. For example, as far as the network lifetime is considered, our algorithms are affected slightly by the number of sinks. On the other hand, the basic algorithm that we used for comparisons exhibited around 40% performance increase as the sink count is increased from one to four.

We propose both wireless technology independent algorithms as in Chapters 3

and 4 and an algorithm designed for a specific wireless technology, i.e., ZigBee, as in Chapter 5. While designing wireless technology independent solutions we were required to work also with some primitive functionalities, but we had some degree of freedom in design, while, for example, designing our own control messages. On the other hand, ZigBee provided a rich set of functionality to work with, such as device discovery and distributed address assignment mechanism, but we also had to comply with the restrictions imposed by the protocol, since we tried to keep modifications and extensions to ZigBee at minimum.

For our generic algorithms presented in Chapters 3 and 4, we assumed that all the sensor nodes have periodic data to send to specific nodes (i.e., sinks), hence the data rate is rather stable and destinations are fixed. We designed our algorithms accordingly, that is, we only handled node failures, rather that trying to handle changes in the traffic patterns. Differently, in our ZigBee specific algorithm, considering the application areas of ZigBee, we assumed that dynamic traffic patterns are possible, that is both the source and destinations of the data flow and its rate can change over time. Hence, in order to achieve an energy efficient algorithm, we needed to monitor traffic patterns and act accordingly to change the routing paths.

As future work items, the followings can be investigated based on the studies presented in this thesis and considering network environments composed of nodes with heterogeneous power sources.

- Additional centralized and distributed algorithms can be obtained considering different requirements in WSNs and their performance can be evaluated.
- Routing algorithm frameworks for other types of networks (e.g., wireless ad hoc, wireless mesh) can be investigated.
- To further extend the performance of backbone-based algorithms, putting non-backbone nodes into sleep mode, instead of keeping them idle, can be investigated.
- Power-source-aware algorithms targeting different wireless technologies (e.g., Bluetooth [9], WiMAX [96]) can be designed.

Bibliography

[1] M. Tekkalmaz and I. Korpeoglu, "Power-source-aware Backbone Routing in Wireless Sensor Networks," in *IEEE International Conference on Communication Systems (ICCS'10)*, pp. 46–50, 2010.

[2] M. Tekkalmaz and I. Korpeoglu, "PSAR: Power-source-aware Routing in ZigBee Networks," *Wireless Networks*, vol. 18, no. 6, pp. 635–651, 2012.

[3] S. Roundy, D. Steingart, L. Frechette, P. Wright, and J. Rabaey, "Power Sources for Wireless Sensor Networks," in *Wireless Sensor Networks*, vol. 2920 of *Lecture Notes in Computer Science*, pp. 1–17, Springer, Berlin Heidelberg, 2004.

[4] Z. Watral and A. Michalski, "Selected Problems of Power Sources for Wireless Sensors Networks," *IEEE Instrumentation Measurement Magazine*, vol. 16, no. 1, pp. 37–43, 2013.

[5] C. Knight, J. Davidson, and S. Behrens, "Energy Options for Wireless Sensor Nodes," *Sensors*, vol. 8, no. 12, pp. 8037–8066, 2008.

[6] "The Network Simulator ns-2." `http://www.isi.edu/nsnam/ns/`, 2013. [Online; accessed 8-July-2013].

[7] X. Ma and W. Luo, "The Analysis of 6LoWPAN Technology," in *IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA'08)*, vol. 1, pp. 963–966, 2008.

[8] *ZigBee Specification*. ZigBee Standards Organization, December 2006.

[9] "Bluetooth." `http://www.bluetooth.com`, 2013. [Online; accessed 8-July-2013].

[10] *IEEE Standard for Local and Metropolitan Area Networks – Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE Computer Society LAN/MAN Standards Committee, June 2006.

[11] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[12] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, "Energy Conservation in Wireless Sensor Networks: A Survey," *Ad Hoc Networks*, vol. 7, no. 3, pp. 537–568, 2009.

[13] Y. Xu, J. Heidemann, and D. Estrin, "Geography-Informed Energy Conservation for Ad Hoc Routing," in *7th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'01)*, pp. 70–84, 2001.

[14] M. Zorzi and R. Rao, "Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance," *IEEE Transactions on Mobile Computing*, vol. 2, no. 4, pp. 349–365, 2003.

[15] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *Wireless Networks*, vol. 8, no. 5, pp. 481–494, 2002.

[16] A. Cerpa and D. Estrin, "ASCENT: Adaptive Self-Configuring Sensor Networks Topologies," *IEEE Transactions on Mobile Computing*, vol. 3, no. 3, pp. 272–285, 2004.

[17] P. B. Godfrey and D. Ratajczak, "Naps: Scalable, Robust Topology Management in Wireless Ad Hoc Networks," in *3rd IEEE International Symposium on Information Processing in Sensor Networks (IPSN'04)*, pp. 443–451, 2004.

[18] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong, "Approximate Data Collection in Sensor Networks Using Probabilistic Models," in *22nd IEEE International Conference on Data Engineering (ICDE'06)*, 2006.

[19] D. Tulone and S. Madden, "PAQ: Time Series Forecasting for Approximate Query Answering in Sensor Networks," in *Wireless Sensor Networks*, pp. 21–37, Springer, 2006.

[20] S. Goel and T. Imielinski, "Prediction-Based Monitoring in Sensor Networks: Taking Lessons from MPEG," *ACM SIGCOMM Computer Communication Review*, vol. 31, no. 5, pp. 82–98, 2001.

[21] A. Jain and E. Y. Chang, "Adaptive Sampling for Sensor Networks," in *Proceeedings of the 1st ACM International Workshop on Data Management for Sensor Networks, in conjunction with VLDB'04*, pp. 10–16, 2004.

[22] T. Kijewski-Correa, M. Haenggi, and P. Antsaklis, "Wireless Sensor Networks for Structural Health Monitoring: A Multi-Scale Approach," in *ASCE Structures Congress*, 2006.

[23] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-Driven Data Acquisition in Sensor Networks," in *13th International Conference on Very Large Databases*, pp. 588–599, VLDB Endowment, 2004.

[24] M. Gupta, M. S. Obaidat, and S. K. Dhurandher, "Chapter 13. Energy-Efficient Sensor Networks," in *Handbook of Green Information and Communication Systems*, pp. 353–369, Academic Press, 2013.

[25] S. Mahfoudh and P. Minet, "Survey of Energy Efficient Strategies in Wireless Ad Hoc and Sensor Networks," in *7th IEEE International Conference on Networking (ICN'08)*, pp. 1–7, 2008.

[26] R. C. Shah and J. M. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks," in *IEEE Wireless Communications and Networking Conference (WCNC'02)*, vol. 1, pp. 350–355, 2002.

[27] S.-M. Senouci and G. Pujolle, "Energy Efficient Routing in Wireless Ad Hoc Networks," in *IEEE International Conference on Communications*, vol. 7, pp. 4057–4061, 2004.

[28] H. Hassanein and J. Luo, "Reliable Energy Aware Routing in Wireless Sensor Networks," in *2nd IEEE Workshop on Dependability and Security in Sensor Networks and Systems (DSSNS'06)*, pp. 54–64, 2006.

[29] M. Cardei and D.-Z. Du, "Improving Wireless Sensor Network Lifetime Through Power Aware Organization," *Wireless Networks*, vol. 11, no. 3, pp. 333–340, 2005.

[30] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," *Wireless Networks*, vol. 12, no. 1, pp. 63–78, 2006.

[31] M. Cardei, J. Wu, and S. Yang, "Topology Control in Ad Hoc Wireless Networks with Hitch-hiking," in *1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, pp. 480–488, 2004.

[32] S. Lin, J. Zhang, G. Zhou, L. Gu, J. A. Stankovic, and T. He, "ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks," in *4th ACM International Conference on Embedded Networked Sensor Systems*, pp. 223–236, 2006.

[33] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *33rd IEEE Annual Hawaii International Conference on System Sciences*, p. 10, 2000.

[34] K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Maximum Lifetime Data Gathering and Aggregation in Wireless Sensor Networks," in *IEEE International Conference on Networking (ICN'02)*, vol. 2, pp. 685–696, 2002.

[35] A. Srinivas and E. Modiano, "Minimum Energy Disjoint Path Routing in Wireless Ad-hoc Networks," in *9th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'03)*, pp. 122–133, 2003.

[36] A. Nasipuri and S. R. Das, "On-demand Multipath Routing for Mobile Ad Hoc Networks," in *8th IEEE International Conference on Computer Communications and Networks*, pp. 64–70, 1999.

[37] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-Resilient, Energy-Efficient Multipath Routing in wireless Sensor Networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, pp. 11–25, Oct. 2001.

[38] S. Kwon and N. B. Shroff, "Energy-Efficient Interference-Based Routing for Multi-Hop Wireless Networks," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, 2005.

[39] N. Shrestha, *Reception-Awareness for Energy Conservation in Ad Hoc Networks*. PhD thesis, 2007.

[40] M. Ye, C. Li, G. Chen, and J. Wu, "EECS: An Energy Efficient Clustering Scheme in Wireless Sensor Networks," in *24th International Performance, Computing, and Communications Conference (IPCCC'05)*, pp. 535–540, 2005.

[41] O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[42] A. Chamam and S. Pierre, "A Distributed Energy-Efficient Clustering Protocol for Wireless Sensor Networks," *Computers & Electrical Engineering*, vol. 36, no. 2, pp. 303–312, 2010.

[43] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: A Survey," *Wireless Communications*, vol. 11, no. 6, pp. 6–28, 2004.

[44] D. Braginsky and D. Estrin, "Rumor Routing Algorthim for Sensor Networks," in *1st ACM International Workshop on Wireless Sensor Networks and Applications*, pp. 22–31, 2002.

[45] C. Schurgers and M. B. Srivastava, "Energy Efficient Routing in Wireless Sensor Networks," in *IEEE Military Communications Conference (MILCOM'01), Communications for Network-Centric Operations: Creating the Information Force*, vol. 1, pp. 357–361, 2001.

[46] F. Ye, A. Chen, S. Lu, and L. Zhang, "A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks," in *10th IEEE International Conference on Computer Communications and Networks*, pp. 304–309, 2001.

[47] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Large-Scale Wireless Sensor Networks," in *8th ACM Annual International Conference on Mobile Computing and Networking*, pp. 148–159, 2002.

[48] V. Rodoplu and T. H. Meng, "Minimum Energy Mobile Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1333–1344, 1999.

[49] Q. Li, J. Aslam, and D. Rus, "Hierarchical Power-Aware Routing in Sensor Networks," in *DIMACS Workshop on Pervasive Networking*, 2001.

[50] Y. Yu, R. Govindan, and D. Estrin, "Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," Tech. Rep. UCLA-CSD TR-010023, UCLA Computer Science Department, May 2001.

[51] I. Stojmenovic and X. Lin, "GEDIR: Loop-free Location Based Routing in Wireless Networks," in *IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 1025–1028, 1999.

[52] D. Simplot-Ryl, I. Stojmenovic, and J. Wu, *Handbook of Sensor Networks*, ch. 11. Energy-Efficient Backbone Construction, Broadcasting, and Area Coverage in Sensor Networks, pp. 343–380. Wiley Series on Parallel and Distributed Computing, Wiley, 2005.

[53] B. Das, R. Sivakumar, and V. Bharghavan, "Routing in Ad Hoc Networks Using a Spine," *6th International Conference on Computer Communications and Networks (ICCCN'97)*, p. 34, 1997.

[54] A. Boukerche, X. Cheng, and J. Linus, "Energy-Aware Data-Centric Routing in Microsensor Networks," in *6th ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWIM '03)*, pp. 42–49, 2003.

[55] P. Santi and J. Simon, "Silence is Golden with High Probability: Maintaining a Connected Backbone in Wireless Sensor Networks," in *Wireless Sensor Networks*, vol. 2920 of *Lecture Notes in Computer Science*, pp. 106–121, Springer Berlin Heidelberg, 2004.

[56] F. Dai and J. Wu, "On Constructing k-connected k-dominating Set in Wireless Ad Hoc and Sensor Networks," *Journal of Parallel and Distributed Computing*, vol. 66, no. 7, pp. 947–958, 2006.

[57] A. Kashyap, S. Khuller, and M. A. Shayman, "Relay Placement for Higher Order Connectivity in Wireless Sensor Networks," in *25th IEEE International Conference on Computer Communications (INFOCOM'06)*, pp. 1–12, 2006.

[58] K. Mikhaylov and J. Tervonen, "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, 2011.

[59] W.-G. Seah, Z. A. Eu, and H. Tan, "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, 2009.

[60] S. Chalasani and J. Conrad, "A Survey of Energy Harvesting Sources for Embedded Systems," in *IEEE SoutheastCon*, pp. 442–447, 2008.

[61] S. Sudevalayam and P. Kulkarni, "Energy Harvesting Sensor Nodes: Survey and Implications," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 3, pp. 443–461, 2011.

[62] J. Gilbert and F. Balouchi, "Comparison of Energy Harvesting Systems for Wireless Sensor Networks," *International Journal of Automation and Computing*, vol. 5, no. 4, pp. 334–347, 2008.

[63] C. Park and P. Chou, "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, 2006.

[64] Y. Tan and S. Panda, "Energy Harvesting from Hybrid Indoor Ambient Light and Thermal Energy Sources for Enhanced Performance of Wireless Sensor Nodes," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 9, pp. 4424–4435, 2011.

[65] A. Hande, T. Polk, W. Walker, and D. Bhatia, "Indoor Solar Energy Harvesting for Sensor Network Router Nodes," *Microprocessors and Microsystems*, vol. 31, no. 6, pp. 420–432, 2007. Special Issue on Sensor Systems.

[66] S. Lee, B. D. Youn, and B. C. Jung, "Robust Segment-Type Energy Harvester and Its Application to a Wireless Sensor," *Smart Materials and Structures*, vol. 18, no. 9, 2009.

[67] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting Heterogeneity in Sensor Networks," in *24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, vol. 2, pp. 878–890, 2005.

[68] Y. Ma, S. Dala, M. Alwan, and J. Aylor, "ROP: A Resource Oriented Protocol for Heterogeneous Sensor Networks," in *Virginia Tech Symposium on Wireless Personal Communications*, 2003.

[69] A. Kansal, J. Hsu, M. Srivastava, and V. Raghunathan, "Harvesting Aware Power Management for Sensor Networks," in *43rd Annual Design Automation Conference (DAC'06)*, pp. 651–656, ACM, 2006.

[70] T. Voigt, H. Ritter, and J. Schiller, "Utilizing Solar Power in Wireless Sensor Networks," in *28th IEEE Annual International Conference on Local Computer Networks (LCN'03)*, pp. 416–422, 2003.

[71] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards," *Computer Communications*, vol. 30, pp. 1655–1695, May 2007.

[72] P. Suarez, C.-G. Renmarker, A. Dunkels, and T. Voigt, "Increasing ZigBee Network Lifetime with X-MAC," in *Workshop on Real-World Wireless Sensor Networks (REALWSN'08)*, p. 5, 2008.

[73] D.-H. Cho, J.-H. Song, and K.-J. Han, "An Adaptive Energy Saving Mechanism for the IEEE 802.15.4 LR-WPAN," *Lecture Notes in Computer Science*, vol. 4138/2006, pp. 38–46, 2006.

[74] J.-A. Kim, Y.-H. Jean, and H.-S. Park, "Period Assignment and Optimal Scheduling for Zigbee-Based Sensor Networks," in *International Joint Conference SICE-ICASE*, pp. 1030–1035, Oct. 2006.

[75] Y. Li, W. Ye, and J. Heidemann, "Energy and Latency Control in Low Duty Cycle MAC Protocols," in *IEEE Wireless Communications and Networking Conference*, vol. 2, pp. 676–682, March 2005.

[76] D. Puccinelli, E. Sifakis, and M. Haenggi, "A Cross-Layer Approach to Energy Balancing in Wireless Sensor Networks," *Lecture Notes in Control and Information Sciences*, vol. 331/2006, pp. 309–324, 2006.

[77] N. Boughanmi and Y. Song, "A New Routing Metric for Satisfying Both Energy and Delay Constraints in Wireless Sensor Networks," *Journal of Signal Processing Systems*, vol. 51, pp. 137–143, May 2008.

[78] R. Peng, S. Mao-heng, and Z. You-min, "ZigBee Routing Selection Strategy Based on Data Services and Energy-Balanced ZigBee Routing," in *IEEE Asia-Pacific Conference on Services Computing (APSCC '06)*, pp. 400–404, Dec. 2006.

[79] H.-C. Huang, Y.-M. Huang, and J.-W. Ding, "An Implementation of Battery-Aware Wireless Sensor Network Using ZigBee for Multimedia Service," in *International Conference on Consumer Electronics*, pp. 369–370, Jan. 2006.

[80] N. Shillingford, D. C. Salyers, C. Poellabauer, and A. Striegel, "DETOUR: Delay- and Energy-Aware Multi-Path Routing in Wireless Ad Hoc Networks," in *International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'07)*, pp. 1–8, Aug. 2007.

[81] J. Ma, M. Gao, Q. Zhang, and L. M. Ni, "Energy-Efficient Localized Topology Control Algorithms in IEEE 802.15.4-Based Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 5, pp. 711–720, 2007.

[82] F. Cuomo, S. D. Luna, U. Monaco, and T. Melodia, "Routing in ZigBee Benefits from Exploiting the IEEE 802.15.4 Association Tree," in *IEEE International Conference on Communications (ICC'07)*, pp. 3271–3276, Jun. 2007.

[83] B. Nefzi and Y.-Q. Song, "Performance Analysis and Improvement of ZigBee Routing Protocol," in *7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*, vol. 7, 2007.

[84] J. Sun, Z. Wang, H. Wang, and X. Zhang, "Research on Routing Protocols Based on ZigBee Network," in *13th International Conference on International Information Hiding and Multimedia Signal Processing (IIH-MSP'07)*, pp. 639–642, 2007.

[85] J. Y. Ha, H. S. Park, S. Choi, and W. H. Kwon, "EHRP: Enhanced Hierarchical Routing Protocol for ZigBee Mesh Networks," *IEEE Communications Letters*, vol. 11, pp. 1028–1030, Dec. 2007.

[86] T. Kim, D. Kim, N. P. S. eun Yoo, and T. S. Lopez, "Shortcut Tree Routing in ZigBee Networks," in *2nd International Symposium on Wireless Pervasive Computing (ISWPC '07)*, Feb. 2007.

[87] R. G. Gallager, P. A. Humblet, and P. M. Spira, "A Distributed Algorithm for Minimum-Weight Spanning Trees," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 5, no. 1, pp. 66–77, 1983.

[88] K. M. Chandy and J. Misra, "Distributed Computation on Graphs: Shortest Path Algorithms," *Communications of the ACM*, vol. 25, pp. 833–837, Nov. 1982.

[89] T. Camilo, J. S. Silva, A. Rodrigues, and F. Boavida, "GENSEN: A Topology Generator For Real Wireless Sensor Networks Deployment," in *5th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, 2007.

[90] M. Stemm and R. H. Katz, "Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices," *IEICE Transactions on Communications*, vol. E80-B, pp. 1125–1131, August 1997.

[91] "DOT, Graph Description Language." `http://en.wikipedia.org/wiki/DOT_(graph_description_language)`, 2013. [Online; accessed 8-July-2013].

[92] "Graphviz - Graph Visualization Software." `http://www.graphviz.org/`, 2013. [Online; accessed 8-July-2013].

[93] A. Prince-Pike, *Power Characterisation of a Zigbee Wireless Network in a Real Time Monitoring Application*. PhD thesis, AUT University, 2009.

[94] G. Ding, Z. Sahinoglu, B. Bhargava, P. Orlik, and J. Zhang, "Reliable Broadcast in ZigBee Networks," in *2nd Annual IEEE Communications Society Conference on Sensor and AdHoc Communications and Networks*, pp. 510–520, Sept. 2005.

[95] M.-S. Pan and Y.-C. Tseng, "The Orphan Problem in Zigbee-Based Wireless Sensor Networks," in *10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM'07)*, pp. 95–98, 2007.

[96] J. G. Andrews, A. Ghosh, and R. Muhamed, *Fundamentals of WiMAX: Understanding Broadband Wireless Networking*. Pearson Education, 2007.