UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

DATA GATHERING TECHNIQUES ON WIRELESS SENSOR NETWORKS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

ARAVIND MOHANOOR

Norman, Oklahoma

2008

DATA GATHERING TECHNIQUES ON WIRELESS SENSOR NETWORKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

_____

Dr. Sridhar Radhakrishnan, Chair

_____

Dr. S. Lakshmivarahan

_____

Dr. Sudarshan Dhall

_____

Dr. John Antonio

_____

Dr. Marilyn Breen

Acknowledgments

I wish to express my sincere gratitude to my dissertation advisor Dr. Sridhar Radhakrishnan for his guidance and inspiration without which this work would have been impossible. His constant enthusiasm and support was a big factor in my persisting through the many peaks and troughs which are commonplace during graduate studies. It is fair to say that I learnt much from him about research as well as life.

I would like to express my gratitude to my committee members Dr. S. Lakshmivarahan, Dr. Sudarshan Dhall, Dr. John Antonio and Dr. Marilyn Breen for serving on my dissertation committee. I would also like to thank Dr. Venkatesh Sarangan of Oklahoma State University for many interesting discussions about research and for helping me introduce rigor to my approach to research writing.

I would like to express many thanks to Dr. Henry Neeman for a very stimulating and enjoyable experience as a Teaching Assistant for CS 1313. I would also like to express thanks to Dr. Shivakumar Raman for the financial support during the last year of my doctoral studies.

I am very grateful for the support and well wishes I received from all my friends in Norman and elsewhere. It would be hard to list all of them, but special thanks to Muhammad Javed, whose help, encouragement and counsel helped me and I am sure, many others who know him. I personally do not know of anyone more willing to help others. I would also like to thank my friends Shankar Banik, Jonghyun Kim, Tao Zheng, Ta Chun Lin, Yuh Rong Chen, Yu Hsin Li, Celi Sun, Pedro Diaz Gomez, Jayashree, Bharath Ramanujam, Waleed Numay, Casmir Agbaraji, Igor Wasinski, Ramnathan

Muthuraman, Naren Pappu, Sujith Jagini, Sudhir Vallamkondu, Prashant Kakani, Nicolas Lundgaard, Tom Hughes, Shirish Deshpande, Ganesh Krishnamurthy, Hira Shrestha, Zhaowen, Kyle Abbott, Moshe Gutman, Clay Packard, Darren White, Jason Black and so many others who have been a part of my life during my years in graduate school.

I am also thankful to the wonderful and very supportive staff members in Computer Science, especially Barbara Bledsoe, Sandy Johnson, Chyrl Yerdon, Jim Summers and all others who work/worked very hard to keep the Computer Science department running smoothly and for the kind words and encouragement when students come for help and advice.

Last but not least, I would like to thank all my family members for their constant support and encouragement, especially my parents, M. C. Satagopan and Vijayalakshmi Satagopan. My continued progress towards my doctoral degree would not have been possible without the moral support from my sister, Aarthi Narasimhan, my brother-in-law P.L.Narasimhan and my cousin Harish.

Table of Contents

List of figures

List of tables

Abstract

The nearly exponential growth of the performance/price and performance/size ratios of computers has given rise to the development of inexpensive, miniaturized systems with wireless and sensing capabilities. Such wireless sensors are able to produce a wealth of information about our personal environment, in agricultural and industrial monitoring, and many other scenarios. Each sensor due to its miniature nature has severe resource constraints in terms of processing power, storage space, battery capacity and bandwidth of radio. Our goal in this research is to maximize the extraction of information out of the sensor network by efficient resource utilization.

The typical deployment scenarios of these wireless sensors require that the individual computers (sensors) communicate with each other over multiple hops. A natural representation of this network is as a communication graph where each sensor is represented as a node in the graph and each wireless link between two sensors is represented as an edge. With this representation, it is clear that there are many choices of paths which may be used for communicating the data. Hence a good choice of the communication path is an important aspect of optimal resource utilization in such networks.

Solving these problems lead to the following graph and path problems:

a) The lifetime of a sensor network is defined as the number of packets which can be transmitted (i.e. the amount of data collected) before two sensors are unable to communicate with each other due to depletion of battery power along the

intermediate nodes connecting them. To collect the largest amount of data possible, using a multi-metric shortest path called as the shortest widest path, as well as close derivatives, is crucial.

b) Optimal utilization of the available bandwidth and thus improving the perceived throughput is beneficial in multiple ways. First of all, better throughput is a desirable end goal in itself. Also it can be shown that better throughput requires the sensors to be transmitting for smaller durations of time and thus also saves energy. In order to optimize the throughput in a multi-hop wireless network, we must focus on a strategy of finding paths which are 'interference aware'. Unlike earlier work which primarily concentrated on link and node scheduling for this problem, our path scheduling approach produces superior throughput at very reasonable computational costs.

c) As we have seen, path problems on sensor networks play an important role in good network resource utilization. Typically we are also interested in implementing these path problems in a distributed manner. When we develop distributed algorithms for wireless sensor networks, we must be respectful of the typical packet size in a wireless sensor network, which is currently of the order of tens of bytes. This puts an impediment on developing distributed path algorithms which transmit large sized messages. Exploiting earlier work on low bit complexity distributed algorithms provides a way around this impediment.

d) Another important technique for distributedly computing paths is to use network decomposition strategies. Our work presents a network decomposition strategy well

suited for wireless sensor networks as it provides energy aware execution of distributed algorithms without sacrificing scalability.

e) As individual sensors become more powerful and start carrying multiple radios, the problem of activating the radios in an energy aware fashion will turn out to be critical. Our work on radio activation provides additional insight into this problem and shows that the essential question is one of finding and creating such topologies where high power radios form high degree clusters so that the number of nodes connected per high power radio activated is fairly high.

f) We can use these results to provide qualitative specifications for the data being collected. Our work on Quality of Information (QoI) discusses how we can define attributes for information quality so as to perform data collection with good resource utilization.

Chapter One

1 Introduction

The increasing availability of cheap, low power, embedded processors with radios, sensors and actuators is enabling the use of wireless communication and computing for interacting with the physical world in a plethora of applications such as security and surveillance, habitat monitoring, medical monitoring and others [30].[1] A wireless network comprising such devices, also called a wireless sensor network (WSN), is being deployed in a wide variety of situations ranging from experimental deployments with yet to be discovered tangible benefits [89, 86, 54] to the ROI-enhancing applications [13, 69, 46] with directly measurable commercial benefits[2]. Figure 1.1 demonstrates an example of a wireless sensor network used for volcano monitoring.

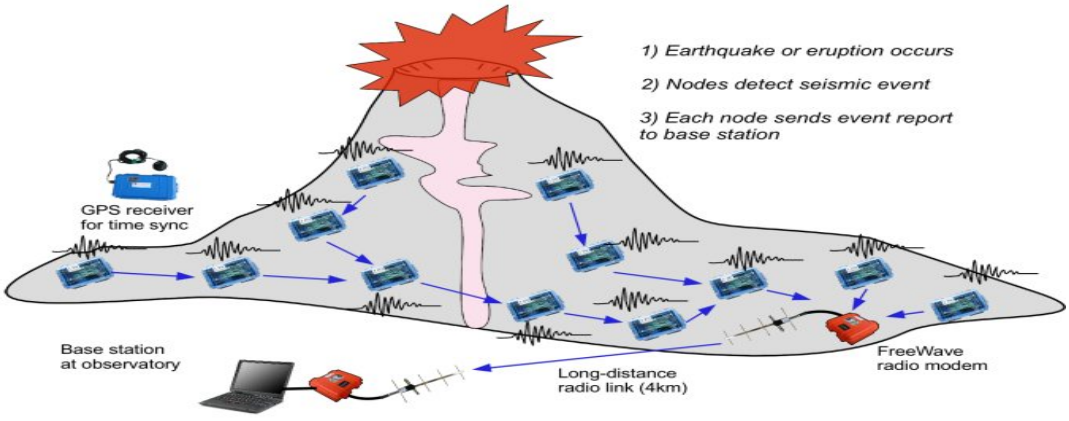


Figure 1.1 Wireless sensor network used for volcano monitoring[3]

[1] Refer to Appendix A for some discussions about the physical capabilities of sensor devices
[2] Refer to Appendix B for a list of some real world wireless sensor networks and their applications
[3] http://fiji.eecs.harvard.edu/Volcano

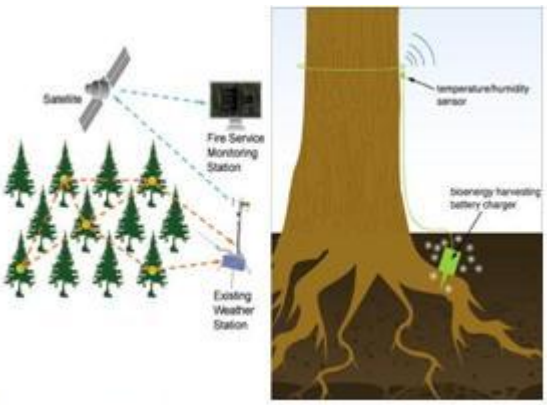


Figure 1.2 Forest fire detection using wireless sensor network[4]

Similarly, figures 1.2 and 1.3 demonstrate how wireless sensor networks are being used in forest fire detection and for precision agriculture. Precision agriculture is the practice of fine grained monitoring and management of crops, which is enabled by the use of wireless sensors which can collect and report the requisite data.

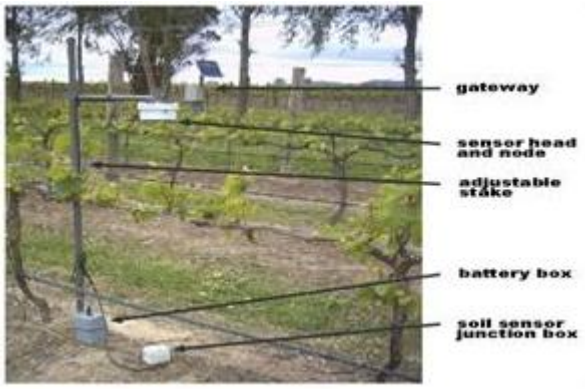


Figure 1.3 Sensor network for precision agriculture[5]

The wide variety of application scenarios proposed for sensor networks is matched by an equally large number of constraints and requirements when translated to mathematical models and graph formalizations, and one could say that this is still a

[4] http://web.mit.edu/newsoffice/2008/trees-0923.html
[5] http://blog.xbow.com/xblog/2007/10/worlds-largest-.html

nascent period of discovery in terms of graph algorithms and network protocols for wireless sensor networks. While the exhaustive research on wired as well as wireless networks could be adopted according to specific requirements for wireless sensor networks, the extremely resource constrained nature of these networks indeed pose a different set of questions and more importantly, fundamentally challenge the layered network model (the OSI seven layer model) in various ways. It is not uncommon to find popular sensor network protocols which tune all lower layers directly to suit the application level requirements [28, 33]. This is in part due to the autonomous nature of typical sensor network deployments, which often makes it unnecessary to consider adversarial behavior on the part of network participants.

While initial research has focused on developing protocols which may span multiple layers of the seven layer model, there is also a case to be made for finding out the optimal ways in which individual resources in the network such as battery power, bandwidth, storage and packet size could be used. The knowledge of optimal utilization of individual resources would help identify ways of data collection such that we can simultaneously optimize along multiple resource constraints. This helps provide a data-specification abstraction to the end user which is both flexible and resource friendly.

To provide an understanding of the kinds of resource constraints we encounter in sensor networks, we consider a sensor with an Intel StrongArm processor which consumes 400mW of power in normal mode, 100mW in idle mode and 50μW in sleep mode. The energy required to transmit a single bit by such a sensor is about 1μJ. These sensors are powered by batteries with typical voltage rating 1.5V and has a capacity of 400mAh. Since a watt-hour is 3600J, this means the number of bits which can be

transmitted using a single battery is about 270 MBits. Additionally, the energy required for a single computation is about 1 nJ per instruction, which means the ratio of communication energy cost to computation energy cost is nearly 1000! Energy is a vital resource and communication costs can be overwhelming. So we can see that the energy constraints (as well as other resource constraints) require that we make prudent use of the available resources. Besides, the typical deployment scenario of a wireless network calls for unsupervised operation and it is possible that once the battery of a sensor dies, the sensor is simply discarded. It is expected that the redundancy of the coverage coupled with the inexpensive nature of the sensors would make this a possibility. In such a scenario, it is crucial that the network is operated in a resource friendly manner.

1.1 Energy as network resource – extending the network's lifetime

Energy management in wireless networks is of paramount importance due to the limited energy availability in the wireless devices. Since wireless communication consumes a significant amount of energy, it is important to minimize the energy costs for communication as much as possible by practicing energy aware routing strategies. Such routing strategies can increase the network lifetime. In the second chapter, we focus on developing routing strategies for multiple hop wireless networks where all the nodes are powered by a battery or other external power sources such as solar energy. Usually network lifetime is quantified through the number of packets that can be transferred in the network before the source and destination get disconnected from each other [48, 65]. A suitable energy-aware routing strategy for wireless networks is to use those wireless nodes with high energy levels and avoid those with low energy levels.

In developing energy aware routing techniques, wireless networks are modeled as graphs wherein, the vertex represents a wireless device and an edge between two vertices indicates that they are in direct communication range of each other. The weight on a vertex indicates the residual energy available at that wireless node and the weight on an edge (u, v) represents the amount of energy required by node u (resp. v) to transmit one unit of data to node v (resp. u). The *residual energy of a path* is defined as the minimum energy level of any node in the path. The max-min routing paradigm suggested in the literature [1, 48, 81] aims to maximize the network lifetime by finding the path where the residual energy is the maximum and forwards packets through this path termed as the *maximum residual energy path*. The *energy consumed along a path* (or simply the *energy of a path*) is the sum of the weights on the edges along the path.

Notable routing strategies which utilize the concept of the residual energy (either directly or indirectly) proposed so far include MMBCR [81], MRPC [1] and max-min zP_{min} [48]. These research works also caution that merely using the residual energy strategy may lead to higher energy consumption in the network, since the energy consumed along the data forwarding path is not taken into consideration. In our work, we show a theoretical justification of this notion by developing a relationship between the residual energy of the nodes along a path and the total energy of the path. A good energy-aware routing technique should balance two different goals: choosing a path with maximal residual energy and choosing a path with minimal energy consumption.

We note that the residual energy along a path is a concave metric[6], whereas the energy consumed along a path is an additive metric.

We present three polynomial time combinatorial techniques which can provide a good balance between metrics 1 and 2. The first technique, called the Shortest Widest Path, first maximizes the concave metric (the residual energy of a path) and then minimizes the additive metric (energy consumed along a path). The second technique, which we call the Shortest Width Constrained Path, finds paths with a suitably high residual energy (which may not be the maximum), and then minimizes the energy consumed along such a path. Lastly, our third approach (Shortest Fixed Width path) is similar to the second approach in the sense that it finds a minimum energy path among the paths that have a high residual energy. However, unlike the shortest width constrained path, it does not change the residual energy with each route calculation; the residual energy level is changed only when it becomes infeasible to find paths between the source and the destination at the current residual energy level. Our simulation studies show that the performance of the proposed techniques is superior to that of the best known routing approach proposed in the literature (Park and Sahni [65]). We also discuss the performance of our proposed algorithms in a distributed setting. Even if the nodes lack an accurate global knowledge of the instantaneous node energy levels, we find that the two phased routing techniques perform reasonably well. We also find that the proposed distributed techniques outperform the distributed implementation (that we have developed) of the algorithm proposed by Park and Sahni [65].

[6] For definitions of concave and additive metrics, see Wang and Crowcroft [84].

1.2 Bandwidth as network resource – improving wireless throughput

The throughput observed in a single path in a multiple hop wireless network can be no more than a third of the single hop bandwidth under the standard radio model [48]. There are some applications where one might require a higher end-to-end throughput than that available through the use of a single path. As an example, the volcano monitoring example mentioned in Section 1.1 is a typical application which requires frequent monitoring and hence fairly high data rates. In contrast, improving the network throughput may not be as important in other sensor network applications where data is generated much less frequently. In any case, a better utilization of the available bandwidth could lead to less idling and hence improved network lifetime. The traditional way to achieve higher throughputs in wired networks is to use multiple paths in parallel so as to improve the aggregate bandwidth. While a similar approach can be adopted for wireless sensor networks too, wireless networks suffer from the additional challenge that in most network topologies the discovered paths may lie close to each other. Consequently, packet transmissions in one path may interfere with transmissions taking place in other paths in the path set leading to a significant reduction in the overall throughput [87].

This reduction in throughput in wireless networks can be avoided by appropriate path selection combined with careful packet transmission scheduling. It has been noted in the literature [48] that the maximum possible throughput equaling the single hop bandwidth can be achieved by using three non-interfering paths. Hence having multiple paths which do not interfere with each other is ideal. However, we show in chapter 3

that this problem is the same as the problem of finding chordless cycle containing a pair of vertices in a graph, which is actually NP-Complete [10]. We then turn our attention towards finding path sets in static wireless networks which would provide the same level of aggregate throughput as non interfering paths while at the same time permitting interfering links.

Our contributions are: (a) we demonstrate that it is possible for a set of paths between source *s* and destination *t* with some interference between them to provide high aggregate throughputs provided the interfering edges among the paths follow certain favorable patterns; we present a combinatorial approach for finding such paths in a wireless network. (b) we extend our approach to scenarios involving multiple *s-t* pairs and show that the proposed approach can improve the throughput in such scenarios too (c) our combinatorial approach can also provide a straightforward mechanism for scheduling the transmissions at various links and finally, (d) the computation of the transmission schedule is shown to be amenable to a distributed implementation under the proposed approach.

1.3 Message size constraints

The size of messages exchanged in a sensor network is also an important constraint. Besides the fact that larger size messages require larger energy for transmission, the usual fragmentation of packets would require multiple transmissions for successful packet transmission. This affects both the latency of the message exchange as well as leads to a potentially larger chance for transmission loss and thus larger retransmission cost over lossy wireless links. Distributed algorithms, which are primarily based on

message transfers, cover the entire gamut from algorithms based on single bit messages [44] to algorithms which may be fairly oblivious to the size of the messages being transferred [70]. However, a sensor network provides an environment where distributed algorithms with low message-bit complexity can be more easily implemented.

We focus on a specific problem – namely, increasing the wireless throughput for multiple source destination pairs in a wireless network. The ideas from chapter 3 are adapted for a distributed implementation. The fourth chapter of this dissertation provides distributed techniques for the throughput improvement problem which have low message-bit complexity and which use only single messages even under small packet sizes.

1.4 Improving resource utilization using multiple radios

While the first generation of wireless sensors had limited processing and storage capabilities, advances in technology, in combination with increased application demands have resulted in more powerful second generation sensor nodes. These nodes possess relatively higher processing and storage capabilities achieved through the use of powerful CPUs, and large memories [34], [57]. These nodes are also capable of operating multiple radios simultaneously, each with a different power, range and bandwidth rating. Though such multi-radio sensors are currently used as gateways or cluster-heads in sensor networks, technological advancement may soon equip even the commonly used sensor nodes with multiple radios.

While the capabilities of sensor nodes have increased along several fronts, they will continue to be powered by batteries. Consequently, energy conserving mechanisms are

of paramount importance even in next generation wireless sensor networks. The radios in a multi-radio sensor node may differ not only in terms of their communication capabilities but also in terms of energy efficiency and usage. High bandwidth, long-range radios usually possess higher energy efficiency, in terms of energy expended per bit transmitted, than low bandwidth, short-range radios [80]. However, high bandwidth radios also consume more power when idling than low bandwidth radios. Therefore, activating several high bandwidth radios when there is not a lot of data to be transmitted may result in considerable energy wastage. On the other hand, due to their greater reach, long-range radios can reduce the network diameter; consequently, the latency involved in delivering sensory data to a prescribed destination will decrease with the use of long range radios. Several of them may need to be activated when the application demands smaller data delivery latency. Thus the issue of radio activation is closely tied to the requirements of the application and is the focus of chapter 5 in this dissertation.

Earlier research on multi-radio systems assume that the network remains connected even when all the sensor nodes activate only their lowest power radio. However, in a general setting, such a requirement on the connectivity cannot be guaranteed. Radios with higher power and longer range may have to be activated even to make the network connected. In chapter 5, we focus on energy efficient radio activation in a sensor network where each node has $K > 1$ radios. The radios r_1, r_2,…, r_k in a node are such that the one hop reachability distance (resp. energy expended) of (resp. by) radio r_i is greater than that of r_j, $1 \leq j < i \leq k$. Given such a network, the problem of energy efficient radio activation is to minimize the total energy spent by the active radios

across all nodes in order to maintain a connected network. We make several contributions: (1) We show that the problem of energy efficient radio activation is NP-Complete. (2) We propose four different polynomial time approximation methodologies for solving this problem in networks with $K = 2$. The first two methodologies employ a series of non-trivial reductions to leverage on existing approximation solutions for other known NP-Complete problems. The third methodology is based on the minimum spanning tree algorithm. The fourth methodology is a greedy algorithm that is proposed afresh. (3) We extend these solutions to the general case of $K > 2$ radios as well. (4) Our analytical and experimental studies of the four solutions reveal that the greedy algorithm and the minimum spanning tree solution have the best *worst case* performance while the greedy algorithm has the best *average case* performance.

1.5 Lifetime aware network decomposition for executing distributed algorithms

As noted earlier, energy aware routing strategies help in extending the lifetime of a wireless network. This is very important for sensor networks where the energy is an important nonreplenishable resource. A suitable energy-aware routing strategy for wireless networks is to use those wireless nodes with high energy levels and avoid those with low energy levels. The routing strategies on sensor network involve the following general steps, a) find routes; b) perform routing; c) update network values and perform step a).

Consider a centralized algorithm wherein a single node (call it *central* node) keeps track of the topology information. The central node will determine the routes (step a)

by executing a local algorithm. When a source node requires a message to be routed to the destination, it sends a request to the central node which will provide the entire route to the destination. After the receipt of the information from the central node, the source node can perform routing (step b). Assuming that the source node follows the exact route provided by the central node, the central node can determine the energy changes of the intermediate node (without the intermediate nodes explicitly informing the central node) and re-compute the routes locally for the next route request. In addition to the energy consumed when packets are routed along the route path, energy is also consumed at intermediate nodes along the path from source to central site and vice-versa for route request and response. This straight-forward algorithm has all the weaknesses of any centralized algorithm such as lack of fault-tolerance and problems associated with hot spots created by request/response information travelling to and from the central node. In fact with repeated route requests it is easy to observe that the neighbors of the central site may quickly lose energy thereby making the central node unreachable and consequently decreasing lifetime. One could choose a new central node and use a simple distributed algorithm such as the distributed depth first search [70] to learn the topology of the network including the node and link information. Yet another weakness of the centralized algorithm is that for large resource limited sensor networks a single central node may have neither the space capacity to store the entire network nor the computation power to compute the paths in a short period of time, or even enough energy to perform the computation as it is nearing the end of its battery life.

Given a distributed system consisting of computational nodes, a distributed algorithm solves a particular problem of interest by exchanging messages among the nodes. In the distributed system each node knows its neighbors by their unique identities and the total number of nodes in the distributed system. A distributed algorithm is evaluated based on the total number of messages exchanged (*message-complexity*) and the time-taken for the completion of the distributed algorithm (*time-complexity*). Depending on the problem to be solved the distributed algorithm must be rerun after a node or link update either on the entire network or a portion of the network. Distributed algorithms are scalable as they do not require a single node to keep track of the entire topology information. The fundamental weakness of the distributed algorithms for sensor networks stems from the fact that after step b) of the routing strategy is completed, the intermediate nodes have new energy levels and now the distributed algorithm to determine routing paths (step a) has to be re-executed. That is after each route request is complete the distributed algorithm is re-run and thereby the message complexity is overwhelmed by the number of route requests that have been completed.

From the above discussion it is clear that the centralized algorithm is message efficient, but ineffective on lifetime as a result of hot spots and other issues relating to a centralized site. The distributed algorithm addresses the deficiencies of the centralized algorithm but is ineffective in terms of lifetime due to large number of messages required to recompute the routing paths after a completion of a route request. We propose a network decomposition approach that will combine both the centralized and distributed approaches described above by decomposing the network into smaller

networks (referred to as *clusters*). The centralized algorithm is executed on each cluster and the distributed algorithm is executed on the central nodes (referred to as *cluster head*) of each cluster.

Network decomposition is akin to divide-and-conquer approaches to problem solving wherein, a larger problem is broken into smaller sub-problems and solutions of the smaller sub-problem are combined to arrive at the solution to the larger problem. Network decomposition has been effectively used to solve many problems in sequential, parallel, and distributed environments. Network decomposition techniques have shown to reduce the message complexity of distributed algorithms by (i) decomposing the network into a set of connected components, (ii) running a pseudo-distributed algorithm on each connected component (we will call the connected component a *cluster*), and (iii) solving the optimization problem by executing a distributed algorithm involving cluster heads of each cluster. Note that a node that is along the path connecting two cluster heads will only forward messages.

Using network decomposition approaches one can alleviate the problems resulting in having central site. Updates in each cluster are sent to its cluster head. The cluster heads perform a local computation using the topology information as in the case of centralized algorithm. The cluster heads communicate using “meta” data and execute a distributed algorithm to solve the problem at hand. Conceptually since the number of cluster heads is smaller and fewer nodes will participate in the distributed algorithm the message complexity could be smaller. The above idea has been used to solve many distributed algorithms effectively [4].

Awerbuch and Peleg [5, 6, 7, 8] have published a series of seminal works in the area of distributed algorithms that use the concept of network decomposition. These works and the work by Linial [51] and Naor and Stockmeyer [64] exploit the concept of "locality" in distributed computations. The concept of locality is that certain functions when locally computed do not affect the global solution. For certain problem the solutions of the local computation can be cleverly combined to obtain global solution. Considering network problems on networks that have been decomposed, certain coloring problem instances can be solved efficiently for the entire network by cleverly stitching together solutions for each cluster.

In chapter 6, we introduce the widest path problem and its application to improving network lifetime. We present an algorithm to perform widest path routing (or called the maximum residual energy path routing) given a set of clusters. Ideal network decomposition which is suitable for improving network lifetime is then described and a decomposition algorithm for such a lifetime aware decomposition is also presented.

1.6 Quality of information techniques for knowledge centric sensor networks

A primary necessity for sensor network deployments is to be able to collect data about environmental (and other) phenomena under observation and transform it into useful, actionable knowledge. However, sensor networks due to their resource constrained nature have some key differences from general communication networks (such as the World Wide Web, and corporate intranets). The differences are fundamental, and hence we use the term message centric networks to refer to big, powerful networks such as the

WWW and corporate intranets, and knowledge centric networks to refer to sensor networks which usually lie on the other end of the spectrum in terms of scale.

Unit of atomicity – The unit of atomicity can be defined as the 'indivisible' unit of information which still retains semantics. In a knowledge centric network, where combining information is encouraged and loss of information is tolerated, the unit of information is the aggregate knowledge rather than the individual message.

Resource assumptions – the simple act of resending a message is commonplace (and even vital for everyday tasks such as browsing the internet) on a message centric network, where we can make assumptions of virtually unlimited resources. Resending a single message would require careful planning on a resource constrained sensor network, where minimal assumptions are made about the availability of resources.

Data gathering - Nearly all messages generated can be and are usually stored or collected in a message centric network, while that is neither a requirement nor a prudent choice on a knowledge centric and resource poor sensor network.

Data dispersal – Data dispersal refers to the replication of the same data and its dispersal over multiple media and devices (such as backing up important files on to a USB drive, a backup disk, and online storage). In a message centric network such as the internet, data dispersal is common and quite useful. Such data dispersal would be costly on a typical sensor network.

Search techniques – this is perhaps a vital difference and a key motivation for the QoI strategy. Any data collected, in order to be made useful, needs to be analyzed and processed. This would usually require doing a search over the data at some point in

time. In a message centric network, due in large part to the resource rich nature, the exponential growth of data is tolerated and search techniques evolve to deal with the rate of data generation. We call this the “store and search” strategy. In a fundamentally resource constrained sensor network, the rate of data generation and transmission is controlled by using a top down strategy where the search comes first - in terms of usefulness of data collected, i.e. the “Quality of Information” . Here we first search for what needs to be stored - and hence we “search and store”.

Hence the transformation of data into information (or knowledge) requires a more top down approach which can balance information needs and resource utilization rates. If we begin by defining our information needs (i.e. specifying the Quality of Information requirements), we would be able to better utilize the often non-renewable resources of a sensor network. We believe that adding the Quality of Information (QoI) as another dimension will greatly benefit the knowledge which can be extracted from a sensor network. Mapping the aspects of QoI to different kinds of sensor network applications will allow the user to more clearly specify what he or she wants from the sensor network deployment. By providing a framework to deliver what the user wants, we give more flexibility to the user for defining his/her needs and to understand and analyze the tradeoffs involved.

The most important benefit of the QoI approach to routing on sensor networks is the explicit knowledge of the various tradeoffs involved, which leads to higher quality of data collected from the sensor network. The explicit use of QoI attributes provides a considerable variety of options for data collection. A second benefit of the QoI approach is a better utilization of network resources. In many scenarios, the use of QoI

for specifying the requirements for the data collection process will actually allow for better utilization of network resources than the case where QoI is not considered. For example, we may wish to collect information from highly relevant sensor nodes. We expect to find a fair degree of redundancy in the network; so many sensors could possibly satisfy the relevance constraint. We may choose only a few sensors among them for the data collection task. The sensors chosen may have higher residual energies, and thus we could perform the data collection in an energy-balanced fashion. We could also select sensors which are closer to the sink, hence reducing the latency of data collection. We can clearly see that using the QoI approach (in this case, specifying that the user is interested in the 'relevance' of the data) allows us to utilize network resources far more effectively while also satisfying the end user requirements.

We present the case for using an application centric viewpoint for improving network resource utilization. Specifically, we recommend the use of well defined attributes for the information quality to be applied to the data which is being collected from the network - called as Quality of Information (QoI), similar to Quality of Service. Using QoI attributes for specifying the type of data to be collected would be an abstraction which provides a fair amount of flexibility to the end user while also allowing good network resource utilization.

1.7 Organization of the dissertation

The rest of the dissertation is organized as follows. Chapter 2 describes multi-metric shortest path techniques for extending the lifetime of a sensor network and thus keeping

it operational for a long time. Chapter 3 proposes interference aware routing by analyzing the impact of interference on the throughput which can be achieved in a wireless network. By identifying interference patterns which are non-destructive, i.e. do not lead to a reduction of throughput, we propose ideas for combining route selection and transmission scheduling in such a way as to increase the throughput utilization. Chapter 4 provides distributed techniques for the implementation of the aforementioned interference aware routing algorithms. Using multiple radios can lead to improved resource utilization in sensor networks. Chapter 5 identifies and provides solutions for a connectivity problem which arises in the context of multi-radio sensor networks. It has been shown that the performance of distributed algorithms in terms of scalability and message complexity can be improved using network decomposition techniques. Chapter 6 describes network decomposition techniques which are lifetime aware and shows how these ideas can be applied in the context of energy aware routing discussed in Chapter 2. Chapter 7 makes the case for data collection in sensor networks from an application centric viewpoint and shows how it can impact resource utilization in a wireless sensor network. The conclusions of this dissertation are presented in Chapter 8 along with a discussion of the future directions in which this research could be extended.

Chapter 2

2 Extending network lifetime

2.1. Introduction

Energy management in wireless sensor networks is an important consideration due to the limited energy availability in battery powered wireless devices. Wireless communication consumes a significant amount of energy and it is important to minimize the energy costs for communication as much as possible by practicing energy aware routing strategies. Such routing strategies can increase the network lifetime. In this chapter, we focus on developing routing strategies for multiple hop wireless networks where all the nodes are powered by a battery or other external power sources such as solar energy. Usually network lifetime is quantified through the number of packets that can be transferred in the network before the source and destination get disconnected from each other [48, 65]. A suitable energy-aware routing strategy for wireless networks is to use those wireless nodes with high energy levels and avoid those with low energy levels.

We model the wireless network as a graph wherein, the vertex represents a wireless device and an edge between two vertices indicates that they are in direct communication range of each other. The weight on a vertex indicates the residual energy available at that wireless node and the weight on an edge (u, v) represents the amount of energy required by node u (resp. v) to transmit one unit of data to node v (resp. u). The *residual energy of a path* is defined as the minimum energy level of any node in the path (referred to as **metric 1** in our work). The max-min routing paradigm

suggested in the literature [1, 48, 81] aims to maximize the network lifetime by finding the path where the residual energy is the maximum and forwards packets through this path termed as the *maximum residual energy path*. The *energy consumed along a path* (or simply the *energy of a path*) is the sum of the weights on the edges along the path (referred to as **metric 2** in our work). While some defining characteristics of wireless networks, such as lossy links, non-uniform transmission range etc. cannot be completely described by this somewhat idealized graph model it can be used as a good starting point for estimating lifetimes. Also, we are interested in maximizing the lifetime which can be achieved, and the more realistic network models are unlikely to improve on these bounds. That is, the realistic network models typically provide a smaller value of the computed lifetime when compared to the more idealistic model we use, and we are interested in finding lifetimes which are as close as possible to the maximum theoretically achievable value.

Earlier routing strategies which utilize the concept of the residual energy (either directly or indirectly) proposed in the literature include MMBCR [81], MRPC [1] and max-min zP_{min} [48]. These research works also caution that merely using the residual energy strategy may lead to higher energy consumption in the network, since the energy consumed along the data forwarding path is not taken into consideration. They suggest that a good energy-aware routing technique should balance two different goals: choosing a path with maximal residual energy and choosing a path with minimal energy

consumption. We note that the residual energy along a path is a concave metric[7], whereas the energy consumed along a path is an additive metric.

In this chapter, we present three polynomial time combinatorial techniques which can provide a good balance between metrics 1 and 2. The first technique, called the Shortest Widest Path, first maximizes the concave metric (the residual energy of a path) and then minimizes the additive metric (energy consumed along a path). The second technique, which we call the Shortest Width Constrained Path, finds paths with a suitably high residual energy (which may not be the maximum), and then minimizes the energy consumed along such a path. Lastly, our third approach (Shortest Fixed Width path) is similar to the second approach in the sense that it finds a minimum energy path among the paths that have a high residual energy. However, unlike the shortest width constrained path, it does not change the residual energy with each route calculation; the residual energy level is changed only when it becomes infeasible to find paths between the source and the destination at the current residual energy level. Our simulation studies show that the performance of the proposed techniques is superior to that of the best known routing approach proposed in the literature (Park and Sahni [65]). We also discuss the performance of our proposed algorithms in a distributed setting. Even if the nodes lack an accurate global knowledge of the instantaneous node energy levels, we find that the two phased routing techniques perform reasonably well. We also find that the proposed distributed techniques outperform the distributed implementation (that we have developed) of the algorithm proposed by Park and Sahni [65]. The results of our research work has been presented in [59] and [62].

[7] For definitions of concave and additive metrics, see Wang and Crowcroft [84].

The chapter is organized as follows. Section 2.2 provides an overview of the related work. Section 2.3 provides the definition of the network lifetime problem as well as our basic solution. In Section 2.4, we deduce a relationship between the residual energy of nodes along a path and the minimum energy for a given residual energy value. In Section 2.5, we describe two other solutions which may be considered derivatives of our basic solution. Section 2.6 compares the performance of our basic algorithm with other approaches on a benchmark topology to show that using the widest path (also called as max-min) approach usually improves the network lifetime. We use empirical evaluations to discuss the performance of our three solutions on general topologies in Section 2.7. We describe the distributed implementations of the solutions we propose, and their performance, in Section 2.8. We conclude our discussion in Section 2.9.

2.2 Related work

We are interested in lifetime maximization using centralized approaches. Localized algorithms for the lifetime maximization problem have been proposed in the literature under some restricted models. For example, Madan and Lall [53] propose a linear programming based approach for lifetime maximization where the information generation rate is assumed to be constant. Also, Zussman and Segall [92] have proposed localized algorithms for the anycast routing model for emergency network applications. In the centralized case, notable routing strategies which utilize the concept of the residual energy (either directly or indirectly) include CMMBCR [81], max-min zP_{min} [48] and CMRPC [1]. The pioneering work done by Toh et al [81] establishes multiple formulations for the online energy aware routing problem, of which CMMBCR (Conditional Min Max Battery Capacity Routing) is shown to be better than

the remaining approaches. CMMBCR uses minimum energy paths in the first phase, and then shifts to paths with maximum residual energy after node energy levels in the network fall below a certain threshold.

Qun Li et al. [48] describe the max-min zP_{min} algorithm. The max-min zP_{min} algorithm attempts to balance metric 1 and metric 2 by calculating a path based on the residual energy levels, but then rejecting any path whose total energy is more than a factor *z* times the minimum energy path. The quality of the solution provided by the max-min zP_{min} algorithm depends on the empirically generated parameter *z*, and this does not always provide an optimal solution.

The CMRPC algorithm [1], which is similar to CMMBCR algorithm proposed in [81], uses the residual 'packet capacity' instead of the residual energy for optimization. The residual packet capacity denotes the capacity of each edge in the graph based on the residual energy, the communication cost of the edge as well as the initial energy levels.

Chang and Tassiulas [15] combine metrics 1 and 2 into a single metric and run Dijkstra's algorithm on this new metric. While it is a good heuristic, this method does not actually optimize either metric and makes their approach closely dependent on the empirical values assigned to the parameters used as power terms in the combined metric.

Park and Sahni [65] present the Online Maximum Lifetime (OML) heuristic, which is an enhancement of the CMAX algorithm presented by Kar et al [41]. OML initially removes edges with low residual energy from the graph. The edge weights are then

modified such that a high cost (and thus a heavy penalty) is associated with edges having low residual energy or high communication cost. Dijkstra's algorithm is run on the modified graph such that the paths found usually use nodes with high energy levels and edges with low energy costs. They report that OML gives the best network lifetime among all routing approaches in the current literature.

2.3 Problem definition and proposed solution

Let $G = (V, E)$ represent a wireless network with nodes V and edges E. Let $w(u)$, $u \in V$, represent the available energy at node u. Let $c(u, v)$, $(u, v) \in E$, be the energy required to transmit a packet from node u to node v. We assume that $c(u, v) = c(v, u)$, for all $(u, v) \in E$.

Let $P(v_0, v_k) = v_0, v_1, \ldots, v_k$, be a path in G. The energy of the path $P(v_0, v_k)$ denoted $e(P(v_0, v_k))$ is given by

$$e\big(P(v_0, v_k)\big) = \sum_{i=0}^{k-1} c(v_i, v_{i+1}) \quad (1)$$

The *residual energy of a path* $P(v_0, v_k)$ denoted $r(P(v_0, v_k))$ is defined as

$$r(P(v_0, v_k)) = min_i(w(v_i)\text{-}c(v_i,v_{i+1})),\ 0 \le i < k. \quad (2)$$

When a packet is sent along $P(v_0, v_k)$, we need to perform the following *energy decrease operation* on each node along the path except on the node v_k: $w(v_i) = w(v_i) - c(v_i, v_{i+1})$, $0 \le i < k$. That is, after the packet is sent by a node, the energy level of the node is decremented by the amount of energy required to send the packet. In our model, we have not included the cost of reception explicitly to avoid clutter in our discussions

and such a cost can be easily incorporated in our proposed work. We discuss this at the end of this section.

Given a source s, a destination t, and a *single packet* to be routed, we can define two problems formally:

a. *Minimum energy path problem*: Find a path $P(s, t)$ with minimum $e(P(s, t))$.
b. *Maximum residual energy path problem*: Find a path $P(s, t)$ with maximum $r(P(s, t))$.

Let G_0 be set to the initial network G. Assume that $P_0(s, t)$ is a path in G_0. Now after routing a single packet along the path $P_0(s, t)$ and applying the decrease operation we obtain a new network G_1. In the network G_1 the edge weights are the same as in G_0 but the node energy levels are different. If a node u's energy level becomes 0 after the decrease operation, node u and its associated edges $(u, v) \in E$ as well as $(v, u) \in E$ are removed from the network. For the second packet we can again find a path $P_1(s, t)$ in G_1 and the process continues until there exists no path between s and t in some network G_k. That is, we can send at most k packets from s to t before the network is disconnected. The goal of the *network lifetime problem* with respect to a source s and destination t is to find paths $P_0(s, t)$, $P_1(s, t)$,… $P_{k-1}(s, t)$, such that the value of k is maximized. Here we would like to point out that while our goal is to maximize the network lifetime, it has been shown that computing the value of k is NP-hard [65]. While it would be more appropriate to call this the lifetime improvement problem, we have decided to use the standard terminology in the literature and refer to this as the lifetime maximization problem.

We will begin by outlining our solution to the network lifetime problem. The graph *G* is modified into an energy graph *EG* = (*V, E´*) as follows. We leave the vertices intact but replace each single undirected edge in *G* with two directed edges. The weight of a directional edge in *EG* is made equal to the difference between the originating node's energy level and the transmission cost along the edge. This is also the residual energy of a node as defined in Li et al [48]. In Figure 2.2 (a) we have shown an example wireless network and in Figure 2.2 (b), the corresponding energy graph.

```
Algorithm MaxResidualEnergy (EG, source)
// s – source node
//Adj[s] – adjacency list representing the neighbors of the source
// weight(u,v) = capacity of edge (u, v) in graph EG
// width(u) = weight function for a node u in graph EG
begin
1. width[s] = 0
2. width[v] = weight(s,v) if v ∈ Adj[s]
3. width[v] = 0 for all other nodes
4. S = s
5. Q =V[EG] – s
6. while Q ≠ □ {
7.    find u ∈ Q where width[u] is maximum ∀ u ∈ Q
8.    Q = Q – u
9.    S = S ∪ {u}
10.   for each vertex v ∈ Adj[u]
11.      if v ∉ S do RELAX(u, v, weight(u, v))
12. }
End

RELAX (u, v, weight(u,v))
1.if width[v] < min(width[u], weight(u, v))
2.   width[v] = min(width[u], weight(u,v))
3.   Pred(v) = u
```

Figure 2.1 MaxResidualEnergy algorithm to find the widest *s*-*t* path

Given a source node *s* and a destination node *t*, a two-phased routing algorithm called as shortest-widest path, is executed on this energy graph EG to find a suitable path between *s* and *t*. In phase I, we apply a variant of the Dijkstra algorithm (Algorithm MaxResidualEnergy) which returns a path whose residual energy will be the maximum in the network. Let the path returned by phase I have a residual energy of

B. It is to be noted that there could be many paths in the network between *s* and *t* with a residual energy of B. The algorithm MaxResidualEnergy given in Figure 2.1 computes the value of the maximum residual energy of the paths originating from a given source to all other nodes.

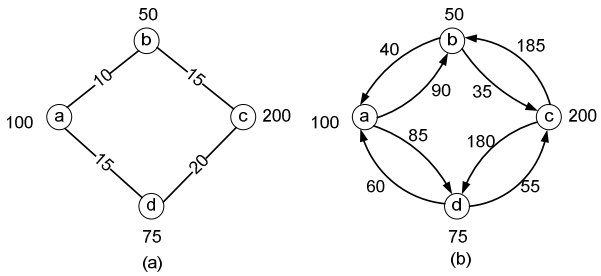


Figure 2.2(a) A graph showing energy levels at nodes and energy required to transmit at each edge.

Figure 2.2 (b) shows the corresponding energy graph.

In phase II, we choose from the set of all paths with a residual energy of *B*, a path which has the lowest energy consumption. This can be accomplished as follows. Let *E"* be the set of edges in *EG* whose residual energy is less than *B*. These edges are pruned from *EG* and by using Dijkstra's algorithm, the least energy cost path on *EG\E"* is determined. If there are many such paths, we arbitrarily choose one among them.

It can be noted that this algorithm can also handle the energy cost of reception if such information was available. We would need to modify step 1 of the RELAX procedure to add the energy cost of reception. It must also be noted that the pruning is temporary, in other words, the pruned edges are restored before the next route computation. Each time a path is computed, we will invoke Dijkstra's algorithm twice in sequence. Hence our algorithm has a complexity equal to *k* times the complexity of

Dijkstra's algorithm, where *k* is the number of packets transmitted. The algorithm ShortestWidestPath given in Figure 2.3 calculates the shortest widest path on the modified graph EG for a given source and destination pair.

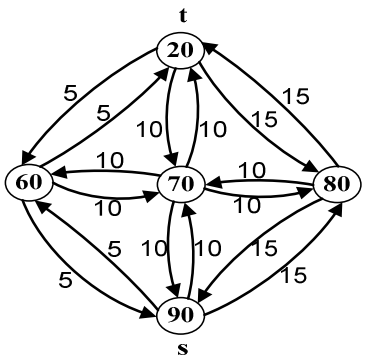
```
Algorithm ShortestWidestPath(EG, source, dest)
// source – source node; dest – destination node
// MinimumEnergyPath(EG, source, dest) uses Dijkstra's algorithm to find the
// minimum energy path in the graph EG based on energy cost on each edge
begin
1. w = WIDEST(EG, source, dest)
2. for each edge e ∈ EG
3.    if (weight[e] < w) EG = EG\e
4. p = MinimumEnergyPath(EG, source, dest)
5. Restore all edges back in EG
end

WIDEST(EG, source, dest)
1. MaxResidualEnergy(EG, source)
2. return width[dest]
```

Figure 2.3 Shortest Widest Path algorithm

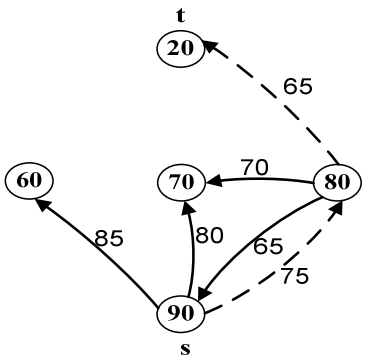
2.4 Relationship between the total energy and the residual energy of paths

While the maximum residual energy path computation identifies the path whose bottleneck edge has maximum energy and allows us to discover the *maximum residual energy subgraph*, we can also define *residual energy constrained subgraphs*.

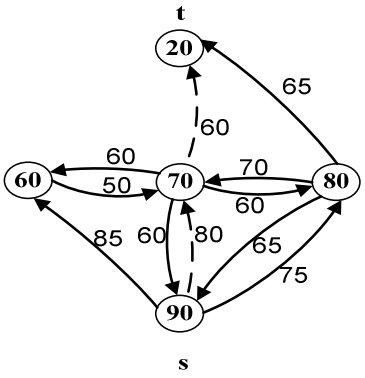
Definition*:* Let *EG*(*w*) represent the subgraph constructed from the original residual energy graph *EG*, by pruning all those edges in *EG* which have residual energies less than *w*.



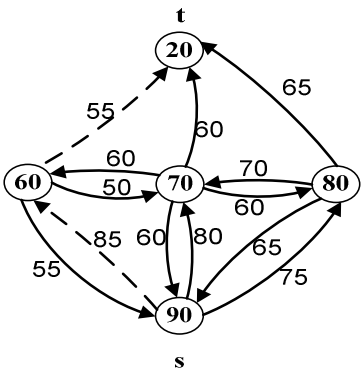
Original graph G. The node labels are node energy levels. The edge labels are the energy costs to transmit a packet along the edge



EG(65)
Minimum energy path (represented by dashed line) requires 30 units of energy



EG(60)
Minimum energy path (represented by dashed line) requires 20 units of energy



EG(55)
Minimum energy path (represented by dashed line) requires 10 units of energy

Figure 2.4: Total energy of path vs the residual energy

Let $E_{min}(w)$ represent the energy consumed along the minimum energy path from source *s* to destination *t* in $EG(w)$. For example, in the graph shown in Figure 2.4, we can observe that for $EG(65)$, which is the subgraph constructed by pruning all edges whose residual energies are below 65, the minimum energy path from *s* to *t* requires 30 units of energy, i.e. $E_{min}(65) = 30$. On the other hand, $E_{min}(60) = 20$ and $E_{min}(55)= 10$. That is, as the constraint on the residual energy increases, the energy required for the minimum energy path also increases.

If we were to repeatedly compute the minimum energy path for all the possible residual energy values of a graph (a graph can have at most *m* such discrete values, where *m* is the number of edges) we would obtain a non-decreasing graph similar to the one shown in Figure 2.5. This result is stated in the following lemma:

Lemma* 2.1:* Let $E_{min}(w)$ represent the energy consumed along the minimum energy path from *s* to *t* in $EG(w)$. Given the residual energies of all the edges in the graph *EG* in increasing order as $(w_1, w_2 \ldots, w_m)$, i.e. $w_1 \leq w_2 \leq \ldots \leq w_m$, then $E_{min}(w_1) \leq E_{min}(w_2) \leq \ldots \leq E_{min}(w_k)$.

Proof: Let $G_1 = EG(w_1)$ and $G_2 = EG(w_2)$ where $w_1 \leq w_2$. Any edge in G_2 also exists in G_1, by definition. Thus the minimum energy path in G_1 cannot have higher energy than the minimum energy path in G_2. In other words, $E_{min}(w_1) \leq E_{min}(w_2)$. By induction, we get the result. ■

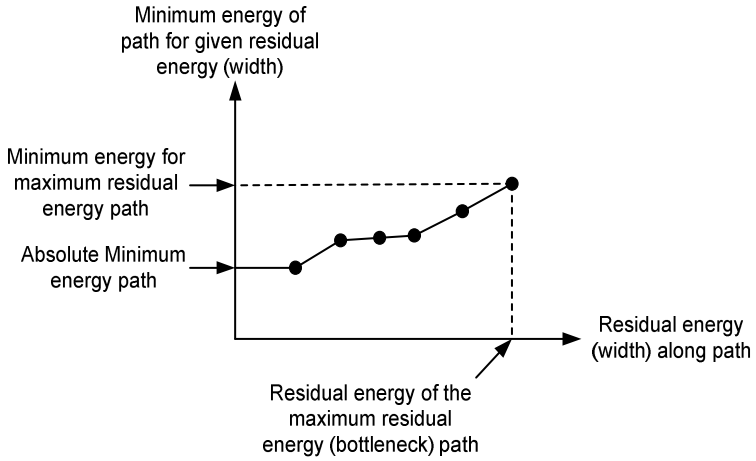


Figure 2.5: Total energy of path Vs residual energy graph

We gain some useful insight from the relationship between the residual energy along a path and the minimum energy path possible for such a residual energy. For one,

knowing that we have a lower energy path, we could avoid routing the packet along a path with identical residual energy but which consumes a much higher energy for the entire path. From Figure 2.5 we can infer that using a higher residual energy path also automatically implies we may spend more energy in forwarding a packet along that path.

2.5 Derivatives of Shortest Widest path

In addition to the shortest widest path discussed earlier, we propose two other online energy aware routing algorithms. These algorithms too are two-phased strategies and are derivatives of the shortest widest path algorithm discussed before. The derivatives are based on the following idea: we first prune off all edges in the graph which have residual energy levels (in the modified energy graph) below a certain cutoff value, and we find the minimum energy path on the remaining subgraph. The difference between the two algorithms lies in the value we choose for the cutoff and the way we select the cutoff values.

2.5.1 Shortest width constrained path

Besides the shortest widest path (or the minimum energy maximum residual energy path), we can also use paths which are “tradeoffs” in the solution space. That is, by sacrificing the high residual energy of the absolute widest path, we could use a path with a slightly lesser residual energy, but which can provide us the benefit that it consumes lesser energy along the path. We call this routing approach the *shortest width constrained path*. We place a constraint on the width (residual energy) to be a certain fraction of the maximum possible residual energy for the given source destination pair. Suppose the width of the absolute widest path between source and destination is *W*. Let

the width (bottleneck residual energy) of the minimum energy path between the source and destination be W' . Let $\Delta W = W - W'$. We multiply ΔW by a factor η (η < 1) and add that to W' to get the constraint width W_{const}. In other words, $W_{const} = W' + \eta \Delta W$. The edges whose width is less than this constraint width are removed (temporarily) from the graph. The minimum energy path is computed on the remaining edges, and this path is used for routing. In our experiments, we set η = 0.5. The shortest width constrained path algorithm is described in Figure 2.6.

```
Algorithm ShortestWidthConstrainedPath (EG, source, dest)
begin
  1. w = Widest(EG, source, dest)
  2. p = MinimumEnergyPath (EG, source, dest)
  3. for each edge e in p
  4.    if (weight[e] < minW) minW = weight[e]
  5. w' = w  + (η × minW)
  6. for each edge e ∈ EG
  7.    if (weight[e] < w') EG = EG\e
  8. p = MinimumEnergyPath(EG, source, dest)
  9. Restore all edges back in EG
end
```

Figure 2.6 Shortest Width Constrained Path

2.5.2 Shortest fixed width path

A third algorithm that we propose fixes the width (residual energy) of the path at a certain value (for each source destination pair), prunes the edges with residual energy which is less than this fixed value, and finds the minimum energy path on the pruned graph. This procedure is repeated until no path can be found for the given width, at which point the width is decreased (by a constant fraction) and so on, until the source and destination get disconnected. As an example, let us suppose the widest path between the source and the destination has a width of 100. We may select our fixed width to be a high fraction of 100, say 80. Now we prune all edges in the graph with

width less than 80, and keep finding the minimum energy path until the source and destination get disconnected (on the pruned graph). Now we change the fixed width to 60, and repeat the process. We call this the *shortest fixed width algorithm*. The shortest fixed width path algorithm is given in Figure 2.7.

```
Algorithm ShortestFixedWidthPath(EG, source, dest)
//initialNodeEnergyLevel – the initial energy level of the nodes in the network
begin
  1.fixedWidth = 0.95 × initialNodeEnergyLevel
  2.for each edge e ∈ EG
  3.   if (weight[e] < fixedWidth) EG = EG\e
  4.p = MinimumEnergyPath(EG, source, dest)
  5.if (p not found)
  6.   if fixedWidth ≤ 0 stop
  7.else
  8.   fixedWidth = fixedWidth – (0.2 × initialNodeEnergyLevel)
  9.goto step 2
end
```

Figure 2.7 Shortest Fixed Width Path

2.6 Performance of the shortest widest path approach on a benchmark topology

We compare the performance of the proposed shortest widest path approach with those of two other approaches in the literature on an illustrative topology shown in Figure 2.8 (from Li et al [48]). In the network shown in Figure 2.8, each node (other than source 1 and destination n) has energy $20 + \epsilon$. The weight of each edge (along the semi-circle) is set to 1, but the weight of each straight edge is set to 2. The energy of the source is infinite. We can note that the residual energy of the path along the semicircle is 19 units, while the straight edge path $(1, x, n)$, where $3 \leq x \leq n\text{-}2$ has residual energy 18 units which is less than the path along the arc of the semicircle. Hence Li et al. [48] state that using a max-min (widest path) approach, it is possible that only twenty messages can be sent before the network gets disconnected (by sending all messages along the semicircular arc). The authors then state that using the straight line edges $10(n\text{-}4)$ messages can be sent before the network gets disconnected [48], where n is the number

of nodes in the network. This is achieved by alternately sending the packet through different nodes lying inside the semicircle. For example, the first message will take the path (1, *n*-2, *n*). The second message will take the path (1, *n*-3, *n*), and so on until the $(n\text{-}4)^{th}$ message will take the path (1, 3, *n*).

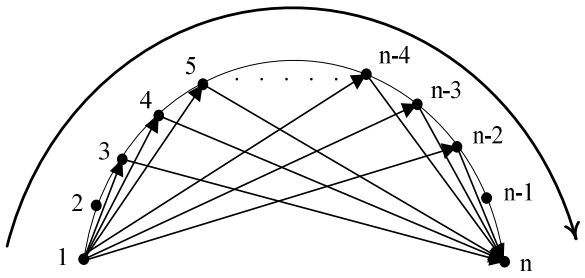


Figure 2.8: Benchmark topology from max min zP_{min}

Banerjee and Misra [1] define the residual packet capacity as the number of packets which can be transmitted by a node at its current energy level. In their algorithm CMRPC, they define a parameter γ which represents the threshold energy level of the critical nodes. When nodes reach this energy level, they shift from minimum energy routing to maximum residual capacity routing.

Suppose the parameter γ is set at 0.5 (representing 50% of node's energy). The authors of the MRPC algorithm do not mention how they make the choice of minimum energy paths when there is more than one. Suppose we use the sequence {(1, *n*-2, *n*), (1, *n*-3, *n*),..., (1, 4, *n*), (1, 3, *n*)}, we can send 5(*n*-4) messages using the straight line edges after which each forwarding node (except 2 and *n*-1) will be left with energy 10 + ϵ. Since we reach the threshold value, we start using the maximum residual capacity paths. Using the maximum residual capacity paths, only 10 more messages could be sent, for a total of 10 + 5(*n*-4) messages. In fact, we may not be able to send more than $10\gamma(n\text{-}4) + 20(1\text{-}\gamma)$ messages. The maximum value of this quantity happens when $\gamma = 1$ (which

means packet capacity is never used), in which case we can still send only 10(n-4) messages.

Algorithm	Total messages transmitted
Greedy max-min [48]	20
Qun Li et al. [48]	10(n-4)
Banerjee and Misra [1]	10(n-4)
Shortest Widest Path	10(n-3)

Table 2.1: Number of messages transmitted using different algorithms

On the other hand, if one were to use the two-phased shortest widest path approach we have proposed, the following paths will be used for routing. We will repeat the sequence {(1, n-2, n-1, n), (1, n-2, n-1, n), (1, n-3, n), (1, n-4, n), (1, n-5, n)…, (1, 4, n), (1, 3, n)} before source and destination get disconnected. Consequently, it is easy to see that a total of 10(n-3) messages can be sent before the nodes run out of energy. This demonstrates a key aspect of our two phased strategy. There are multiple possible widest paths which can be used for routing, and a good choice among these possibilities will still allow us to use the widest path and achieve a good lifetime (we use the widest path which is shortest). A poor choice of the widest path in Li et al. [48] makes them conclude that it is unsuitable in general to use the widest path, which is not actually the case. A summary of these results is shown in Table 2.1.

2.7 Performance on general topologies

Having studied the performance on a benchmark topology, we now discuss the performance of our three algorithms on general topologies. We used LEDA (Library of Efficient Data structures and Algorithms) [31] as our simulation tool.

2.7.1 Simulation settings

In our experimental study, we compare the performance of the proposed algorithms with the on-line maximum life-time (OML) heuristic proposed by Park and Sahni [65] and the max-min zP_{min} algorithm proposed by Li et al [48]. Reference [65] has shown the superiority of OML over other existing works. For completeness, we have also included performance evaluation comparisons with the max-min zP_{min} algorithm [48].

Topologies Used: We use a topology which is identical to that used for OML. We randomly populate a 25×25 grid with 50 nodes. We add edges to the network if the nodes are within each others' transmission range, which is decided by the transmission radius r_T. The energy cost of transmitting a single packet is calculated as $0.001 \times d^3$ where d is the Euclidean distance between the nodes. These settings are identical to the ones used for OML.

Session Length: A parameter that we generalize in our simulation study is session length. Earlier works assume that a single packet is transmitted in a session between a given node-pair. However, in reality, it is highly likely that multiple packets will be exchanged in a session between two nodes. Therefore, in our experiments, we assume that k packets are transmitted in a session between a given node-pair. We vary the value of k and observe the performance of the different routing schemes. As in other works in the literature, we calculate the route afresh for each packet transmission.

Traffic pattern used: We conduct our experiments assuming an *any-to-any communication* model, i.e. source-destination pairs are selected at random and packets are transmitted between them.

Unless otherwise mentioned, we use the following default values: there are 50 nodes placed randomly on a 25×25 grid, the transmission radius is set to 8, the session length is set to 1 (single packet), the initial energy level for each node is set to 30 and any-to-any communication pattern is assumed.

We use 10 different random topologies, and 10 different request sequences for each such random topology. Each request sequence is an infinite set of requests of the form $\{(s_1,t_1), (s_2, t_2),.., \}$, where (s_i, t_i) represent the source and the destination for the given packet. The same source destination pair is allowed (and expected) to repeat in the set. During algorithm execution, we choose the next outstanding request from this set until network disconnection. The average value of these 100 runs is reported here. The lifetime of the network is calculated as the total number of packets which can be transmitted in the network before the first session failure occurs.

2.7.2 Performance comparison with basic algorithms

First, we would like to demonstrate that we do indeed get significantly improved lifetimes by using power-aware algorithms. We show this by comparing the lifetime obtained by the Shortest Widest path and derivative algorithms and two other power-aware algorithms – namely OML and max-min zP_{min} - against the lifetime obtained by two basic algorithms - the Minimum Energy Path and the shortest path (hop count). We set the session length as 1, the transmission radius as 8, and the number of nodes in the network as 50. We used 10 different topologies and 10 request sequences and report the average value of the 100 runs.

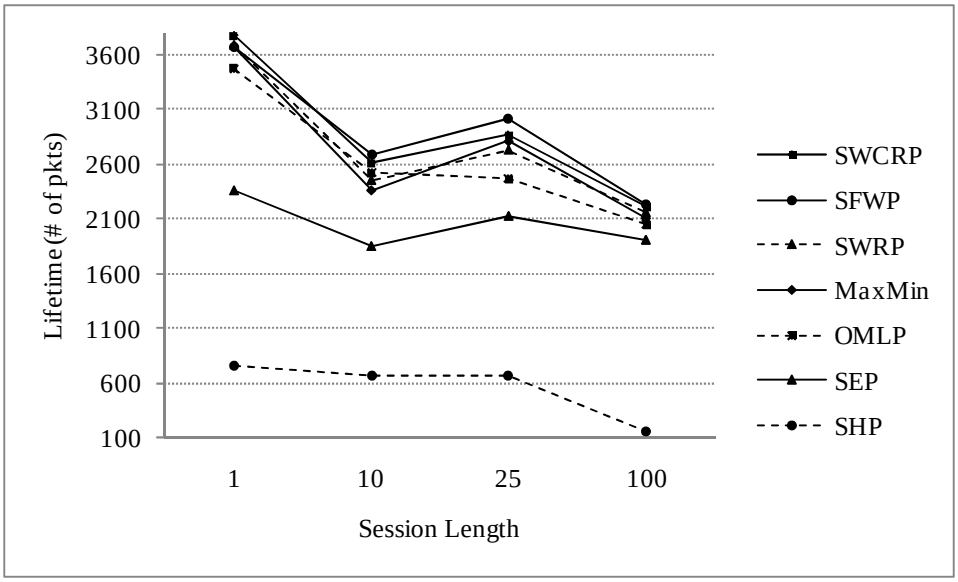


Figure 2.9 Performance of power-aware algorithms vs basic algorithms

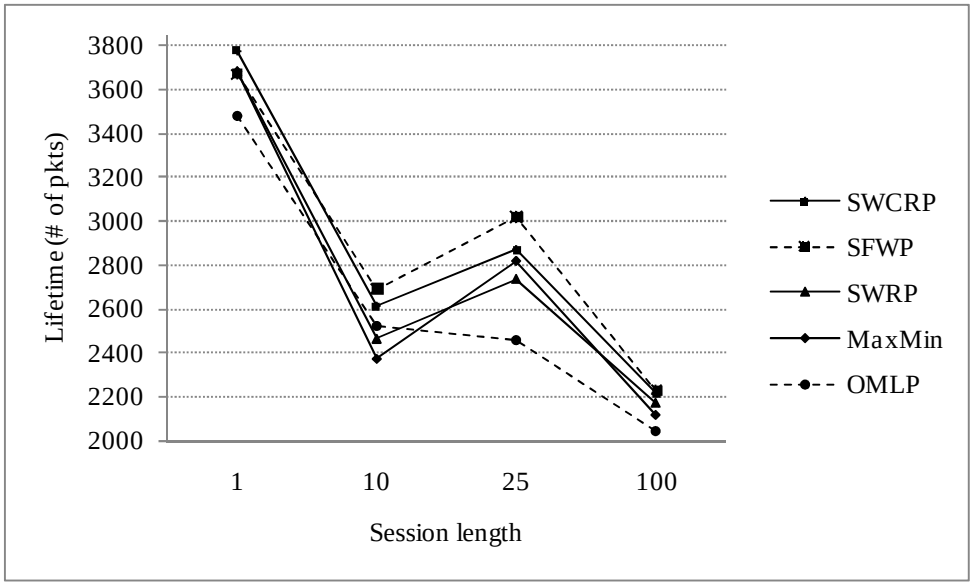
In Figure 2.9 we show the variation of lifetime as a function of the session length. In the graph shown, the acronyms used for representing the different algorithms are as follows: SEP – Shortest energy path, or minimum energy path, SHP – shortest hops path or shortest path, MaxMin – max-min zP_{min} proposed by Li et al. [48], OMLP – Online Maximum Lifetime heuristic proposed by Park and Sahni [65], SWRP – shortest widest residual path, SFWP – shortest fixed width path and SWCRP – Shortest Width Constrained Residual Path. We have observed that using power aware algorithms improve the lifetime by nearly 70% over the shortest path algorithm and by as much as 30% over the minimum energy path. Similar results were seen when performing comparisons using other metrics such as transmission radius and node densities. Hence this justifies the energy expended in finding power aware routes.

2.7.3 Effect of session length

We first observe the effect of varying the session length *k* on the lifetime achieved by various routing algorithms. Here, we send *k* packets at once for each session. Figure

2.10(a) shows the results of our experiments. In general, the widest path and its derivative algorithms have a much better lifetime than the OML heuristic, and the shortest width constrained path algorithm (SWCRP) consistently betters the max-min zP_{min} algorithm (MaxMin). This supports our rationale behind selecting the widest path as well as its derivative algorithms for improving the network lifetime.

Figure 2.10(b) shows the fraction of energy remaining in the network at the time of first session failure. That is, we consider the remaining energy levels at each node as a fraction of its initial energy level, and calculate their average. It is interesting to note that the network nodes retain a higher average residual energy under the proposed routing algorithms than under OML and max-min zP_{min}. This shows that the widest path and its derivative algorithms are able to send more packets at fewer energy cost and hence there is more energy available for the nodes to use for other tasks. We obtained similar data for all the other simulation values.



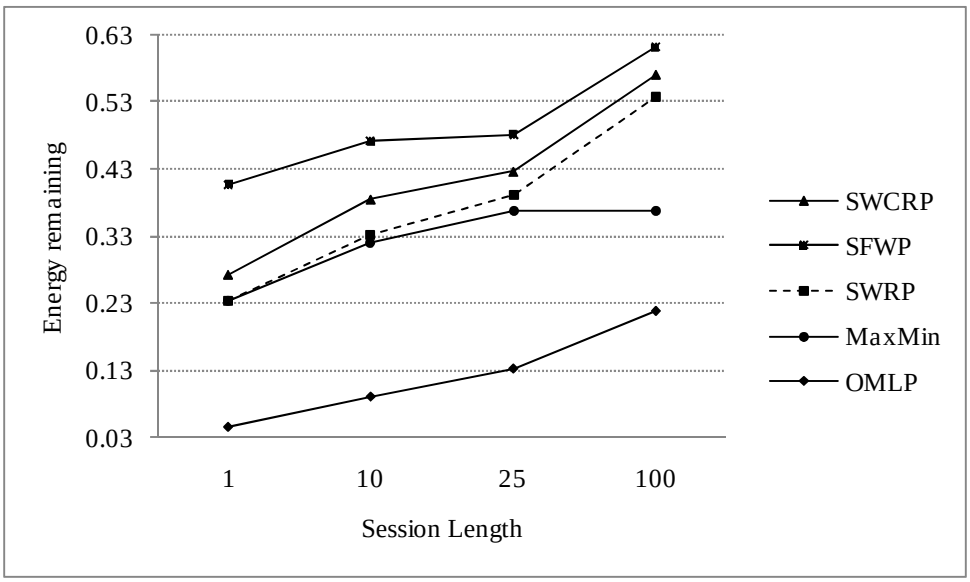
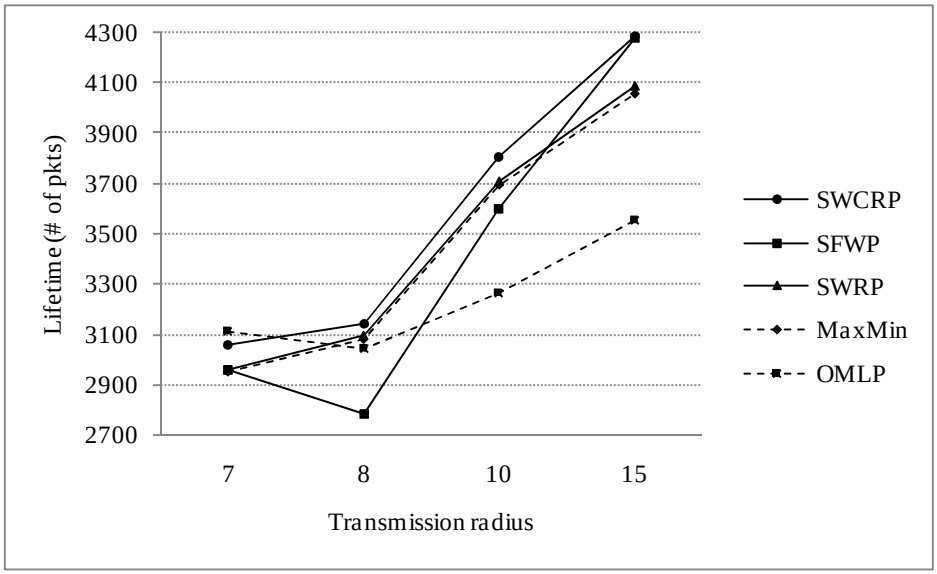


Figure 2.10 (a): Lifetime vs Session length (b): Energy remaining vs session length

2.7.4 Effect of transmission radius

Figures 2.11(a) and 2.11(b) show the impact of the transmission radius on the lifetime and energy levels of the sensor network. We can see again that the shortest widest path and its derivatives perform much better than OML and equal or better the performance of max-min zP_{min}.



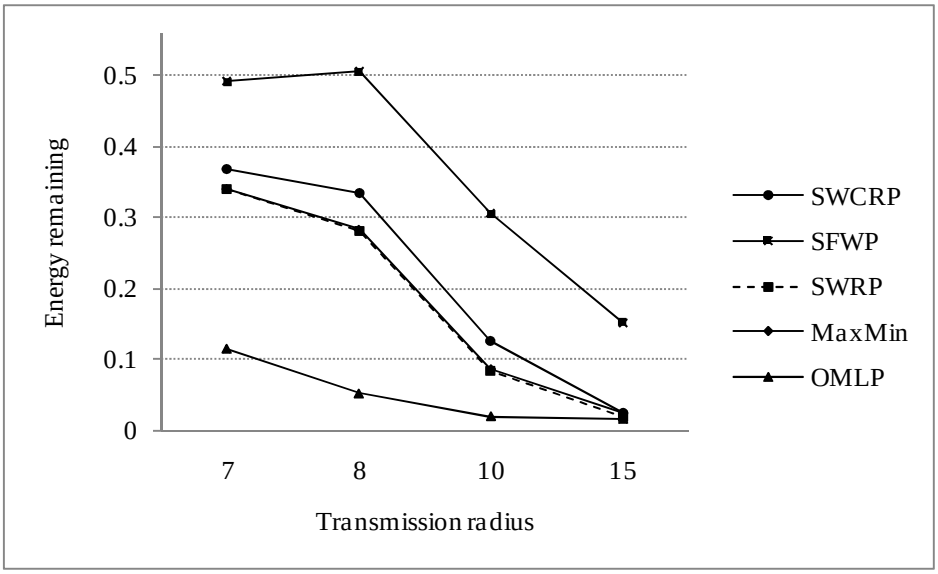
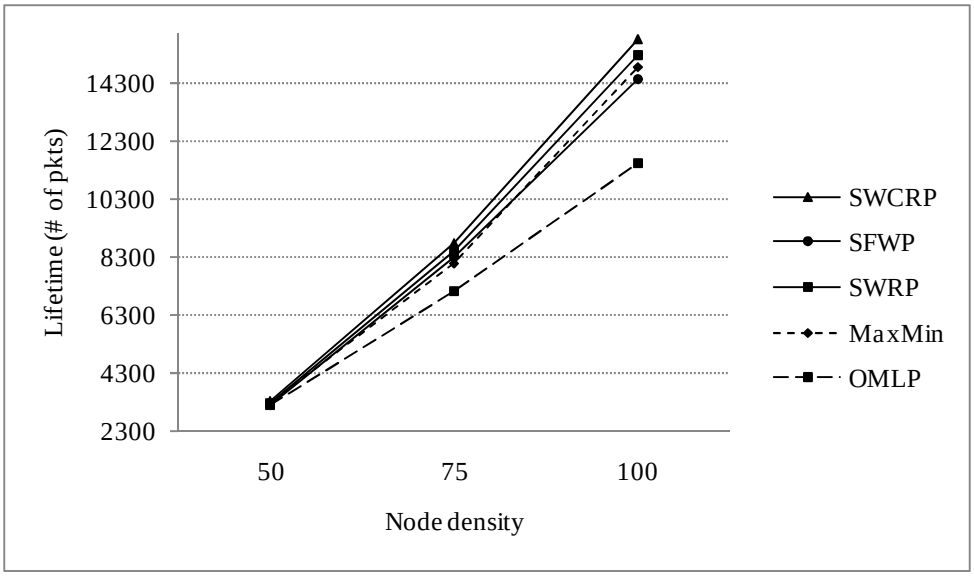


Figure 2.11 (a): Lifetime vs transmission radius (b): Energy remaining vs transmission radius

2.7.5 Effect of node density

We evaluate the performance of the algorithms for the node densities 50, 75 and 100. We increase the number of nodes while keeping the total area constant, thus increasing the density. The results are presented in Figure 2.12 (a) and (b). Here again we can see that the shortest widest path and its derivatives are generally outperforming OML and max-min zP_{min}.



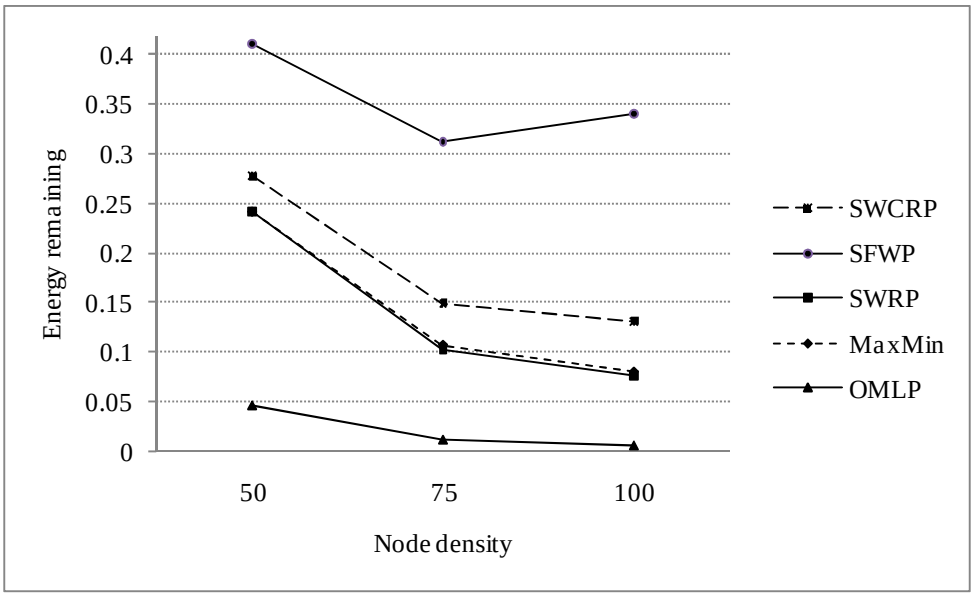


Figure 2.12 (a): Lifetime vs node density (b): Energy remaining vs node density

Among the three widest path algorithms proposed, we note that both the shortest width constrained path as well as the shortest fixed width path algorithms provide better lifetimes in general (than the shortest widest path).

2.7.6 Impact of communication cost estimate on lifetime

While communication is a primary source of energy depletion in sensor networks, we would also need to consider other factors of energy depletion to get a better understanding of the lifetime. For this we have considered the following question – if the estimate of the communication cost of each edge in the network is off by x% (in other words, the energy to transmit a packet costs x% more energy than estimated) what is the impact on the network lifetime?

Here we consider the impact on the Shortest Widest Residual Path algorithm (other algorithms indicate similar or better results). We consider estimate errors of x = 5, 10, 15 and 20% respectively. Other simulation settings are similar to the settings in Section 2.7.2. We note the difference in the lifetime – that is, we measure how much lesser (as a percentage) the actual lifetime is from the value computed if the communication cost was estimated correctly. Figure 2.13 shows the results of this evaluation. As expected, the percentage error in lifetime increases as the error in the cost estimate increases. The x = y line represents the values of x and y such that the percentage difference in the lifetime equals the percentage error in computing the communication cost. The curve for the percentage lifetime difference always lies below the x=y line and thus we note that the percentage error in lifetime does not rise as quickly as the error in the communication cost estimate, which shows that our algorithm is not adversely affected by this error.

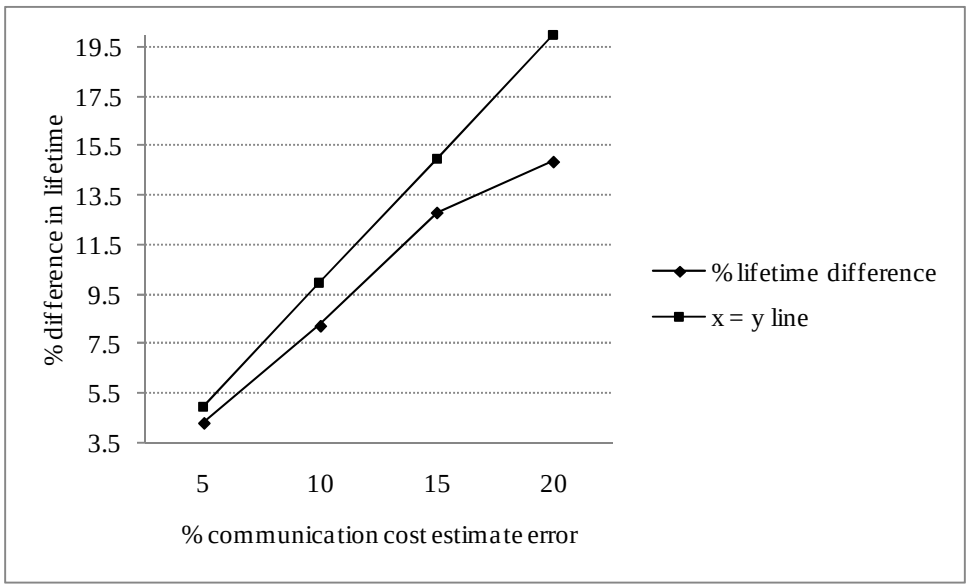


Figure 2.13 Impact of the communication cost estimate on actual lifetime

2.8 Distributed implementation

It is highly desirable to have a distributed implementation of any routing algorithm. We now discuss how the aforesaid routing strategies can be implemented in a distributed setting. Primarily, there are two models of developing distributed implementations. In the purely distributed model, the algorithm is completely based on message passing. That is, the algorithm's implementation is only based on passing messages and when it terminates, each node knows the neighbor to whom it must forward packets so that the algorithm rules are followed. The other approach is to make each node aware of the global state by enabling each node to advertise its local state to the entire network so that each node can locally compute the route based on the global state. In this case, the source will know the entire route to reach the destination, and may add the information about the route in the packet itself.

2.8.1 Collecting global state information

Multiple message passing and the convergence latency involved in the purely distributed scheme consumes time and energy for each run of the algorithm and hence may not be desirable in a wireless network. For this reason and motivated by the fact that real world protocols such as OSPF [68] use similar strategies, we use the second approach. We construct a rooted spanning tree on the wireless network graph and the global information is collected as follows: all nodes at level *j* are expected to send their energy level information to their parents, before the nodes at level *j*-1 start transmitting their information to their parents. We start with the nodes at the lowest level on the tree sending their energy information to the parents on the tree. This provides us a bound on the number of links used (and thus the amount of energy spent) for collecting the global information. In other words, each node sends all the information it has to its parents all at once. Once the complete network information is collected at the root, the root then transmits this information to all the nodes using the links on the spanning tree. It is important to note that we only collect global information periodically (in our experiments we use a time based periodicity, i.e. each node advertises this information once every 360 seconds) and not after every single message transmission.

There is additional latency involved in this approach of collecting global state information. In a wireless network using 802.15.4 wireless nodes, we can have a data rate of 250kbps [65]. If we assume that the energy level and the node ID for each node in the network would require 32 bits of data to encode and suppose we have a 1000 node sensor network, we would still be able to encode all the information within 32000 bits. All this information can still be transmitted within 1 second over a single link in a

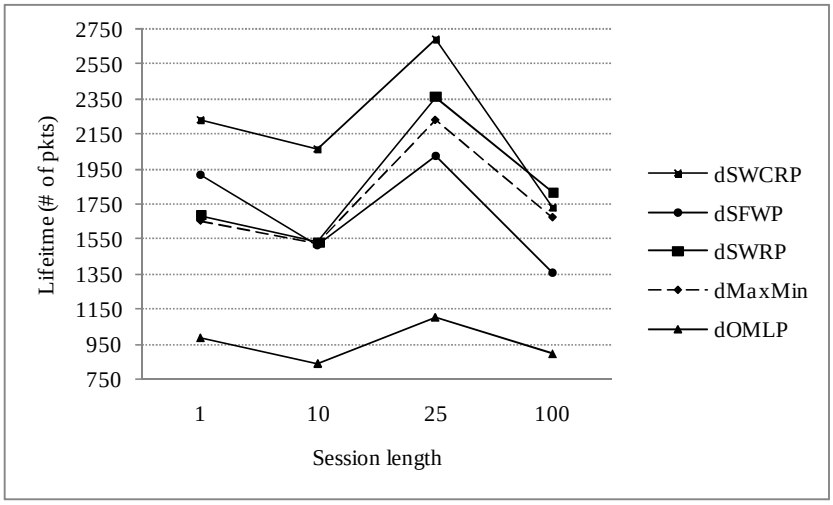
network with a data rate of 250kbps. Depending on the height of the spanning tree, we must still be able to complete the global update process within at most tens of seconds. While this is only a rough estimate and factors such as interference in the wireless network should be taken into account, we argue that since the global information is collected only periodically, it will not prove to be a bottleneck for the operation of the routing algorithms in a distributed setting.

To compute the energy spent during the global update process, we calculate the average energy e_{av} required to transmit a packet across a link in the network, and use twice that number as the total amount of energy required for the global information collection. In other words, we subtract $2e_{av}$ units of energy at each node in the network after each global update process. For purposes of evaluation, we assume that the average packet size used in the expressions for calculating the energy required per packet transmission (mentioned in Section 2.3) is based on continuous transmission for one second.

2.8.2 Performance of distributed implementations

The performance of the distributed versions of the algorithms with varying transmission radius and varying node densities are presented in Figures 2.14 – 2.16. The following acronyms are used for the graphs in this discussion of distributed algorithms: dSWRP – distributed Shortest Widest Residual Path, dMaxMin – distributed max-min zP_{min}, dOMLP – distributed Online Maximum Lifetime heuristic; dSFWP – distributed Shortest Fixed Width path, and dSWCRP – distributed Shortest Width Constrained Residual Path.

From these results, we note that the distributed implementations of the shortest widest path and its derivatives give rise to better network lifetimes than the distributed implementations of OML and max-min zP_{min}. Therefore, one may be able to infer that the distributed implementation of the shortest widest path and its derivatives are less sensitive to the lack of up-to-date global energy level information than the distributed versions of OML and max-min zP_{min}. While the average residual energy level in the network seems to be lower for the proposed algorithms than OML, it has to be noted that the proposed algorithms forward lot more packets in the network than OML (as can be observed from the graphs) and therefore are left with a smaller residual energy.



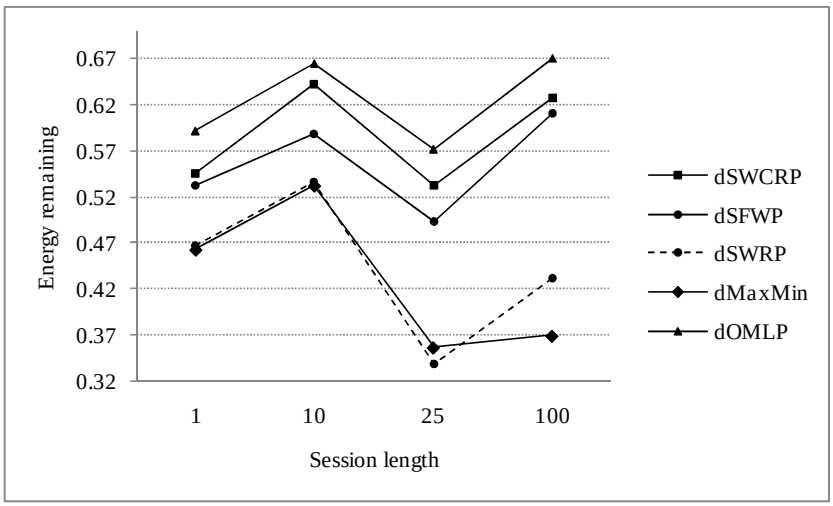


Figure 2.14 (a): Lifetime vs session length (b): Energy remaining vs session length (distributed algorithm)

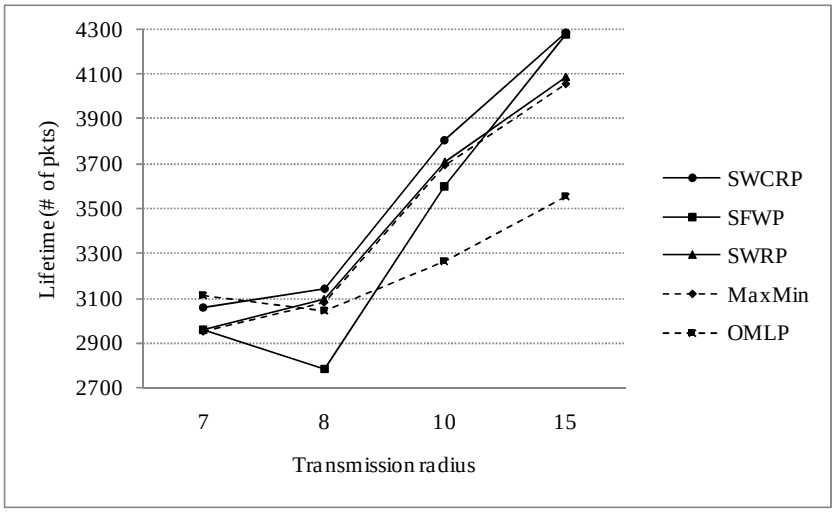


Figure 2.15 (a): Lifetime vs transmission radius

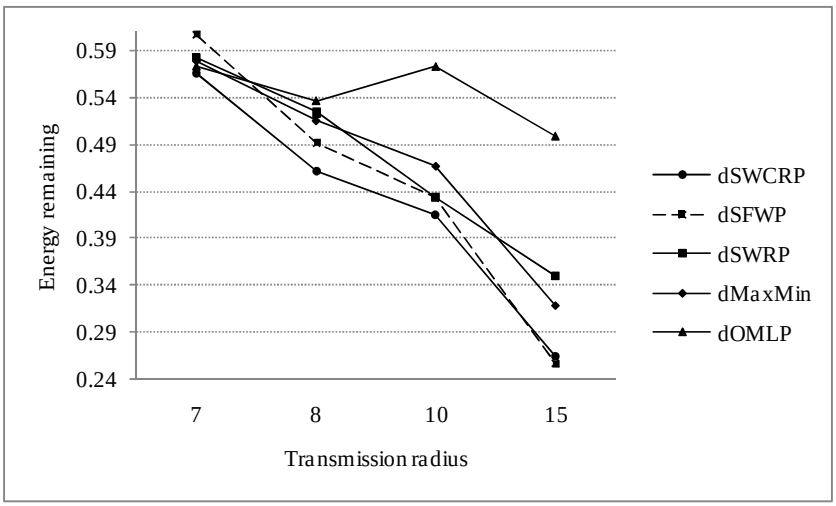
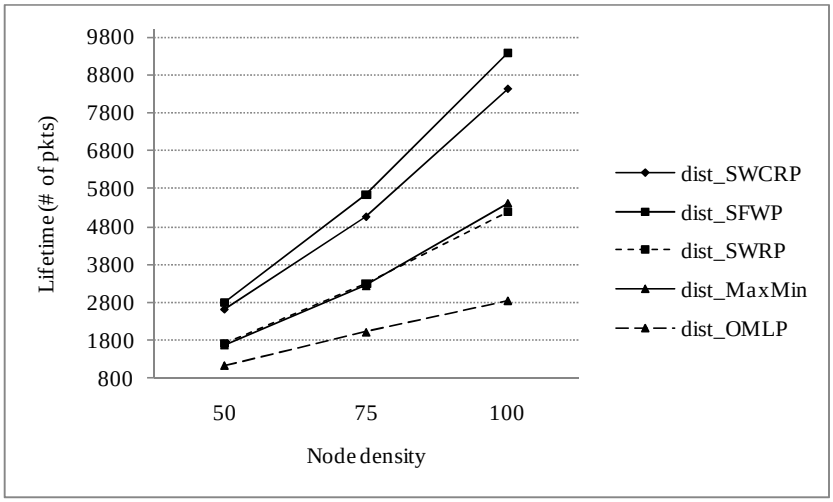


Figure 2.15 (b): Energy remaining vs transmission radius (distributed algorithm)



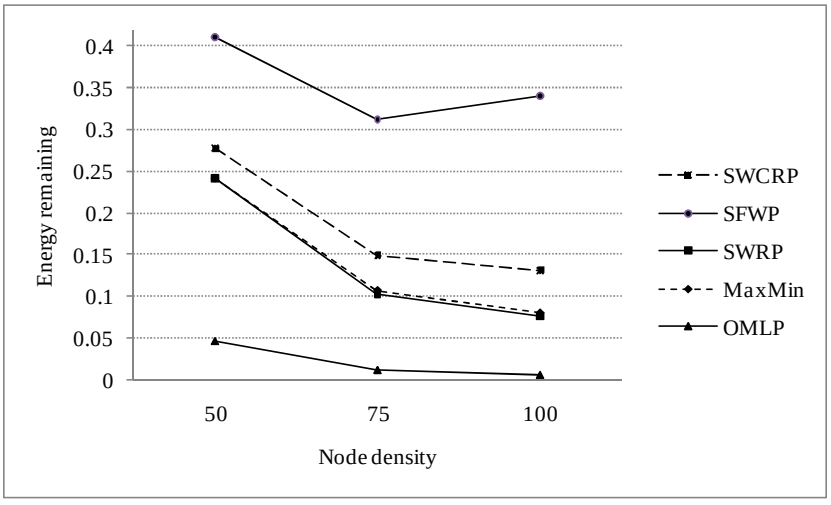


Figure 2.16 (a): Lifetime vs node density and (b): Energy remaining vs node density (distributed algorithm)

2.9 Hop-by-hop routing for multi-metric shortest paths

In a pioneering work, Sobrinho [78] discusses the implications of the concept of isotonicity for various path problems. He introduces the notion of a path algebra which includes a binary operator which takes as input a path and an edge or two paths, and a relation operator which acts as a ordering function between two given paths. Isotonicity, stated simply, is the property of certain types of paths where the application of the binary operator on two given paths (for example by adding an edge to both of them) maintains the order relation between them.

In the context of the widest path problem, it has been shown in [78] that the shortest widest path is non-isotonic. In other words, it is not possible to find a hop-by-hop distributed algorithm for finding the shortest widest path. It would require at least two passes for any distributed algorithm if it wishes to find the shortest widest path. Since the shortest width constrained path as well as the shortest fixed width path depend on finding the widest path as a preliminary step, it follows that they would also be non-isotonic. The reference [78] also proves some types of paths to be isotonic – the ubiquitous shortest path problem as well as the widest shortest path, among others.

2.10 Summary

The shortest widest path algorithm and two of its derivatives have been proposed in this chapter for performing online energy aware routing in wireless networks. All of the proposed algorithms have been shown to improve the network lifetime when compared with the best solution known in the literature. By exploiting a simple relationship between the energy consumed along a path and the residual energy of the bottleneck

nodes along the path, we have presented a solution space where we have a better chance of prolonging the network's lifetime.

Chapter 3

3 Throughput of wireless networks

3.1 Introduction

The throughput observed in a single path in a multiple hop wireless network can be no more than a third of the single hop bandwidth under the standard radio model [47]. There are some applications where one might require a higher end-to-end throughput than that available through the use of a single path. For example, a city-wide wide area network implemented as a multi-hop static mesh network, may benefit from higher throughputs. The traditional way to achieve higher throughputs in wired networks is to use multiple paths in parallel so as to improve the aggregate bandwidth. While a similar approach can be adopted for wireless networks too, wireless networks suffer from the additional challenge that in most network topologies, the discovered paths may lie close to each other. Consequently, packet transmissions in one path may interfere with transmissions taking place in other paths in the path set leading to a significant reduction in the overall throughput [87].

This reduction in throughput in wireless networks can be avoided by appropriate path selection combined with careful packet transmission scheduling. It has been noted in the literature [47] that the maximum possible throughput equaling the single hop bandwidth can be achieved by using three non-interfering paths. Hence having multiple paths which do not interfere with each other is ideal. However, we show in this chapter that this problem is the same as the problem of finding chordless cycle containing a pair of vertices in a graph, which is actually NP-Complete [10]. We then turn our attention

towards finding path sets in static wireless networks which would provide the same level of aggregate throughput as non interfering paths while at the same time permitting interfering links.

Our contributions are: (a) we demonstrate that it is possible for a set of paths between source '*s*' and destination '*t*' with some interference between them to provide high aggregate throughputs provided the interfering edges among the paths follow certain favorable patterns; we present a combinatorial approach for finding such paths in a wireless network. (b) we extend our approach to scenarios involving multiple *s*-*t* pairs and show that the proposed approach can improve the throughput in such scenarios too. (c) our combinatorial approach can also provide a straightforward mechanism for scheduling the transmissions at various links and finally, (d) the computation of the transmission schedule is shown to be amenable to a distributed implementation under the proposed approach. Preliminary results of this work has been presented in [61].

The rest of the chapter is organized as follows. Section 3.2 discusses literature that is relevant to the proposed work. Section 3.3 presents the notations and conventions of a timing diagram termed the *Wireless Pipeline Scheduling Diagram* which is a visual tool to aid in the understanding of the material. Section 3.4 discusses the impact of the hop length on the aggregate throughput when we are able to discover paths which do not interfere with each other. Section 3.5 presents the relationship between the patterns of interference that can exist between multiple paths and the achievable aggregate throughput rates. Even when paths interfere, we show that there exist certain interference patterns (which we call *non-destructive* interference) which can still

support high aggregate throughputs. Using this idea, in Section 3.6 we develop an algorithm for finding interference aware path sets between a *s*-*t* pair. In Section 3.7, we extend our approach for finding paths under the multiple *s*-*t* pairs case, which is more typical in static wireless mesh networks. In Section 3.8, we present some performance evaluations and in Section 3.9, we present our conclusions and directions for future research.

3.2 Related Work

Wu and Harms [87] identify the issue of interference between multiple paths on a wireless network and show that there is a loss in aggregate throughput due to interference. Given a wireless network, a source *s* and a destination *t*, previous results related to the study of throughput performance using multiple paths in a wireless multihop network can be categorized into three classes. (a) the first class of solutions try to find multiple node disjoint paths between *s* and *t* such that there are no edges connecting two vertices belonging to different paths (we call such edges *interpath links*), (b) solutions under the second class use a centralized multi-commodity flow based linear programming (LP) formulation to exhaustively search and determine the maximum achievable throughput with interpath links (c) the third class of solutions adopt a combinatorial approach to find multiple paths which can support a high throughput even with interpath links.

Techniques presented under class (a) do not consider the case of simultaneous multiple *s*-*t* transmissions. Solutions suggested under class (b) neither give the paths nor provide a schedule for transmissions, but nevertheless are important as they establish the bounds for throughput against which other schemes could be compared.

While schemes under classes (b) and (c) consider multiple *s-t* transmissions, approach (c) is combinatorial and hence could possibly result in distributed solutions.

Hu et al. [18], Saha et al. [72] and Jones et al. [39] discuss techniques which fall under class (a). Hu et al. [18] address this issue by creating a forbidden region around the first path and finding a second path outside this forbidden region. While this is a good strategy for finding interference free paths, their strategy does not work if there are no interference free paths in the network. Saha et al. [72] propose using directional antennas to reduce the interference between paths and thereby improve the throughput of multipath routing. Jones et al. [39] introduce an interference metric to assess the quality of disjoint paths and use this metric to find high quality multiple paths which can improve the throughput. Saha et al. [72] and Jones et al. [39] both presume that interference is inherently destructive and should be avoided. Consequently, they favor node-disjoint paths with no interpath links which could possibly preempt some *s-t* pairs from enjoying a high throughput transmission when several such node pairs wish to communicate simultaneously. In our approach, by accommodating certain interpath links which do not lead to interference (which we call as *nondestructive edges*), we can potentially find multiple paths with high throughput values in more scenarios than in [72] and [39].

Jain et al. [36] and Buragohain et al. [12] discuss strategies which fall under class (b). Jain et al. [36] show that finding optimal throughput for multiple *s-t* pairs is NP-Hard, and it is NP-Hard even to approximate the optimal throughput. Additionally, they provide a systematic analysis of the achievable lower and upper bounds for the throughput using a multi-commodity linear programming approach. Buragohain et al.

[12] improve on the work by Jain et al. [36] by proposing a node based LP formulation combined with a node ordering technique, which allows them to achieve a 1/3 approximation of the optimal throughput. However, their technique too, unlike ours, does not provide the actual schedule for achieving the throughput owing to the LP formulation.

The approach proposed in this chapter falls under class (c). A relevant work under this class is presented in Liaw et al. [50]. Liaw et al. [50] determine the maximum achievable throughput for a given wireless network by computing all possible shortest paths of a given length *k* and independent paths for these shortest paths in such a way that for a given shortest path an independent shortest path is a vertex disjoint path. Clearly, this approach of computing the maximum throughput will have exponential complexity. They further make an important observation to indicate that the maximum throughput can be achieved when spatial-reuse with respect to the source is very high, where spatial-reuse is defined as the maximum number of nodes (including the source) that can transmit simultaneously without causing interference. In this chapter, we generalize this spatial-reuse concept and show that by having additional relaxations we can improve the throughput considerably. We provide a polynomial time heuristic to find multiple paths for a single *s*-*t* pair as well as single paths for multiple *s*-*t* pairs that can achieve high aggregate throughputs. We also give a simple method to determine the best transmission schedules to achieve these throughputs.

3.3 System Model

We adopt a system model with the following features which is similar to the one used in [18]:

a) The transmission time in the medium is divided into slots and the nodes are assumed to have synchronous clocks. Clock synchronization among the network nodes can be achieved by using strategies similar to the one presented in [56].
b) A node can either transmit or receive once in a time slot. The width of the slot is sufficient enough for the largest packet to be transmitted and received.
c) No node can transmit and receive simultaneously.
d) A node cannot "hear" more than one neighbor at once – in other words, if more than one neighbor of any node transmit in the same time slot, the node loses all the packets.
e) The internal nodes in the path forward the packets they receive without any delay

To aid the purpose of analysis, we use a timing diagram that we call the *Wireless Pipeline Scheduling Diagram* (WPSD) to investigate the theoretical capacity limits of employing multi-path routing in a wireless network under different scenarios. An example of such a timing diagram is shown in Figure 3.1, where it is shown that the maximum possible throughput for a single path is only a third of the available bandwidth. Another example of the wireless pipeline scheduling diagram is shown in Figure 3.2, where we show paths with equal hops. In this case, since the destination 5 receives a new packet during every time slot (after receiving its first packet), the maximum aggregate throughput is achieved.

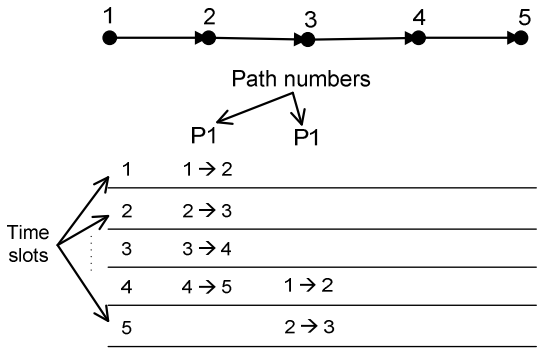
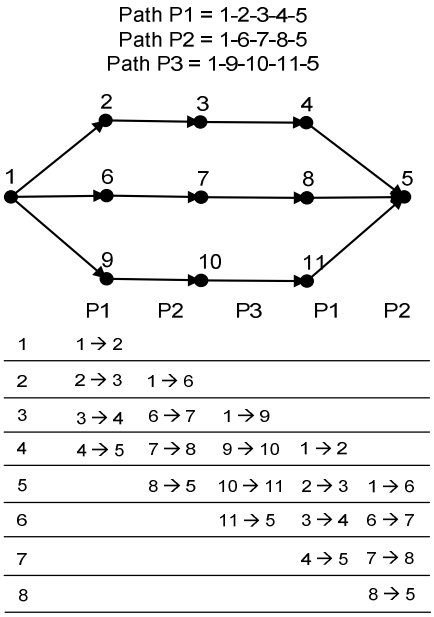


Figure 3.1: Wireless Pipeline Scheduling Diagram for a path P1.

3.4 Impact of path length on achievable aggregate throughput

We now present a formal analysis of the relationship between the path length (in terms of hops) and the achievable aggregate throughput when we have multiple paths which do not interfere with each other. We later use this in our algorithm for finding multiple paths for a given *s*-*t* pair. For simplicity of analysis, we consider the two paths case first. We assume that packets are injected into all the paths in the multi-path set at a constant rate. Let λ be a constant denoting the inter-packet interval time, also referred to as the periodicity. Here λ is the inter-packet arrival time with respect to a single path. (It must be noted that our definition of λ differs from the statistical average arrival rate used commonly for Poisson processes). The nodes along a path relay the packets that they receive in the time slot following the reception without any delay. Allowing the intermediate nodes to delay their forwarding could have an adverse effect on end-to-end delay experienced by the subsequent packets sent along the path.



1	1 → 2				
2	2 → 3	1 → 6			
3	3 → 4	6 → 7	1 → 9		
4	4 → 5	7 → 8	9 → 10	1 → 2	
5		8 → 5	10 → 11	2 → 3	1 → 6
6			11 → 5	3 → 4	6 → 7
7				4 → 5	7 → 8
8					8 → 5

Figure 3.2: WPSD shows that three paths can provide the maximum possible throughput.

3.4.1 Two paths

Let h_1 and h_2 be the number of hops in paths P_1 and P_2 between source s and destination t. The paths P_1 and P_2 are called non-interfering paths if there are no interfering edges[8] between P_1 and P_2. It is to be noted that though P_1 and P_2 may not have any interfering edges, since they may be of unequal hop-lengths, collisions can still occur at the destination t. Let t_1 and t_2 represent the time slots when the first packets are respectively injected into paths P_1 and P_2. Let the packets traveling along P_1 be designated as type I and packets traveling along P_2 be designated as type II.

Lemma 3.1: At the destination node, no collisions occur between packets of type I and type II if t_1 and t_2 are chosen such that $(h_1 + t_1) \bmod \lambda \neq (h_2 + t_2) \bmod \lambda$.

Proof: Based on the parameters given, we can see that the destination t receives packets of type I at times $S_1 = \{(h_1\text{-}1)+t_1, (h_1\text{-}1)+t_1+\lambda, (h_1\text{-}1)+t_1+2\lambda, (h_1\text{-}1)+t_1+3\lambda, \ldots\}$ and it

[8] An edge e is considered an interfering edge if its two endpoints lie on P_1 and P_2, but the edge e itself belongs to neither P_1 nor P_2.

receives packets of type II at times S_2 = {(h_2-1)+t_2, (h_2-1)+t_2+λ, (h_2-1)+t_2+2λ,(h_2-1)+t_2+3λ,…}. We can ensure that there are no collisions at the destination if S_1 and S_2 have no elements in common. Given two sequences S_a = {a + λ, a + 2λ, a + 3λ, …} and S_b = {b + λ, b + 2λ, b + 3λ,…} we can ensure that S_a and S_b do not share a common element if a mod λ ≠ b mod λ. The proof is as follows: if a mod λ ≠ b mod λ then (a + k_1λ) mod λ = a mod λ ≠ (b mod λ) = (b + k_2λ) mod λ ∀ k1, k2 ≥ 0. Substituting (h1 – 1) + t1 for a and (h_2 – 1) + t_2 for b, this condition translates as (h_1 + t_1) mod λ ≠ (h_2 + t_2) mod λ. That is the condition for scheduling time slots t_1 and t_2 so that there are no collisions at the destination. ■

While the primary condition for achieving collision free transmissions along the two paths has been stated, we make some additional observations:

1. For a given set of paths λ should be made as small as possible to achieve the maximum aggregate throughput. While the example discussed using Figure 3.1 has made clear that λ cannot be less than 3, it is possible (and in fact desirable) that λ is exactly 3.
2. The two initial time slots t_1 and t_2, should both be as small as possible so the transmission can start early.
3. Also, t_1 cannot be the same as t_2, since it would mean that we are transmitting different data simultaneously to two neighbors during the same time slot.

In the two paths case, we can find suitable values for t_1 and t_2 for all values of h_1 and h_2 such that λ is exactly 3. Let r_1 and r_2 represent the remainders after dividing h_1 and h_2 by 3 (3 represents the minimum value we seek for λ). There are only three possible values for each of r_1 and r_2, representing a total of nine combinations.

r_1	r_2	t_1	t_2	$(r_1+t_1) \bmod 3$	$(r_2+t_2) \bmod 3$
0	0	1	2	1	2
0	1	1	2	1	0
0	2	1	3	1	2
1	0	2	1	0	1
1	1	1	2	2	0
1	2	1	2	2	1
2	0	3	1	2	1
2	1	2	1	1	2
2	2	1	2	0	1

Table 3.1 Suitable t_1 and t_2 values for various values of r_1 and r_2. Parameters r_1 and r_2 represent the remainders of dividing the hop lengths of the two paths by 3.

We give appropriate values for t_1 and t_2 for each case in Table 3.1. We see that the last two columns show that $(r_1 + t_1) \bmod 3 \neq (r_2 + t_2) \bmod 3$ for all these cases, thus satisfying the condition in Lemma 3.1. Figure 3.3 shows an example where we have unequal hop paths, but the aggregate throughput is still two thirds of the channel bandwidth as $\lambda = 3$ (and the destination receives a packet twice every three time slots). Here we see that h_1 is 3 and h_2 is 6, and thus both r_1 and r_2 are 0. We note that $t_1 = 1$ and $t_2 = 2$, where t_1 and t_2 are the first time slots during which sender 1 transmits its packets to the neighbors in paths P_1 and P_2 respectively.

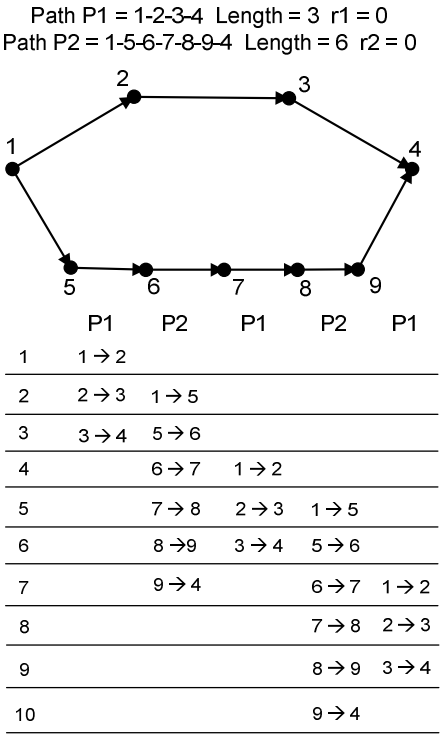
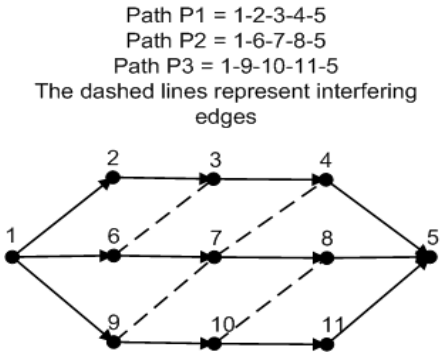


Figure 3.3 Schedule for unequal hop paths with equal remainders r_1 and r_2.

In the three path case too, given the remainders r_1, r_2 and r_3 (for paths P_1, P_2 and P_3), it is possible to determine the smallest periodicity λ as well as time slots t_1, t_2, t_3 (all as small as possible) such that there are no collisions at the receiver. The following two lemmas provide the necessary guidelines for achieving this.

Lemma 3.2.1: Let us assume we have three paths with hop counts h_1, h_2 and h_3. There are no collisions at the destination if $(h_1 + t_1) \bmod \lambda \neq (h_2 + t_2) \bmod \lambda \neq (h_3 + t_3) \bmod \lambda$.

Proof: Follows from Lemma 1.■



	P1	P2	P3	P1	P2
1	1 → 2				
2	2 → 3	1 → 6			
3	3 → 4	6 → 7	1 → 9		
4	4 → 5	7 → 8	9 → 10	1 → 2	
5		8 → 5	10 → 11	2 → 3	1 → 6
6			11 → 5	3 → 4	6 → 7
7				4 → 5	7 → 8
8					8 → 5

Figure 3.4 The set of paths allow maximum throughput despite interpath interference

Lemma 3.2.2: Let r_1, r_2 and r_3 be the remainders after dividing the hop counts h_1, h_2 and h_3 by 3. The condition in lemma 3.2.1 is satisfied for $\lambda = 3$ only if $r_1 = r_2 = r_3$ or if $r_1 \neq r_2 \neq r_3$. If $r_1 = r_2 \neq r_3$, then the smallest value of λ which satisfies Lemma 3.2.1 is 4.

Proof: A simple enumeration of all the possible time slots t_1, t_2 and t_3 proves the lemma.■

r_1	r_2	r_3	P	t_1	t_2	t_3	r_1	r_2	r_3	P	t_1	t_2	t_3	r_1	r_2	r_3	P	t_1	t_2	t_3
0	**0**	**0**	**3**	**1**	**2**	**3**	1	0	0	4	1	3	4	2	0	0	4	1	2	4
0	0	1	4	1	2	3	1	0	1	4	1	3	4	**2**	**0**	**1**	**3**	**2**	**3**	**1**
0	0	2	4	1	3	2	**1**	**0**	**2**	**3**	**1**	**3**	**2**	2	0	2	4	1	2	3
0	1	0	4	1	2	4	1	1	0	4	1	2	4	**2**	**1**	**0**	**3**	**1**	**3**	**2**
0	1	1	4	1	2	3	**1**	**1**	**1**	**3**	**1**	**2**	**3**	2	1	1	4	1	3	4
0	**1**	**2**	**3**	**1**	**2**	**3**	1	1	2	4	1	4	2	2	1	2	4	1	3	2
0	2	0	4	1	2	3	**1**	**2**	**0**	**3**	**1**	**2**	**3**	2	2	0	4	1	3	2
0	**2**	**1**	**3**	**2**	**1**	**3**	1	2	1	4	1	2	4	2	2	1	4	1	2	4
0	2	2	4	1	2	4	1	2	2	4	1	2	3	**2**	**2**	**2**	**3**	**1**	**2**	**3**

Table 3.2: Suitable t_1, t_2, and t_3 values for various values of r_1, r_2, and r_3

As in the two paths case, it is possible to enumerate the t_1, t_2 and t_3 values for various combinations of r_1, r_2 and r_3 (shown in Table 3.2).

Note: If we decide to choose no more than two paths between a *s*-*t* pair, we would seek a periodicity of $\lambda = 3$, 4 or 5 to give rise to a bandwidth utilization of $2B/3$, $2B/4$, and $2B/5$ respectively, all of which are superior to the single path bandwidth of $B/3$. If we decide to choose three paths, we would seek a periodicity of no more than $\lambda = 4$ giving an aggregate bandwidth of $3B/4$ which is superior to the maximum possible two path bandwidth of $2B/3$.

3.5 Non destructive interference patterns

The problem of finding two non-interfering paths between a source and destination is the same as the problem of finding a chordless cycle containing two given vertices in a graph, which has been shown to be NP-Complete [10]. Our goal in this section is to show that there exist interfering paths which achieve the same aggregate throughput as non-interfering paths.

Figure 3.4 shows a topology with three paths having mutual interference (the dashed lines represent the interfering edges). However, as the example illustrates, the path set still provides the maximum possible aggregate throughput equal to the single hop bandwidth *B*. The reason for this is that the paths exhibit *non-destructive interference,* i.e., the nodes on the two ends of all the interfering edges receive and transmit their packets simultaneously, thereby avoiding collisions. Such interfering edges are referred to as *non-destructive edges*. A set of two or three vertex disjoint paths are said to have non-destructive interference if it is possible to schedule packet

transmissions along the paths without any collisions for a desired periodicity λ despite the presence of interfering edges. In other words, a set of vertex disjoint paths are said have non-destructive interference if all the interfering edges among the paths are non-destructive in nature.

Lemma 3.3: Let e be an interfering edge between nodes v_1 and v_2 which lie along paths P_1 and P_2. Let the nodes v_1 and v_2 be k_1 and k_2 hops away from the source s. The edge e will be non-destructive if :

(i) $\left| ((k_1 + t_1) \; mod \; \lambda - (k_2 + t_2) \; mod \; \lambda \right| \neq 1$, and

(ii) $\left| ((k_1 + t_1) \; mod \; \lambda - (k_2 + t_2) \; mod \; \lambda \right| \neq \lambda - 1$.

Proof: We first consider the case where the interfering edge e can cause a problem with the packet transmissions on the two paths. Essentially, if node v_1 transmits its packet at time slot t and node v_2 receives its packet during the same time slot, then there will be a collision at v_2 and v_2 will not receive its intended packet. Similarly, if v_2 transmits a packet at time slot t and v_1 receives a packet during the same time slot, then v_1 will not receive any packet. Hence we see that the time slots during which v_1 and v_2 receive their packets must not differ by 1.

We can see that node v_1 receives its packets at time slots $S_1=\{t_1+(k_1-1)+ \lambda, t_1+(k_1-1)+2\lambda, t_1 + (k_1-1)+3\lambda,\dots \}$ and node v_2 receives its packets at time slots $S_2 = \{t_2+(k_2-1)+\lambda, t_2+(k_2-1)+2\lambda, t_2 + (k_2-1)+3\lambda,\dots\}$. Following an argument similar to the proof in lemma 3.1, we will have no time slots with unit difference in sequences S_1 and S_2 if we can ensure that $\left| (k_1 + t_1) \bmod \lambda - (k_2 + t_2) \bmod \lambda \right| \neq 1$ and $\left| (k_1 + t_1) \bmod \lambda - (k_2 + t_2) \bmod \lambda \right| \neq \lambda - 1$. ■

We now state the following theorem which directly follows from Lemma 3.3.

Theorem 3.1: Two (resp. three) vertex disjoint paths have non-destructive interference between them for a given periodicity λ if all the interfering edges among the two (resp. three) paths are non-destructive with respect to λ and if Lemma 3.1 (resp. Lemma 3.2.1) is satisfied.■

As an example, in Figure 3.4, let us consider the (interfering) edge between nodes 6 and 3. Node 6 receives its packets at the following time slots: {2, 5, 8, ...} i.e. it has t_1 = 2 and k_1 = 1. We note that node 3 also has the same schedule for receiving packets: {2, 5, 8, ...}and in this case we see that t_2 = 1 and k_2 = 2. In effect, the packets arriving and leaving from these two nodes never collide with each other. Also, the three path lengths are the same, and hence lemma 3.2.1 is satisfied for periodicity λ = 3.

3.6 Computing interference aware multi-path sets for a single *s*-*t* pair

Jain et. al [36] have shown that computing the multi-path set that gives the highest throughput between a given source and destination is NP-Hard [36]. In this section, we describe a polynomial-time heuristic for finding a good set of paths between a source *s* and destination *t* that can give a high throughput. We incorporate our knowledge of the non-destructive interference patterns into this heuristic. The heuristic works by first discovering a path between the given *s*-*t* pair. Any path between *s* and *t* would be fine, though in this chapter we use the shortest path. the heuristic then finds another path between *s* and *t* which has 'good' interference awareness with respect to the first path.

The heuristic works as follows. We first find the shortest path from the source to destination. To find a second path, we mark all the nodes in the graph with labels which

indicate which nodes are their neighbors in the first path. Using these labels, we compute a suitable interference aware path and a transmission schedule which guarantees that the interpath links are non-destructive.

We model the given wireless network as a graph $G = (V, E)$, where V is the set of all nodes and E is the set of all edges between the nodes. Let P_1 be the first path from source s to destination t in graph G. For each internal node $n \in P_1$, let $d(n)$ be the number of hops from s to n along P_1. Let $neighbors(n)$ represent the set of nodes in the graph which are neighbors of n in G which do not belong to P_1. We now label all the nodes $v \in G$, $v \notin P_1$ with a label set $L(v)$ as follows: for each node n, for each node $v_1 \in neighbors(n)$, $L(v_1) = L(v_1) \cup d(n)\}$. Since the same node in graph G could be a neighbor of multiple nodes in path P_1, it is evident that L is a set of numbers rather than a single value. Once we have completed labeling all the nodes in G, we remove (temporarily) all the internal nodes of P_1 from graph G. On the remaining graph, a *good interference aware path* from s to t will be a path P_2 whose node labels obey the following property: there is *at least one* feasible pair of time slots (t_1, t_2) such that $\forall$ nodes $n_2 \in P_2$, all edges incident on n_2 are non-destructive. Let $d_2(n_2)$ be the number of hops from s to n_2 along P_2. Now let Δt represent the time slot difference $t_1 - t_2$. The condition for all edges to be non-destructive is as follows: for each element $l \in L(n_2)$, $|d_2(n_2) - l + \Delta t| \bmod \lambda \neq 1$ and $\neq \lambda - 1$ - this directly follows from Lemma 3 (but remembering that Δt cannot be 0).

The Algorithm InterferenceAwarePath is a polynomial time algorithm that finds a good interference aware path with respect to a path P_1 between a source s and

destination *t*. We note here that determination of a common value of λ for the two paths, a suitable Δ*t*, and the fact that each intermediate node forwards a packet in the next time slot after reception collectively define the transmission schedule needed to realize a high throughput. It must also be observed that the algorithm uses a greedy approach and is hence only a heuristic.

Algorithm InterferenceAwarePath (*s, t, P_1*)
// Input – Source *s*, destination *t*, path P_1 between *s* & *t*
// Output – A good interference aware path P_2 between *s* and *t* along with λ (the schedule for transmission).
// h(*P*, *s*, *u*) – number of hops from source *s* to node *u* in path *P*
1. For each internal node *u* U P_1, mark all nodes *n* which are neighbors of *u* with the label h(P_1, s, *u*)
2. Temporarily remove all internal nodes from the graph *G*
3. Initialize ( )
4. While |*Q*| > 0 and $t \notin P_2$
5. for λ = 3 to 6
6. for Δ*t* = − (λ−1) to (λ−1)
7. *u* ← getBestFeasibleSet(*Q*)
8. P_2 ← P_2 U {*u*}
9. for all neighbors *w* of *u*
10. if |Δt + (h(*P*,*u*,*s*) − label(*w*))| != 1 and |Δt + (h(*P*,*u*,*s*) − label(*w*)) | != λ −1
11. feasibleSet(*w*) = feasibleSet(*w*) U {Δt}

getBestFeasibleSet (Q)

return *v* where *v* ∈ *Q* has max. |feasibleSet(*v*)|

Initialize ( )

1. For each node *v* ∈ *G*
2. feasibleSet(*v*) = {− (λ−1), − (λ −2),…, λ−2. λ−1}
3. *Q* ← *Q* U {*v*}

Figure 3.5 Interference aware algorithm for single *s*-*t* pair

3.7 Interference aware paths for multiple *s*-*t* pairs

We now discuss the case of finding good interference aware paths when we have multiple source destination pairs. Let us say we have a request set $R = \{(s_1, t_1), (s_2, t_2),\ldots, (s_k, t_k)\}$ where we need to find multiple paths between (s_i, t_i) $\forall$ $i \in [1, k]$ such that the aggregate throughput across all these paths is maximized. We consider long

lived flows (duration of flow is infinite) between all the source destination pairs, which is likely to be true in the back-bone of a wireless mesh network.

We construct a set of paths called the *Concurrent Transmission Paths Set* (CTPS) such that the source nodes along the paths in a given CTPS can be scheduled for concurrent transmissions. We construct different CTPSs so that the union of all the CTPSs supplies each *s*-*t* pair in *R* with an individual path such that the chosen paths have a high aggregate throughput.

3.7.1 Diminishing Returns

Suppose we wish to construct the CTPS for the request set $R = \{(s_1, t_1), (s_2, t_2),\ldots,(s_k,t_k)\}$. To begin with, let us consider the pair (s_1, t_1) and discover a path P_1 between these nodes. This path is added to the CTPS. Based on this path, interference aware paths between other node pairs are discovered and added to the CTPS. For this reason, path P_1 is called the *seed*. The keen reader will observe that different seeds may lead to different CTPSs and hence a CTPS is always identified based on its seed as CTPS(P_1).

Consider CTPS(P_1) =$\{P_1\}$. Let us suppose the path P_i between (s_i, t_i) has the highest throughput value (minimum λ) with respect to P_1. While one may be tempted to add P_i to CTPS(P_1), the *principle of diminishing returns* must be applied to determine if this can be done. The *principle of diminishing returns* states that path P_i can be added to a CTPS provided the overall throughput of the CTPS with the addition does not get reduced to a *lower* value than what it would have been without adding it. As an example, if CTPS(P_1) = $\{P_1\}$ and the path P_i with the highest throughput value results

in a λ =6. In this case, adding the path P_i to the CTPS may not help since we may have only two paths in the resulting CTPS and scheduling transmission along them sequentially will result in the same throughput as scheduling them concurrently. However, if the path P_i results in a λ = 5, it may be worth adding the path to the CTPS since concurrent scheduling of the paths will have a higher throughput than sequential scheduling. If path P_i has been added to the CTPS(P_1), we say that the pair (s_i, t_i) has been serviced.

We continue this process until we arrive to the stage wherein we could no longer add any path to CTPS(P_1). Now, we choose a path P_k between an unserviced (s_k, t_k) as the seed and build another CTPS to serve the other un-serviced pairs and so on. We keep repeating the process until all the requests in R have been satisfied.

While reducing the number of CTPSs may help in increasing the overall throughput at times, it may not always be the case. For example, let there be ten *(s, t)* pairs. One possible arrangement of CTPSs might be {(4, 3), (4, 3), (3, 3)}, where the first number in the ordered pair represents the number of paths in the set, and the second number represents the value of λ. A second possible CTPS arrangement might give us {(5, 5), (5, 5)}. In the first case, overall throughput of (B/3+B/3+B/3)/3 = B/3 is obtained, while in the second arrangement we only get (B/5+B/5)/2 = B/5. In other words, the best sets of CTPSs are those that maximize the weighted throughput.

It is easy to see that given the CTPS sets for each request pair (s_i,t_i) as the seed, ascertaining the best possible set of CTPSs is still an NP-Complete problem, since it is equivalent to the set cover problem when all the λ values in the second term of the

ordered pair are equal. Consequently, we propose a greedy heuristic to find the best set of CTPSs. Let $\eta = |CTPS|/\lambda$ for each CTPS. In other words, η represents the ratio of the number of elements in the CTPS set over the λ for the set. We keep adding sets with maximum η to our collection of sets until all the requests have been satisfied.

Based on the above discussions, an algorithm for finding a good set of CTPSs given a request set *R* is outlined below.

Algorithm InterferenceAwarePath

//Input – *k* source-destination pairs $\{(s_1,t_1), \ldots, (s_k,t_k)\}$
//Output – *k* paths (one each for each source destination pair) and the schedule for transmission

//*CTPS* – Concurrent Transmission Paths Set
//*numCTPS* – number of Concurrent Transmission Sets

```
1.  numCTPS = 0;
2.  for each pair (s_i, t_i) ∉ {CTPS[0]...CTPS[numCTPS]}
3.       find shortest path P_i from s_i to t_i
4.      CTPS[numCTPS] = CTPS[numCTPS] ∪ P_i
5.      path P_x = getGoodPath (CTPS[numCTPS])
6.      while (NOT diminishReturns(CTPS[numCTPS], P_x)
7.         CTPS[numCTPS] = CTPS[numCTPS] ∪ P_x
8.         path P_x = getGoodPath (CTPS[numCTPS])
9.      numCTPS = numCTPS + 1
```

Figure 3.6 Interference aware multiple s-t pairs algorithms

getGoodPath
//input – Concurrent Transmission Paths Set with source-destination requests
//output – Path P which is interference aware with respect to the paths in the Concurrent Transmission Paths Set

```
1.  set minLambda = Infinity;
2.  for each req. pair (s_j, t_j) ∉ {CTPS[0]…CTPS[numCTPS]}
3.     λ = minimumPeriodicity(CTPS,j)
4.     if λ < minLambda
5.        minLambda = λ
6.       bestPath = P_j
7.  return P_j
```

minimumPeriodicity
//input – Concurrent Transmission Paths Set *CTPS*
//output – best periodicity using interference constraint

```
1.  for each path P_k in CTPS
2.              Let i be an internal node in P_k
```

```
3.  for each n(i) where n(i) ∈ neighbor(i) & n(i) ∉ P_i
4.      label(n(i)) = label(n(i)) ∪ hops(i, P_i)
5.  scheduleNotFound = true
6.  while (scheduleNotFound) and (λ < β)
7.      for Δt = − (λ −1) to (λ−1)
8.          find path P in graph such that all nodes
                have non-destructive interference
9.      if path P is found then
10.         scheduleNotFound = false
11.     else
12.         λ = λ + 1
```

Figure 3.7 Algorithm for finding good paths given source destination requests

diminishReturns

//input – current concurrent transmission set, and the new path to be added to the CTPS

//output – Boolean result indicating if the new path improves or worsens the overall throughput of the CTPS

1. if $|CTPS| / \lambda_1 > |CTPS \cup P_x| / \lambda_2$ then return true
2. else return false

Figure 3.8 Algorithm to compute diminishing return

The above algorithm for finding interference aware paths for multiple *s*-*t* pairs case involves a Dijkstra-like computation (with similar complexity) and this is invoked at most $O(r) \times O(r)$ times – we do this with respect to each request pair, for every other request pair. Hence the overall complexity of our algorithm is $O(r^2(E+V\log V))$ where E and V represent the number of edges and nodes in the graph, respectively. In the algorithm described above, the parameter β is bounded by the number of requests and is thus $O(r)$ and we could consider different paths as our seeds, adding a factor $O(r)$ to the complexity of the algorithm.

3.8 Performance evaluation

We evaluated the performance of our algorithm against the work done by Jain et al. [36]. In their work, the authors use the concept of a conflict graph to find links which

can be scheduled for concurrent transmission. They formulate this as a LP problem and provide a lower bound for the throughput which can be achieved using their approach. Using simulation settings previously suggested in the literature [12], we randomly place O(n) nodes on a O($\sqrt{n}$) * O($\sqrt{n}$) grid, with a transmission radius of 3.

For our first experiment, we perform a comparison for the single *s-t* pair case. We considered the effect of number of nodes in the network (network size) on the throughput of our approach as well as that of [36]. We used values of n = 25, 36, 49 and 64. We implemented the linear programming constraints specified in Jain et al. [36] using CPLEX. We implemented our algorithm using the LEDA graph library. For each network size, we took the average value of the throughput over 5 trials. The simulation results in Figure 3.5 and 3.6 indicate that our algorithm consistently provides high throughput for various network sizes. In Figure 3.5 CPLEX represents the throughput values obtained by using the algorithm in [36], while IAMP (Interference Aware Multiple Path) refers to our work.

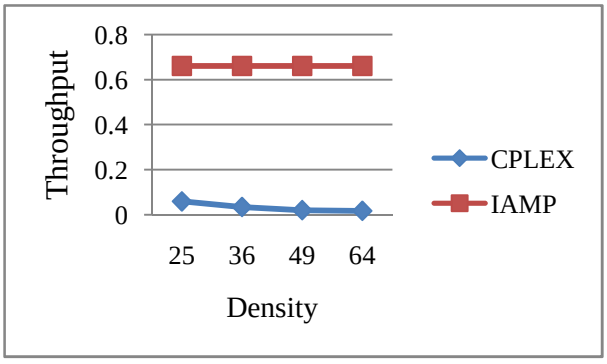


Figure 3.5 Impact of node density on throughput

For our second experiment, we perform a comparison for the multiple *s-t* pairs case. We use a 5 × 5 grid with 25 nodes placed at random and a transmission radius of 2. We simulated 10 trials with differing topologies and source destination pairs. The average

of the 10 trials was considered for 1, 2 and 3 flows (number of source-destination pairs). Then we applied our multiple *s-t* pair algorithm on this topology. The throughput for Jain et al. [36] is computed as per the constraints given in their paper. In their paper, they describe steps to compute the lower bound of the achievable throughput, which we use for comparison. The throughput value in our algorithm is simply the inverse of the smallest value of λ found by the algorithm, multiplied by the number of *s-t* paths. We were able to obtain interference aware node disjoint paths between the source and destination in all cases) = 0.66. Hence we could obtain a constant throughput of 2×1/3 (λ was 3 in our results). As shown in the graph in Figure 3.6 we can see that our approach provides much better throughput than that obtained by Jain et al. [36].

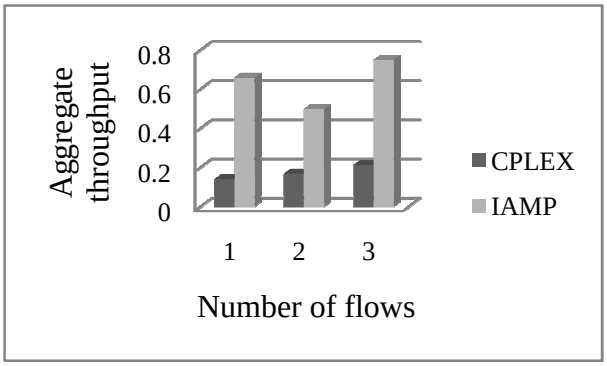


Figure 3.6 Aggregate throughput vs number of flows

3.9 Summary

In this chapter we have shown the impact of interference patterns on the throughput of wireless networks. We use a path based scheduling strategy (as opposed to the more common node or link based scheduling strategy) and this provides higher throughputs since it takes into account global information as opposed to the local information considered by the link and node based strategies. We show that the impact of

interference on multi-hop wireless networks is more due to the pattern of the interference than the number of interfering links themselves. Since energy is a concern in wireless networks, as future work we would like to combine lifetime considerations along with the aforementioned routing strategies to extend the lifetime of the network.

Chapter 4

4 Distributed algorithm for interference aware vertex disjoint paths routing

4.1 Introduction

With the introduction of multi-hop wireless services such as city-wide wide area mesh networks, it has become important to improve the throughput of wireless networks to provide good quality of service. Under the standard radio model, the multihop bandwidth can be at best a third of the single hop bandwidth [36]. Using multiple paths is one way to improve the end-to-end throughput, but interference between these multiple paths causes a significant reduction in the overall throughput [47].

By combining appropriate path selection, and a systematic packet transmission schedule, this throughput reduction can be avoided. While the maximum possible throughput can be achieved using three non-interfering paths [39], the problem of finding such non-interfering paths is the same as the problem of finding a chordless cycle containing a pair of vertices in a graph, which is actually NP-Complete [10].

It is not always necessary to use non-interfering paths, however. By finding vertex disjoint paths between source s and destination t which follow certain patterns of interference, we can achieve throughputs which are optimal or close to optimal. In this chapter, we focus on finding such paths for a single *s-t* pair in a distributed fashion. Our approach can easily be extended to the multiple *s-t* pair case. The results of our work has also been presented in [60].

The chapter is organized as follows. Section 4.2 discusses literature that is relevant to the proposed work. Section 4.3 presents the ideas underlying the centralized combinatorial approach for finding interference aware multiple *s*-*t* paths. Section 4.4 provides the distributed algorithm which is used for finding vertex disjoint multiple *s*-*t* paths. Section 4.5 concludes the discussion.

4.2 Related Work

Many existing centralized algorithms address the issue of improving wireless network throughput by minimizing the impact of interference.

Hu et al. [32], Saha et al. [72] and Jones et al. [36] discuss techniques which try to find multiple node disjoint paths between *s* and *t* such that there are no edges connecting two vertices belonging to different paths. They do not consider simultaneous multiple *s*-*t* transmissions.

Jain et al. [36] and Buragohain et al. [12] discuss strategies which use a centralized multi-commodity flow based linear programming (LP) formulation to exhaustively search and determine the maximum achievable throughput with inter-path links. However these solutions provide neither the paths nor the schedules for transmission. Nevertheless, they are important as they establish the bounds for throughput against which other schemes could be compared.

A third class of solutions adopts a combinatorial approach to find multiple paths which can support a high throughput even with interpath links. The approach proposed in this chapter falls under class (c). A relevant work under this class is presented in Liaw et al. [50]. Liaw et al. [50] find the maximum achievable throughput for a given

wireless network by computing all possible shortest paths of a given length k and independent paths for these shortest paths in such a way that for a given shortest path an independent shortest path is a vertex disjoint path. Clearly, this approach of computing the maximum throughput will have exponential complexity.

All the current works use a centralized approach to this problem and to the best of our knowledge, a purely distributed (message passing) algorithm for solving this problem has not been proposed in the literature.

4.3 Finding interference aware multiple paths

We mention the key lemmas for finding interference aware s-t paths based on our discussion in Chapter 3. We start with the two paths case. We assume that packets are injected into all the paths in the multi-path set at a constant rate. Let λ be the inter-packet interval time, also referred to as the periodicity. Here λ is the inter-packet arrival time with respect to a single path. The nodes along a path relay the packets that they receive in the time slot following the reception without any delay. Allowing the intermediate nodes to delay their forwarding could have an adverse effect on end-to-end delay experienced by the subsequent packets sent along the path.

4.3.1 Two paths

Let h_1 and h_2 be the number of hops in paths P_1 and P_2 between source s and destination t. Let t_1 and t_2 represent the time slots when the first packets are respectively injected into paths P_1 and P_2. Let the packets traveling along P_1 be designated as type I and packets traveling along P_2 be designated as type II.

Lemma 4.1: At the destination node, no collisions occur between packets of type I and type II if t_1 and t_2 are chosen such that $(h_1 + t_1) \bmod \lambda \neq (h_2 + t_2) \bmod \lambda$. ■

Lemma 4.2.1: Let us assume we have three paths with hop counts h_1, h_2 and h_3. There are no collisions at the destination if $(h_1 + t_1) \bmod \lambda \neq (h_2 + t_2) \bmod \lambda \neq (h_3 + t_3) \bmod \lambda$. ■

Lemma 4.2.2: Let r_1, r_2 and r_3 be the remainders after dividing the hop counts h_1, h_2 and h_3 by 3. The condition in lemma 2.1 is satisfied for $\lambda = 3$ only if $r_1 = r_2 = r_3$ or if $r_1 \neq r_2 \neq r_3$. If $r_1 = r_2 \neq r_3$, then the smallest value of λ which satisfies Lemma 4.2.1 is 4. ■

The problem of finding two non-interfering paths between a source and destination is the same as the problem of finding a chordless cycle containing two given vertices in a graph, which has been shown to be NP-Complete [10].

Figure 4.1 shows a topology with maximum possible aggregate throughput even with three paths having heavy mutual interference (the dashed lines represent the interfering edges). In other words, a packet is received by the destination during each time slot starting from time slot 4. The reason for this is that the paths exhibit *non-destructive interference,* i.e., the nodes on the two ends of all the interfering edges receive and transmit their packets simultaneously, thereby avoiding collisions. Such interfering edges are referred to as *non-destructive edges*. A set of two or three vertex disjoint paths are said to have non-destructive interference if it is possible to schedule packet transmissions along the paths without any collisions for a desired periodicity λ despite the presence of interfering edges. In other words, a set of vertex disjoint paths

are said have non-destructive interference if all the interfering edges among the paths are non-destructive in nature.

Lemma 4.3: Let e be an interfering edge between nodes v_1 and v_2 which lie along paths P_1 and P_2. Let the nodes v_1 and v_2 be k_1 and k_2 hops away from the source s. The edge e will be non-destructive if :

(i) $\left| ((k_1 + t_1) \bmod \lambda - (k_2 + t_2) \bmod \lambda \right| \neq 1$, and

(ii) $\left| ((k_1 + t_1) \bmod \lambda - (k_2 + t_2) \bmod \lambda \right| \neq \lambda - 1$.■

Theorem 4.1: Two (resp. three) vertex disjoint paths have non-destructive interference between them for a given periodicity λ if all the interfering edges among the two (resp. three) paths are non-destructive with respect to λ and if Lemma 4.1 (resp. Lemma 4.2.1) is satisfied.■

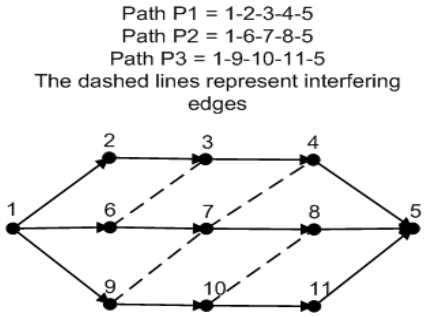
Hence our goal is to find paths such that we can satisfy the conditions of theorem 1 with the smallest possible value of λ greater than or equal to 3.

4.4 A distributed algorithm for interference aware *s*-*t* paths

A key issue in using a centralized approach for finding paths and schedules based on earlier discussions is that the entire network topology information must be stored at the source. This could be difficult on large scale wireless networks, and if the network is fairly dynamic with nodes entering and leaving the network (or new traffic coming in) constantly, it is highly desirable to have a distributed implementation based on message passing for finding interference aware vertex disjoint paths.

Our distributed algorithm follows the strategy used to find vertex disjoint paths in [45]. Our distributed algorithm also uses the knowledge gained from Lemmas 1, 2, 3 and Theorem 1 for finding interference aware vertex disjoint paths. Thus at the end of its execution, the distributed algorithm will find paths which are not only vertex disjoint, but also interference aware.

The centralized algorithm for finding vertex disjoint paths works as follows [45]: we are given a graph $G = (V, E)$ and a source s and a destination t. We first find a path P_1 from source s to destination t (for example, we may use the shortest path from source to destination). The numbers marked in Figure 4.3(a) shows an s-t path in graph G. We remove all the nodes belonging to the path (including the source and destination) and obtain all the connected components of the graph. These connected components are called *bridges*. Figure 4.3(a) shows an example of bridges for the original graph G with respect to the path marked from s to t. The edges connecting a bridge to the path P_1 are called bridge links. Any bridge could have multiple bridgelinks.



	P1	P2	P3	P1	P2
1	1 → 2				
2	2 → 3	1 → 6			
3	3 → 4	6 → 7	1 → 9		
4	4 → 5	7 → 8	9 → 10	1 → 2	
5		8 → 5	10 → 11	2 → 3	1 → 6
6			11 → 5	3 → 4	6 → 7
7				4 → 5	7 → 8
8					8 → 5

Figure 4.1: The set of paths allow maximum throughput despite interpath interference. The right half is a timing diagram, where the rows represent time slots and the columns represent the movement of packets through the paths.

The leftmost bridge link is the edge connecting the bridge to a path node which is closest to s. The rightmost bridge link is the edge connecting the bridge to a path node

which is closest to *t*. The path node belonging to the leftmost (resp. rightmost) bridge link is called the leftmost (resp. rightmost) attachment point. A bridge path is defined as a path between the leftmost and rightmost attachment points whose edges all belong to the bridge.

We can define conceptual arcs between bridges to form a bridge graph. Each bridge is represented as a node in the bridge graph. The source *s* and destination *t* are also nodes in the bridge graph. A conceptual arc in the bridge graph is defined as follows: a given bridge B_i will have candidate bridges to which it could have a conceptual arc.

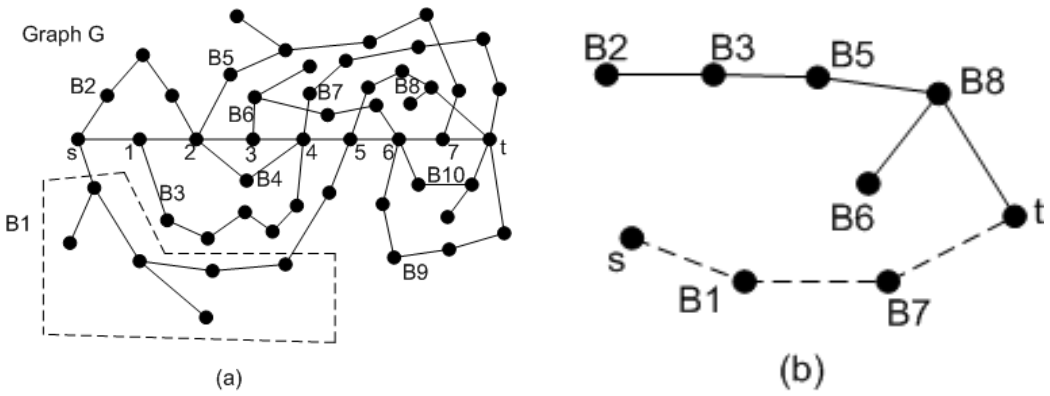


Figure 4.2: (a) Bridges B1..B10 marked on original graph G. (b) conceptual edges of bridge graph G_B.

A bridge link e_m is defined to be to the left (respectively right) of a bridge link e_n, if the path node belonging to e_m is closer to (respectively farther from) *s* than the path node belonging to e_n. A bridge B_z is a candidate bridge for bridge B_i if the leftmost bridge link of B_z is to the left of the rightmost bridge link of B_i and the rightmost bridge link of B_z is to the right of the rightmost bridge link of B_i.Of all the candidate bridges of bridge B_i, we add a conceptual arc to the bridge B_j if B_j has a rightmost attachment point closest to *t* among all the candidate bridges B_z of B_i.

We also add a conceptual arc from source *s* to a bridge B_s if *s* is the leftmost attachment point of B_s and the rightmost attachment point of B_s is closer to *t* than all the other bridges whose leftmost attachment point is *s*. We add a conceptual arc from all bridges B_x to the destination *t* if *t* is the rightmost attachment point of B_x. Figure 4.2(b) shows the conceptual bridge graph for the original graph G in Figure 4.2(a). We enforce a rule that a given bridge (as well as node *s*) can have only a single outgoing conceptual arc to another bridge in the bridge graph. When there are multiple bridges satisfying the conditions described above, we arbitrarily choose one. In the conceptual graph (called a *bridge* graph), if there exists a path between source *s* and destination *t*, then it has been proved that there exists two vertex disjoint paths from *s* to *t* in the original graph *G*.

The distributed algorithm of [44] follows the centralized algorithm given in [45]. For the distributed algorithm, each bridge in the *s*-*t* path in the conceptual graph marks the rightmost attachment point and leftmost attachment point as leftmost and rightmost *special* nodes. The first vertex disjoint path is obtained by moving along the path P_1 until a leftmost *special* node, then moving along the bridge which marked this node, and then moving back along the path at the rightmost *special* node of the bridge, and so forth. The second vertex disjoint path is obtained in the same manner, except that we start along the bridge path of the bridge connected to s. Figure 4.3 shows the vertex disjoint paths (as dashed lines) for the original graph *G* found by using the given *s*-*t* path.

Upon closer observation, we notice that the single *s*-*t* path in the bridge graph decomposes into two vertex disjoint paths in the original graph *G*. Thus we can also

infer that the choice of the path from *s* to *t* in the bridge graph will determine the quality of the interference aware path, which is measured based on the value of λ. In other words, smaller the λ, the better is the quality of the path set, since smaller λs give higher aggregate throughput. Our goal will be to find a path that minimizes λ in the bridge graph in a distributed fashion. We will now implement the following steps in a distributed fashion.

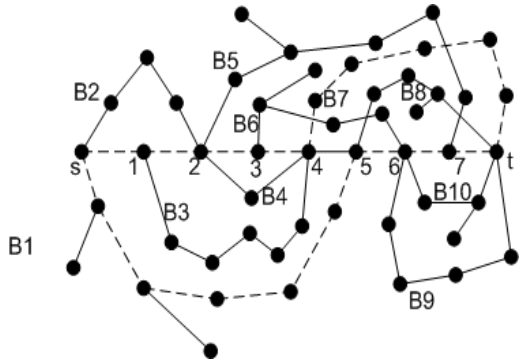


Figure 4.3: Vertex disjoint paths in original graph G

Step 1: Find a path P_1 from node *s* to node *t* in the original graph G: This step can be easily done in a distributed fashion by constructing a spanning tree rooted at node *s*. Nodes in the path P_1 can also be numbered linearly from *s*. This step will have a message complexity of $O(m)$.

Step 2: Decompose the graph into its bridges relative to the path P_1: The formation of the bridges can be accomplished by finding the connected components through a distributed construction of spanning trees for all nodes not on path P_1. At the end of this step, all nodes not on path P_1 will be a part of some spanning tree. The leftmost and rightmost attachment points of each bridge are also known (and this information is propagated to the root of the spanning tree). The identity of the root becomes the bridge identifier. At this point, all the bridges of *G* relative to the path P_1 have been formed

and the leftmost attachment point of a bridge knows the rightmost attachment point of each bridge to which it is attached. Each leftmost attachment point, *v*, stores in variable $r^+(v)$ the rightmost attachment point among all bridges for which *v* is a leftmost attachment point.

The corresponding bridge identifier (i.e. the bridge whose rightmost attachment point is $r^+(v)$) is stored in the variable $B^+(v)$. The root, *v*, of the bridge knows the leftmost and rightmost attachment points of the bridge and stores these in the variables *lap*(*v*) and *rap*(*v*), respectively. For this step, the number of messages is bounded by the number of edges, for a message complexity of $O(m)$.

Step 3: Construct a new graph $G_{EB}(V_{EB}, E_{EB})$ called an *expanded bridge graph*: Node *s* initiates this process by sending a message to the next node on path P_1 containing values *rap*(*v*) and *lap*(*v*) for all bridges whose *lap*(*v*) is *s*. Each node *v* receiving the message does the following:

Case *v* of

1. not an attachment point

A message is sent to the next node in the path at the end of this case statement

2. a left attachment point only

Append to the message *rap*(*i*) and *lap*(*i*) for all bridges (with root identifier *i*) for which *v* is a leftmost attachment point

3. a right attachment point only

a) *bridges* = 0

b) for all j where $v = rap(j)$

i) for each pair of ($rap(i)$, $lap(i)$) in the message

1) if $lap(i) < rap(j)$ and $rap(i) > rap(j)$ and $lap(i) < lap(j)$

A) append i to the array *bridges*

c) Send ADD_BRIDGE_NBOR message containing *bridges* to root j

4. left and right attachment point

a) $bridges = 0$

b) for all j where $v = rap(j)$

i) for each pair of ($rap(i)$, $lap(i)$) in the message

1) if $lap(i) < rap(j)$ and $rap(i) > rap(j)$ and $lap(i) < lap(j)$

A) append i to the array *bridges*

c) Send ADD_BRIDGE_NBOR message containing *bridges* to root j

Append to the message $rap(i)$ and $lap(i)$ for all bridges (with root identifier i) for which v is a leftmost attachment point

End of case statement

The root of a bridge receiving an ADD_BRIDGE_NBOR message knows that there is a conceptual outgoing arc to the bridge identified in i (from the definition of the expanded bridge graph) and appends this to an array *out-arcs*. This process is repeated for the next node in the path P_1 and when node t receives the message it sends back a

message to node *s* and step 3 is completed. This step consists of sending messages along the path and to roots of bridges with time and message complexity $O(n)$.

Step 4: Find a suitable "best path" from node *s* to node *t* in the expanded bridge graph: We will consider the interference pattern that a given bridge B_i makes with the path P_1 and compute the best value of λ. This value is assigned as the node weight of each bridge. We also assign a weight on an arc (B_i, B_j) with the observed periodicity λ caused by the interference pattern of the bridges B_i and B_j combined. Now we execute a distributed shortest path algorithm from *s* to *t* on the expanded bridge graph with a message and time complexity of $O(n^2)$[70].

Assigning node and edge weights for this distributed shortest path algorithm is a key step in this process. Using our knowledge of interference patterns, we can assign node and edge costs to the expanded bridge graph so as to find interference aware paths. Our solution depending on simple factors such as the path lengths and the hop counts of interfering nodes is what allows us to perform this step. The output of the shortest path algorithm will result in marking of the left most attachment points of the bridges in the shortest path. Sections 4.4.1 and 4.4.2 elaborate on this idea.

Step 5: From step 4, a path from node *s* to node *t* will be found in G_{EB} (the graph is assumed to be biconnected). The two vertex disjoint paths between node *s* and node *t* are constructed as follows: Node *s* simultaneously sends a message along path P_1 and across the bridge identified as the next hop in the shortest path. This message contains the right most attachment point of the next hop bridge. When the message along P_1 reaches the leftmost attachment point of the second bridge in the shortest path, it

branches off from P_1 and traverses along the second bridge. When the message sent across the first bridge reaches the right most attachment point, it now changes its course and traverses along P_1.The process continues from the right most attachment of the bridge. Because the message traverse along the path and the nodes in the spanning trees of the bridges, the total time and message complexity of this step is *O(n).*The total time and message complexity of the entire algorithm is bounded by the time and message complexity of the distributed shortest path algorithm used in step 4.

4.4.1 Finding non-interfering paths

It is possible to find non-interfering vertex disjoint paths at step 4 of the vertex disjoint path algorithm. Notice that a bridge may contain nodes which share edges with the selected *s*-*t* path, but it cannot have nodes which share an edge with any other bridge. In other words, a bridge may interfere with the initial path, but never with another bridge. Given bridge B_i, suppose we are able to construct a spanning tree for B_i in step 2 in such a way that the *lap* and the *rap* have a path on the bridge which do not interfere with the initial path. We only choose such bridges in the expanded bridge graph.

Additionally, when choosing edges for the expanded bridge graph, we may choose to add an edge from bridge B_i to bridge B_j in the expanded bridge graph. We add the additional restriction that edge (B_i, B_j) will be added to the expanded bridge graph if and only if the *rap* of B_i and the *lap* of B_j are at least two hops away on the initial *s*-*t* path (in other words $lap(B_j) - rap(B_i) \geq 2$). On this expanded bridge graph, a path from source *s* to destination *t* forms a pair of non-interfering vertex disjoint paths. Note that while this technique may discover non-interfering paths, it is not guaranteed to find a

pair of non-interfering paths even if they exist in the graph since it depends on the selection of the initial *s*-*t* path.

4.4.2 Finding interference aware paths

Just as we did in Section 4.5, we can choose to assign node weights to bridges in the expanded bridge graph and restrict the edges which we may add to the expanded bridge graph to find interference aware paths which can still permit high throughput. A preliminary approach to finding these interference aware vertex disjoint paths is as follows: for each bridge B_i, assign a node weight which is the minimum value of λ it can support with the initial *s*-*t* path chosen as the second path, after allowing for the following scenario – the $lap(B_i)$ may be the h^{th} hop along one vertex disjoint path, where h is computed by looking at all bridges B_j whose $rap(B_j) < lap(B_i)$. After taking into account all the possible h mod λ values (which is the essential quantity of interest – even if there are many bridges before, the number of possible h mod λ values is still bounded by λ itself) we must compute the best possible value for λ that the bridge B_i may be able to sustain even with interfering edges to the initial *s*-*t* path. Similarly we add an edge in the expanded bridge graph between bridge B_i and bridge B_j only if the two bridges do not share any edge with the initial *s*-*t* path except those vertices in the initial *s*-*t* path which are shared by both bridges. The edge is assigned a weight of zero. In other words, the bridge B_i and bridge B_j are permitted to have interfering edges only with those nodes in the initial *s*-*t* path whose node numbers (along the path) are smaller than $rap(B_i)$ and larger than $lap(B_j)$ if an edge exists between them on the expanded bridge graph. The path with the smallest bottleneck node weight on this expanded bridge graph (this can be distributedly computed using a shortest path algorithm) yields

suitable interference aware vertex disjoint paths between source and destination. Here the bottleneck node weight refers to maximum node weight along a given path. Note that in contrast to the algorithm described in section 4.5, this approach does permit interference among the bridges and the initial *s*-*t* path. However, we are still likely to fall short of the throughput realizable from the centralized algorithm proposed in section 3.7 as the algorithm in Section 3.7 considers a much larger set of combinations for the possible paths.

4.5 Summary

In this chapter, we have presented a distributed algorithm for finding interference aware vertex disjoint *s*-*t* paths to improve the throughput of wireless networks. In the context of wireless sensor networks, it is important that the distributed algorithms used for path problems respect the typically small packet size in most sensor networks (of the order of 10s of bytes). The proposed approach will allow interference aware vertex disjoint path discovery requiring single messages communicated even under such restrictive packet sizes.

The inherently hard nature of the problem of finding suitable routing paths for improving wireless throughput also raises the interesting question of how well a distributed algorithm for such a purpose may perform. When packet size constraints are added, this leads to even more restricted implementations for these distributed algorithms. Exploring the tradeoffs between solution quality and the amount of resources allocated to a distributed algorithm, be they total energy, maximum packet size or perhaps storage space, would be interesting future work.

Chapter 5

5 Multi-radio activation

5.1 Introduction

While the first generation of wireless sensors had limited processing and storage capabilities, advances in technology, in combination with increased application demands have resulted in more powerful second generation sensor nodes. These nodes possess relatively higher processing and storage capabilities achieved through the use of powerful CPUs, and large memories [34], [57]. These nodes are also capable of operating multiple radios simultaneously, each with a different power, range and bandwidth rating. Though such multi-radio sensors are currently used as gateways or cluster-heads in sensor networks, technological advancement may soon equip even the commonly used sensor nodes with multiple radios.

While the capabilities of sensor nodes have increased along several fronts, they will continue to be powered by batteries. Consequently, energy conserving mechanisms are of paramount importance even in next generation wireless sensor networks. The radios in a multi-radio sensor node may differ not only in terms of their communication capabilities but also in terms of energy efficiency and usage. High bandwidth, long-range radios usually possess higher energy efficiency, in terms of energy expended per bit transmitted, than low bandwidth, short-range radios [80]. However, high bandwidth radios also consume more power when idling than low bandwidth radios. Therefore, activating several high bandwidth radios when there is not a lot of data to be transmitted may result in considerable energy wastage. On the other hand, due to their

greater reach, long-range radios can reduce the network diameter; consequently, the latency involved in delivering sensory data to a prescribed destination will decrease with the use of long range radios. Several of them may need to be activated when the application demands smaller data delivery latency. Thus the issue of radio activation is closely tied to the requirements of the application.

Earlier research on multi-radio systems primarily used the additional radios to improve the network performance in several ways. The focus of such works have been on transmission scheduling [74], [55], hierarchical power management [79], throughput enhancement [91], and resource discovery and mobility support [77], [67]. Multiple radios have also been used to find suitable end to end paths satisfying certain quality guarantees [74]. There also have been works that clearly document the performance benefits of multi-radio wireless networks in real-life settings [22], [83], [71]. The above works assume that the network remains connected even when all the sensor nodes activate only their lowest power radio. However, in a general setting, such a requirement on the connectivity cannot be guaranteed. Radios with higher power and longer range may have to be activated even to make the network connected. In [14], the authors consider linear networks where a random fraction of the nodes in the network have dual-radio functionality and apply probabilistic techniques to describe the connectivity of such networks.

In this chapter, we focus on energy efficient radio activation in a sensor network where each node has $K > 1$ radios. The radios r_1, r_2,…, r_k in a node are such that the one hop reachability distance (resp. energy expended) of (resp. by) radio r_i is greater than that of r_j, $1 \leq j < i \leq k$. Given such a network, the problem of energy efficient radio

activation is to minimize the total energy spent by the active radios across all nodes in order to maintain a connected network. We make several contributions in this work: (1) We show that the problem of energy efficient radio activation is NP-Complete. (2) We propose four different polynomial time approximation methodologies for solving this problem in networks with $K = 2$. The first two methodologies employ a series of non-trivial reductions to leverage on existing approximation solutions for other known NP-Complete problems. The third methodology is based on the minimum spanning tree algorithm. The fourth methodology is a greedy algorithm that is proposed afresh. (3) We extend these solutions to the general case of $K > 2$ radios as well. (4) Our analytical and experimental studies of the four solutions reveal that the greedy algorithm and the minimum spanning tree solution have the best *worst case* performance while the greedy algorithm has the best *average case* performance. Preliminary results from this work have been submitted in [63].

The rest of this chapter is organized as follows. In section 5.2, we define the different variants of the radio activation problem. Section 5.3 discusses the complexity of the basic version of the radio activation problem while sections 5.4 and 5.5 describe different solution methodologies for solving this problem. We study the average case behavior of these solutions in section 5.6. Section 5.7 focuses on the complexity of the general versions of the radio activation problem and their solutions. We study the average case behavior of these solutions in section 5.8. We discuss literature relevant to the proposed research in section 5.9. Finally, we present our conclusions in section 5.10 and outline our ongoing and future research efforts in this direction.

5.2 Problem definition

Let v_1, v_2,…, v_n be the nodes present in a wireless sensor network. Each node v_i is equipped with K radios r_1, r_2, …, r_k. Let R_k and P_k respectively be the range and power consumption of radio k. Without loss of generality let $R_1 \leq R_2 \leq \ldots R_K$ and $P_1 \leq P_2 \leq \ldots P_K$. Let $I_{i,k}$ be an indicator variable which denotes the on/off status of radio k in sensor node v_i, i.e., $I_{i,k} = 1$, if radio k is turned on in node v_i and is 0, if it is turned off. Two nodes v_i and v_j are said to have an *edge of type k* between them if and only if $d(v_i, v_j) \leq R_k$ and $I_{i,k} = I_{j,k} = 1$, where $d(v_i,v_j)$ denotes the geographical distance between the nodes v_i and v_j. Thus two nodes can have a maximum of K edges between them. A *path* is said to exist between two nodes v_i and v_j if a sequence of distinct nodes v_i, v_a, v_b,…, v_m, v_j can be found such that any two adjacent nodes in the sequence have an edge of some type between them.

Given these notations, we focus on a set of related problems each of which is significant in its own way. To begin with, we consider a network of devices that have dual radios (i.e., $K = 2$) – a low power, short-range radio and a high power, long-range radio. Such devices are already available commercially off-the-shelf [34] and hence have immediate relevance. In such networks the radio activation problem can manifest in two different variations.

Problem MHP2C: In the first kind, the low power radios in all the devices are turned on by default. As this may not guarantee network connectivity, the aim is to turn on minimum number of high power radios so as to make the network connected. This problem is referred to as the *Minimum High Power 2-Radio Connectivity problem*

(MHP2C). Formally, given that $I_{i,1} = 1$, $i \in \{1, 2, \ldots, n\}$, determine $I_{i,2}$, $i \in \{1, 2, \ldots, n\}$ such that:

1) A path exists between v_i and v_j, $i, j \in \{1, 2, \ldots, n\}$, and

2) $\sum_{i=1}^{n} I_{i,2}$ is minimized.

Problem MP2C: In the second kind, no radios are turned on by default and the objective is to activate the best set of low and high power radios to get a connected network with minimum power. This problem is referred to as the *Minimum Power 2-Radio Connectivity Problem* (MP2C). In other words, the objective is to determine $I_{i,k}$, $k \in \{1,2\}$, $i \in \{1, 2, \ldots, n\}$, such that

1) A path exists between v_i and v_j, $i, j \in \{1, 2, \ldots, n\}$

2) $\sum_{i=1}^{n} \sum_{k=1}^{2} (I_{i,k} \times P_k)$ is minimized

Problem MPKC: The above MP2C problem can be generalized to a scenario where each device may have up to K radios. The objective in such a network is to selectively activate each of the K radios in a node to guarantee network connectivity while minimizing the total power consumed across the active radios in all the nodes. This problem is referred to as the *Minimum Power K-Radio Connectivity Problem* (MPKC). Formally stating, the objective is to determine $I_{i,k}$, $i \in \{1, 2, \ldots, n\}$, $k \in \{1, 2, \ldots, K\}$ such that:

1) A path exists between v_i and v_j, $i, j \in \{1, 2, \ldots, n\}$ and

2) $\sum_{i=1}^{n} \sum_{k=1}^{K} I_{i,k} \times P_k$ is minimized

As mentioned earlier, each of the three problems defined above are useful in their own way. In the following sections, we initially focus on the MHP2C problem and take up MP2C and MPKC problems later.

5.3 Complexity of MHP2C problem

We can also observe that the network formed by the dual-radio sensor nodes is best modeled as a multi-graph with at most two edges between any two vertices. We argue that such a multi-graph can be converted to a simple graph for the MHP2C problem without any loss of generality. Under the MHP2C problem, when all the low power radios are turned on, the network as a whole need not be connected. This implies that the network may remain partitioned as a set of connected components. No two components can communicate with each other unless they are connected by a high power radio link. This necessitates that at least one node in each component should turn its high power radio on and the goal is minimize the number of high power radios that are turned on across all components. Figure 5.1 illustrates the MHP2C problem through an example. In the following discussions, the term low (resp. high) power edge will refer to an edge of type 1 (resp. 2) in the network.

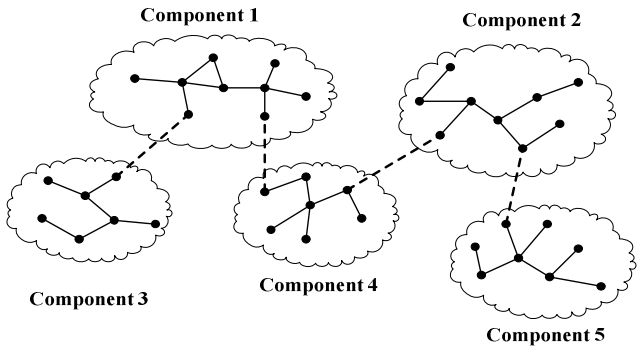


Figure 5.1(a) MHP2C example. The solid lines represent low power edges, the dashed lines represent high power edges. The number of high power radios which need to be turned on in this case is 8.

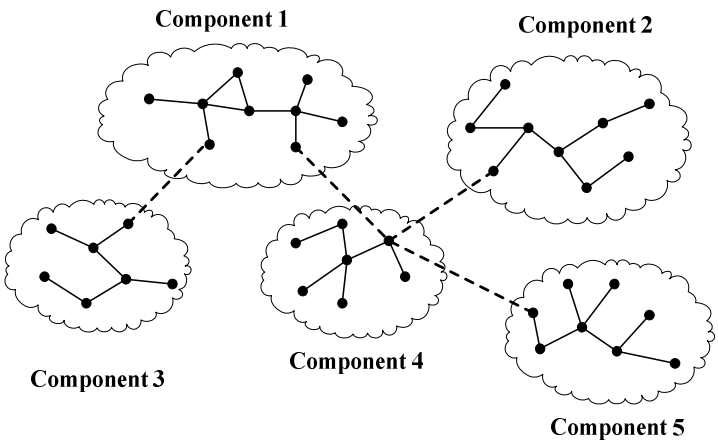


Figure 5.1(b) MHP2C example. A different way of connecting the components. The solid lines represent low power edges, the dashed lines represent high power edges. The number of high power radios which need to be turned on in this case is only 6.

Theorem 1. The MHP2C problem is NP-Complete.

Proof: We show that MHP2C is NP-Complete by obtaining a reduction from the *minimum hitting set problem*. We show that for every instance of the minimum hitting set problem (a known NP-Complete problem), we can create an instance of the MHP2C problem such that the solution to the MHP2C problem is also a solution to the minimum hitting set problem. The minimum hitting set problem is defined as follows: Given a collection C of subsets of a finite set S, the aim is to find a subset $S' \subseteq S$ such that S' contains at least one element from each subset in C. The minimum cardinality subset S' is called the minimum hitting set. The minimum hitting set problem is known to be NP-Complete [23].

We now state our reduction. Let C_1, C_2,…, C_n be the subsets belonging to C. Consider any subset $C_p \in C$. We can construct a disconnected graph $G = (V, E)$ as follows. For each element $i \in C_p \; \forall \; C_p \in C$, we add a vertex $v_{i,p}$ to graph G. This gives

us a set of $\sum|C_i|$ vertices for graph G. The edge set of G is constructed as follows. For each element $i \in C_p$, for every other element $j \in C_p$, an edge $e_{i,j}$ is constructed connecting i and j in graph G. This step is repeated for each $C' \in C$. Now G is a disjoint union of cliques, with each clique representing a subset $C_p \in C$. We now add what we call 'inactive' edges to the graph G as follows: for each element $m \in S$ such that $m \in C_i$ and $m \in C_j$ where $i \neq j$, we add an edge connecting $v_{m,i}$ and $v_{m,j}$.

Now graph G can be viewed as a network formed by dual-radio wireless sensor nodes with each vertex being the sensor node. The cliques in G represent the different low-power connected components in the dual radio network. The edges connecting the vertices within a component represent the low power radio links between the corresponding nodes. We call these edges as component edges. The inactive edges between the cliques represent the high power radio links between the components in the network. An inactive edge becomes an active edge when the high power radios of both nodes on which the edge is incident are turned on. Let a set of high power radios be turned on such that the set of active edges plus the component edges forms a connected sub-graph G' that connects all the components in G.

Consider the set $H = \{i \mid \text{vertex } v_{i,p} \in G'\}$. If $v_{i,p}$ is a node whose high power radio is turned on, then element i of set C_p will be included in the set H. Since G' is connected sub-graph of G, there exists a node $v_{i,p} \in C_p$, $\forall\ C_p \in C_1, C_2, \ldots, C_k$ such that its high power radio is turned on. In other words, the set H defined above is a hitting set of C. It follows that the if one can determine the minimum number of nodes required to turn on their high power radios to obtain a connected subgraph G', then the corresponding H

will also be a minimum hitting set for *C*. Figure 5.2 shows an example of how the reduction works.

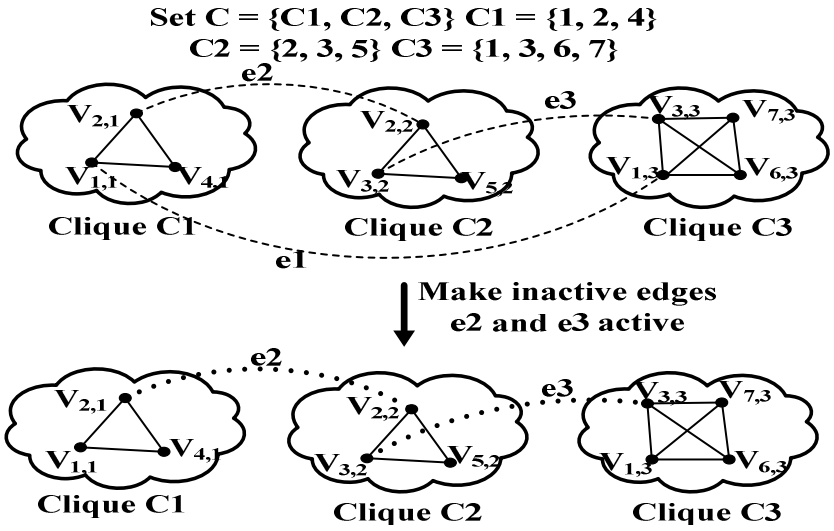


Figure 5.2 Hitting set reduction. In the example shown, solid edges are the component edges within a clique. The dashed edges are inactive edges, and once they become active they turn into dotted edges. The numbers corresponding to the active edges (in this case 2 and 3) are chosen to form the hitting set.

Suppose we could find a solution to the MHP2C problem in polynomial time. The above reduction shows that a polynomial time solution to MHP2C will also solve the minimum hitting set problem in polynomial time. However the minimum hitting set problem is a known NP-Complete problem. Hence MHP2C is also NP-Complete.■

As mentioned earlier, we propose four different methodologies for solving the MHP2C problem. The first two employ a series of non-trivial reductions to leverage on existing approximation solutions for other known NP-Complete problems. The third solution is based on the minimum spanning tree solution while the fourth methodology is proposed afresh.

5.4 Solving MHP2C through existing solutions

While we can prove the NP-Completeness of the MHP2C problem using a reduction from the minimum hitting set problem, unfortunately not every instance of the MHP2C problem can be mapped to a minimum hitting set problem. Consequently, solutions of the hitting set problem may not be applicable for MHP2C. Hence we look towards other approaches which can be used to solve the MHP2C problem.

5.4.1 Node weighted group Steiner tree

The first approach we take is to consider the MHP2C problem as a *node weighted group Steiner tree* problem. The group Steiner tree problem is defined as follows: given an undirected weighted graph $G = (V, E)$ with a cost function w on the edges and a family $N = N_1,...,N_c$ of c disjoint groups of nodes $N_i \in V$, find a minimum cost tree which contains at least one node from each group N_i. The node weighted group Steiner tree problem is identical to the group Steiner tree problem except that the cost function w is defined on the nodes. It is easy to see how the MHP2C problem can also be considered as a node weighted group Steiner tree problem – if we consider the disconnected graph formed by the low power nodes, each component forms a group. Now we need to select at least one high power radio from each group and ensure that the network becomes connected.

There is no readily available solution for solving the node weighted group Steiner tree problem. Most solutions proposed in the literature for the group Steiner tree problem consider the edge weighted case [24], [29]. Arie Segev has shown in [75] that any node weighted Steiner tree problem with nonnegative node costs can be transformed into the edge weighted directed Steiner tree problem as follows

(Transformation I): Let w_j be the node cost associated with node *j*, and w_{ij} represent the edge cost associated with edge (*i, j*). If both node and edge cost coefficients w_i and w_{ij} are nonnegative and a root node is identified, the original problem can be transformed into the standard edge weighted directed Steiner tree problem by defining arc cost coefficients for each edge (*i, j*) such that the new cost coefficient $w_{ij}' = w_{ij} + w_j$.

Through *Transformation I*, we convert our node weighted group Steiner tree problem to a directed edge weighted group Steiner tree problem on the transformed graph. Charikar et al showed in [16] that every directed group Steiner tree problem instance can be solved as a directed Steiner tree problem using the following transformation (*Transformation II*): for each group N_i, introduce a dummy vertex x_i and connect x_i using zero cost edges to each of the vertices in N_i. These dummy vertices are the terminals (or the required nodes) in a directed Steiner tree instance with the same root. After *Transformation II*, the original node weighted group Steiner tree problem becomes a directed edge weighted Steiner tree problem, which is NP-Complete [16]. Let the number of nodes in the network be *n* and the number of groups be *c*. The best known approximation algorithm for solving the directed edge weighted Steiner tree problem is by Charikar et al. [16], where they provide an algorithm which achieves an approximation ratio of $i(i\text{-}1)c^{1/i}$ in time $O(n^i c^{2i})$ for any fixed $i > 1$. Setting $i = \log c$, this is a quasi-polynomial time solution with an $O((\log c)^2)$ approximation ratio.

5.4.2 Minimum Connected Dominating Set solution

The minimum connected dominating set (MCDS) problem is defined as follows: given a graph $G = (V, E)$, a connected dominating set is a set of vertices $V' \in V$ such that a)

every vertex not in V' is connected to at least one member of V' by an edge in E and b) the subgraph induced by V' is connected. The connected dominating set of least size (here the size refers to the cardinality of the set) is the minimum connected dominating set for G. If the objective function that is being minimized is the sum of the node weights, then this is the weighted minimum connected dominating set problem, which is also known to be NP-Complete [25].

Let $H = (V,E)$ be the given multi-radio network where V is the set of all dual-radio sensors and E is the set of all low power and high power edges in the network. Let $H' = (V, E')$, where $E' \in E$ is the set of low power edges in E. The connected components of H' are determined. The given network H is transformed into a graph $G = (V', E'')$ on which an MCDS solution is run to find the set of nodes that need to activate their high power radio. Graph $G = (V', E'')$ is obtained from H as follows. To each component i of H', a dummy vertex i is added. The vertex set V' is the union of set V and the set of all dummy vertices $\{x_i\}$. The edge set E'' is the a) set of all lower power edges in H, b) high power edges (u, v) in E such that u and v are in different connected components of H', and c) edges (x_i, u) from each dummy vertex belonging to a connected component i to all nodes u in the same connected component i. On this resulting graph, the MCDS algorithm is executed and the dominating set is determined. The purpose of adding a dummy vertex is to make sure that at least one vertex from each component is selected to have its high power radio turned on. Without the dummy vertex, it is possible that every vertex in some component to be dominated by a connected dominating set containing vertices not belonging to the component. Figure 5.3 uses a subset of the connected components of Figure 5.1 to explain this approach. The best known

polynomial time approximation algorithm for the minimum connected dominating set problem on unit disk graphs has been proposed by Wan et al [82] and provides a constant approximation factor of 8. Their distributed algorithm has a time complexity of O(n) and a message complexity of O($n$log$n$), which means the centralized version of their algorithm has a complexity of O(n^2log n).

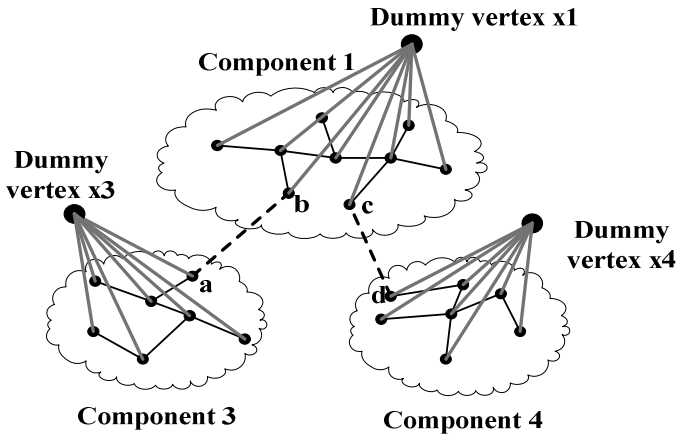


Figure 5.3 Minimum Connected dominating set transformation. The gray edges connect all nodes in a group to the dummy vertex of the group. We can see that the set of nodes {x_3, a, b, x_1, c, d, x_4} form a connected dominating set.

5.4.3 Minimum spanning tree solution

The third approach we adopt for solving the MHP2C problem is to employ a minimum spanning tree (MST) algorithm such as Kruskals [19]. We do this by assigning suitable edge weights to the links in the network. A communication link requires two radios to be turned on. Therefore, for all high (low) power radio edges, we assign an edge weight equal to twice the power consumed by the high (low) power radios. Only those radios that correspond to the edges in the resulting spanning tree are activated. By its very nature, the spanning tree so constructed guarantees that the network is connected.

Lemma 2. The approximation ratio for the MST solution for MHP2C is 2.

Proof: The MST is constructed by sorting all edges by their edge weights and incrementally adding these edges in ascending order to form a tree. Therefore, the low power edges are first used to construct the tree before the high power edges are used. Therefore it is clear that the MST will form low power connected components first and then find the suitable high power edges to connect these components. Hence the number of high power edges used will be no more than the number of components which need to be connected. Since each high power edge effectively turns on no more than two high power radios, the number of high power radios will at most be $2c$, where c is the number of components.

However, we also have a theoretical lower bound on the minimum number of radios which have to be turned on for connectivity – one high power radio from each group (connected component) giving us a total of c such radios. Let OPT represent the optimal number of radios to be turned on for achieving connectivity in our given network. So we know $c \leq OPT$. This gives us an approximation ratio of 2 for the MST solution. ■

5.5 A greedy solution for the MHP2C problem

While the MST solution minimizes the number of high power edges required to connect the network, it does not minimize the number of nodes that are required to turn their radios on. In the following paragraphs we discuss a greedy approach that attempts to achieve this while maintaining network connectivity.

Let the term *group* refer to a subset of nodes that remain connected when no high power radios are turned on. In other words, each group is a connected component of the network formed by using just the low power edges. Without the loss of generality, let us assume that the network consists of more than one group. By a *covered group*, we refer to a group in which at least one high power radio has already been activated. By *uncovered group*, we refer to a group in which no high power radio has yet been activated. Clearly, before the algorithm begins to execute, the set of covered groups, *C*, is empty, while the set of uncovered groups, *U*, contains all the groups. A node is said to be a covered node if it belongs to a covered group. Let *v* be a vertex belonging to group N_i. The span of a node *u* is the number of neighbors *v* such that: (a) (*u*,*v*) is a high power edge, (b) *u* and *v* belong to different groups, and (c) *v* belongs to an uncovered group. Given a network with *n* nodes and *m* high power edges, the span of every node *u* can be easily determined in O(*n*+*m*) time after determining the connected components formed using low power edges.

Our polynomial time approximation algorithm, named *Alg-A*, for solving the MHP2C problem is given in Figure 5.4. At each step, we select the node with the highest span. We turn on the high power radio of the node. We also select the node's high power neighbors; one from each of the groups spanned by the node, and activate their high power radios as well. After the activation, we update the spans of all the nodes in the network and repeat the process.

1)Initialize the span of all nodes
2)**Do while** $u \neq \emptyset$
a) Select node vmax with highest span. Turn on its high power radio.
b) For each uncovered group N' which is a neighbor of vmax
i) $C \leftarrow C \cup N'$
ii) $U \leftarrow U \setminus N'$

iii) Select $u \in N'$ such that u and v_{max} share a high power edge; turn on u's high power radio
c) Update the span of each vertex in the network
3)**End while**

Figure 5.4 Approximation algorithm *Alg-A* for solving the MHP2C problem.

Time Complexity of *Alg-A*: If we have a total of c groups in a network with n nodes and m high power edges, the outer DO-WHILE loop requires at most c steps. We can construct a max heap such as the Fibonacci Heap [19] using the span values of each node in O(n) time. In the entire execution of the algorithm, at most n nodes will be removed and at most m decrease operations are performed (Step 2c) on the heap. These operations on the Fibonacci Heap is very similar to the operations required to execute the Dijkstra's shortest path algorithm [19]. Hence the total time taken to remove the nodes and update the span values will be $m+n \log n$. Therefore the total time complexity is O($cm + n\log n$).

5.5.1 Approximation ratio

Lemma 3. The approximation ratio of *Alg-A* is 2. *Proof:* We derive the approximation ratio of *Alg-A* while observing that it is a loose bound and there may be scope for tightening it. Recall that in each step of *Alg-A* (the outer loop), at least one group is selected and added to the covered set C. We turn on exactly one high power node per uncovered group added at each step. Also, the high power radio of node v_{max} is turned on. In other words, if we add p groups to the covered set at any step, we turn on exactly p+1 high power radios in that step. Since we must necessarily select two groups in the first step, we effectively execute the outer do loop no more than c-1 times (assuming we would have a connected network if all high power radios were turned on). If the outer loop executes c-1 times, then we add at most 2 high power radios per step (since p

= 1 at each step). Hence at the end of algorithm execution, we have turned on at most $2(c\text{-}1) = 2c\text{-}2$ high power radios which gives us an approximation ratio of 2 (by noting that we need to turn on at least one radio from each component to form the connected graph). ■

Algorithm	Approx. Ratio	Complexity
Directed Steiner Tree[16]	$\log c(\log c\text{-}1)$	$O(n^{\log c})$
MCDS[82]	8	$O(n^2 \log n)$
MST[19]	2	$O(m \log m)$
Alg-A	2	$O(cm+n \log n)$

Table 5.1 Approaches for solving MHP2C with c groups

The approximation ratios and the time complexities of different MHP2C solutions discussed so far are summarized in Table 5.1. The asymptotic time complexities of MST and *Alg-A* look comparable and their approximation ratios are same as well. However, as we will see in the following section, our experimental studies validates the fact that *Alg-A* outperforms MST (in terms of the number of high radios activated) in the average case.

5.6 Performance evaluation of MHP2C solutions

The analytical studies carried out thus far have revealed the worst case performance of the different solutions. However, a more useful metric in practice would be the average case performance of these algorithms. We investigate the average case behavior of the different solutions through simulation experiments. We use the LEDA graph algorithms library [2] to implement the different algorithms for our simulation.

We use two types of radios for each sensor node. The low power radio has a range of 2.3 units while the high power radio has a range of 5.5 units. The power required to activate the low power and high power radios are fixed respectively at 1 and 29 units. The power values are derived from the standard assumption that the power consumed

by an active during transmission is proportional to r^{β}, where r is the range and β is the path loss exponent. In our setting, we have taken $\beta = 3.98$ [27]. We randomly place n nodes in a square grid. We increase the number of nodes in the network and correspondingly also increase the square area of the network so as to keep the node density constant. With this approach, we observed that the number of low power components created in the graph were almost uniform (variation less than 10%) for a given number of nodes. The number of nodes n were increased from 250 to 1800 in order to vary the number of low power components from 50 to 400.

We study the performance of the following four MHP2C solutions: (1) Steiner tree solution, (2) Minimum Connected Dominating Set (MCDS) solution, (3) Minimum Spanning Tree (MST) solution, and (4) *Alg-A*, which is a greedy solution. The performance of these solutions is studied by varying the number of low power connected components in the network and the average number of high power radios activated per component is observed. The results of this study are shown in Figure 5.5. Solutions with superior performance should have this average closer to 1.00 – higher values indicate poor performance.

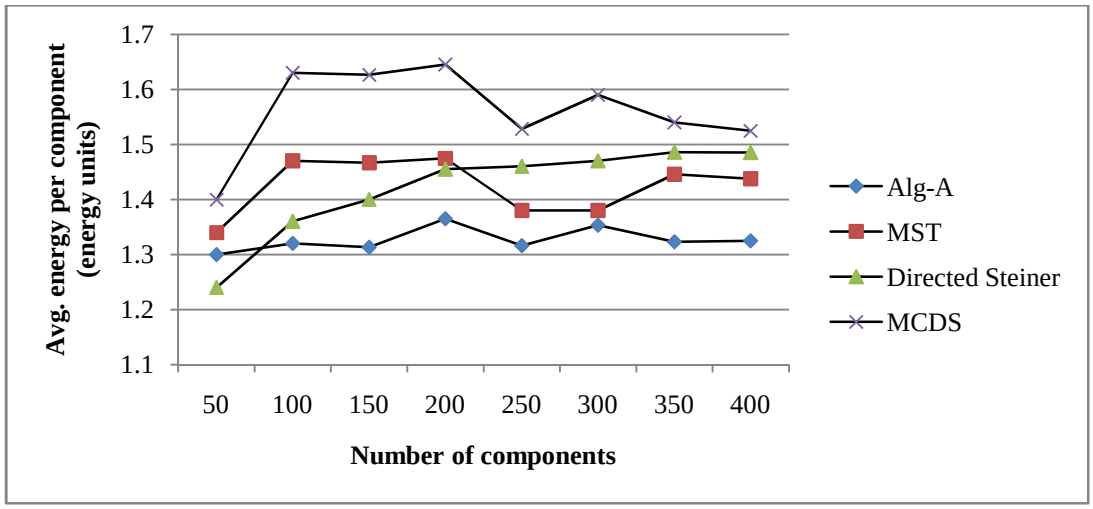


Figure 5.5 Average case performance of different MHP2C solutions

From the graph, we can clearly see that for networks with fewer components, the Steiner tree solution has a good performance. However, as the number of components are increased, its performance degrades and is dominated by *Alg-A* and MST. Between *Alg-A* and MST, we see that *Alg-A* consistently outperforms MST. Therefore, even though these two algorithms have similar worst case performance and time complexities, *Alg-A* might be preferred on account of its average case performance.

5.7 The MP2C and MPKC problems

Having discussed the MHP2C problem, we now take up the other two problems, MP2C and MPKC. To begin with, we discuss the complexities of these two problems.

Lemma 4. The MP2C problem is NP-Complete.

Proof: The MP2C problem contains the MHP2C problem as a special case, wherein the cost of the low power radios is zero. This means that one can turn on all the low power radios to form connected components, without increasing the total energy costs. Solving the MHP2C problem on the resulting network will also provide the answer to

the special instance of the MP2C problem. Since MHP2C is NP-Complete, MP2C is also NP-Complete. ■

Lemma 5. The MPKC problem is NP-Complete.

Proof: The MPKC problem contains the MP2C problem as a special case where K = 2. Hence the MPKC problem is also NP-Complete.■

Recall that in the MHP2C problem, the multi-graph representation of the given multi-radio network can be reduced to a simple graph without any loss in the solution quality. However, a similar reduction in the case of MP2C and MPKC problems will decrease the solution quality. Nevertheless, we still apply such a reduction in order to leverage on the resulting simplicity of the solution. We provide a polynomial time approximation algorithm for the MP2C problem which makes use of the previously discussed *Alg-A* solution for the MHP2C problem.

5.7.1 Approximation algorithms for the MP2C problem

An approximation algorithm, termed *Alg-B* for solving the MP2C problem is given in Figure 5.6. *Alg-B* works as follows. In the first step, we find all the connected components formed by using the low power radios alone. Then, in step 2 we use the previously described *Alg-A* to find the set of high power radios V_H which need to be turned on so as to create a connected graph. At this step, there may be some redundant low power radios which have been turned on in step 1. Such radios are identified and turned off in step 3. For example, if *Alg-A* turns on multiple high power radios within the same component, then these nodes, by virtue of being connected through the high power radios, can all turn off their low power radios (excluding one). However, if any

of these nodes happen to be *cut points* with respect to the low power radios of the component, then clearly their low power radios cannot be turned off.

Lemma 6. The approximation ratio of *Alg-B* is $(2\alpha +1)/(\alpha+1)$, where $\alpha = P_H/P_L$, P_H is the power consumed by the high power radio in active state, and P_L is the power consumed by the low power radio in active state.

Proof: Suppose we have n nodes in the network. In the case of *Alg-B* algorithm, we will turn on a set of high power radios first (step 2). Suppose we turn on h high power radios. The power required for this is hP_H. Notice that steps 3 and 4 do the following – they remove certain nodes from the candidate set of C_i. Effectively, for these nodes, we do not need to turn on the low power nodes since they are connected to the network because of their high power radios already. Let g_i be the number of nodes removed from candidate set S_i (i.e. $g_i = |V_i|-|S_i|$). Let $g = \sum g_i$, $1 \leq i \leq p$. The number of low power radios which need to be turned on will then be $(n-g)$ and the total power required by the low power radios is $(n-g)P_L$. So the total power consumed by all the radios under *Alg-B* is $hP_H + (n-g)P_L$. The maximum value of h is reached when $h = (2c-2)$ and the minimum value of g is reached when $g = 0$. Hence the total power consumed under *Alg-B* is at most $(2c-2)P_H + nP_L$. When $c > 1$, the minimum power required to connect all the components is nP_L+cP_H. It can be easily verified that the resulting approximation ratio of $(nP_L+(2c-2)P_H)=(nP_L+ cP_H)$ is bounded by $(2\alpha+1)/(\alpha+1)$ where $\alpha = P_H/P_L$. ■

1) Find all the components $C = \{C_1, C_2, \ldots, C_k\}$ by turning on low power radios
2) Run *Alg-A* to find the set of high power radios V_H that has to be turned on to connect all the components
3) **For each** component $C_i \in C$ /* Post processing */
a. Let $S_i = \{V_i \mid V_i \in C_i$ be the candidate set of nodes in component $C_i\}$
b. Find the set of articulation points V_a of C_i
c. Turn off all low power radios in C_i

d. Turn on all high power radios in set $V_H \cap C_i$
e. **For each** connected subgraph $G' = (V', E')$ such that $V' \subset C_i$
i. Construct $V_c = \{V_j \mid V_j \in V' \text{ and } V_j \notin V_a\}$
ii. Randomly select $V_s \in V_c$, and $S_i = V_s \cup S_i - V_c$ (i.e. except for randomly selected V_s remove other nodes
in V' from the candidate set C_i)
iii. Using the selected V_s as root construct a spanning tree T_i on S_i
iv. Turn on low power radios of all nodes in spanning tree T_i
f. **End for**
4) **End for**

Figure 5.6 Approximation algorithm *Alg-B* for solving the MP2C problem.

Time Complexity of *Alg-B*. The time complexity of *Alg-A* which is called as a subroutine, dominates the complexity of *Alg-B*. It can be shown that *Alg-B*'s time complexity is O($cm+n\log n$) when c components are created by activating only the low power radios.

Remarks. We note here that a MST based approach can also be used for solving the MP2C problem. Such an approach will be identical to *Alg-B* described in figure 5.6 but for step 2 which will be replaced by the approach described under section 5.4.3. It can be shown that as with *Alg-B*, the total power consumed under the MST approach is at most $nP_L + (2c-2)P_H$. Hence the approximation ratio for the MST based solution for MP2C will also be bounded by $(2\alpha + 1)/(\alpha + 1)$.

One can also develop solutions for MP2C based on the Steiner tree and MCDS solutions. However from the discussions in sections 5.4, 5.5 and 5.6, one can infer that such solutions may not have a superior worst case or average case behaviors in comparison to *Alg-B* and MST. Hence, we do not consider such solutions.

5.7.2 Approximation algorithms for the MPKC problem

A similar approach could be used for solving the MPKC problem, where one could progressively turn on all the lower power radios and keep forming connected components up to the $(K\text{-}1)^{th}$ radio. The algorithm, termed *Alg-C* works as follows. When radios 1 through *i*, $i < K$ are turned on in all the nodes, the network may not be connected but be composed of several components. Let C_q^i denote the connected component q when radios 1 through at most i are activated in all the nodes in the network. To begin with all the C_q^1s and C_q^2s in the network are determined. Any given component C_r^2 in the network will be composed of a subset S_r^1 of C_q^1s. This subset S_r^1 is fed to *Alg-B d*escribed earlier so that the component C_r^2 can be determined such that power consumed by the active radios across all the nodes in C_r^2 to form C_r^2 s is minimized. The process is repeated for other S_q^1s and C_q^2s. Now, the different C_q^3s in the network are determined and the subset S_r^2 of C_q^2 s that comprise a given C_r^3 are identified. These are then fed to *Alg-B* to form the different C_q^3s in the network with minimum power consumption. The above process is repeated until a single C_q^k is output from *Alg-B.*

Lemma 7. The approximation ratio of *Alg-C* is bounded from above by $(1+4\alpha_K)/(1+2\alpha_K)$ where $\alpha_K = P_K/P_1$, P_K is the power consumed by the highest power radio K in active state, and P_1 is the power consumed by the lowest power radio 1 in active state.

Proof: Suppose we have n nodes in the network. Let the power for radios 1, …, K be respectively P_1, P_2, …, P_K. Let C_i denote the number of components in the network formed when radios up to i-1 are turned on in all the nodes. Similar to the discussions

in the proof for lemma 5, the minimum power required for connecting the network will be $P_{min} = nP_1 + C_2P_2 + \ldots + C_KP_K$. The maximum power required under *Alg-C* will be $P_{max} = nP_1 + (2C_2\text{-}2)P_2 + \ldots + (2C_K\text{-}2)P_K$ which is less than $nP_1 + 2C_2P_2 + \ldots + 2C_KP_K$. The maximum value of C_2 is obtained when $C_2 = n$. Let us discuss the maximum possible value for C_3. Recall that radios of type 3 will be turned on only for newly formed components – in other words, if an already existing component formed using radio of type 2 does not connect to even a single node that lies outside the component by turning on radio 3 in any of its constituent nodes, then no node in that component will turn on radio 3. Given the above fact and that maximum value of C_2 is n, the maximum value of C_3 is $n/2$, i.e., components at the level of radio 2 should get paired up. Extending the discussion, we can show that the maximum value of C_i, $2 \leq i \leq K$ will be $n/2^{i-2}$. Also, $C_2P_2 + \ldots + C_KP_K < C_2P_K + C_3P_K + \ldots + C_KP_K$ which is again less than $2nP_K$. From these, we can derive that $P_{max}/P_{min} < (1 + 4\alpha_K)/(1 + 2\alpha_K)$.■

Time Complexity of *Alg-C*. The time complexity of *Alg-C* is also dominated by that of *Alg-A* which is called as a subroutine. Under *Alg-C*, when c components are created by activating only the low power radios, *Alg-A* is invoked on each of these c components no more than K times. Therefore, *Alg-C*'s time complexity is $O(Kcm + Kn\log n)$.

Remarks. Using *Alg-C*'s framework, a MST based approach can also be applied for solving MPKC by replacing *Alg-B* with the MST based MP2C solution. Using a similar argument as shown for the MST approximation for the MP2C problem, we can show

that the MST based approach to the MPKC problem has an approximation ratio of $(1 + 4\alpha_K)/(1 + 2\alpha_K)$.

5.8 Performance evaluation of MP2C and MPKC solutions

As before, we study the average case performance of the MP2C and MPKC solutions through experimentation. When the MP2C solutions are studied, the simulation settings remain the same as in section 5.6. Figure 5.7 shows the performance of *Alg-B* and the MST based solution for the MP2C problem. The number of low power connected components in the network is varied and the average power consumed across all the active radios in a component is observed. A solution that lowers this average value is desired. From the figure, we can clearly see that *Alg-B* consistently outperforms the MST based solution thereby showing that *Alg-B* has a better average case performance than the MST based solution.

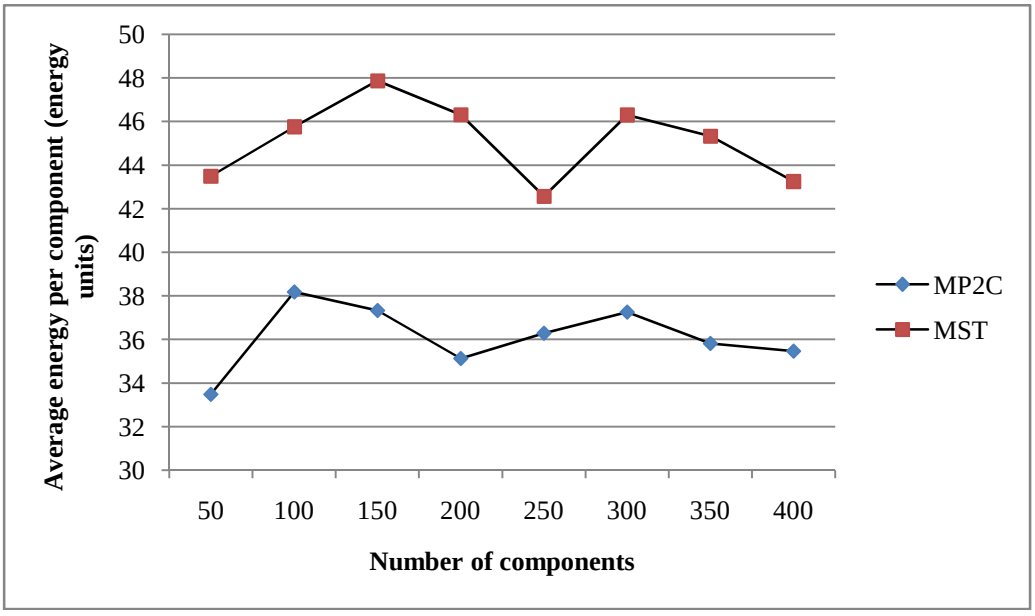


Figure 5.7 Average case performance of different MP2C solutions

We also studied the performance of the two MPKC solutions by setting K = 3. The simulation settings remain same as before, but for the fact that an additional radio is introduced at each node. This new radio has a range of 4 distance units and an energy consumption of 9 energy units. The performance of *Alg-C* and the MST based solution are shown in figure 5.8. From the figure one can clearly see that *Alg-C* maintains the trend by outperforming the MST based solution.

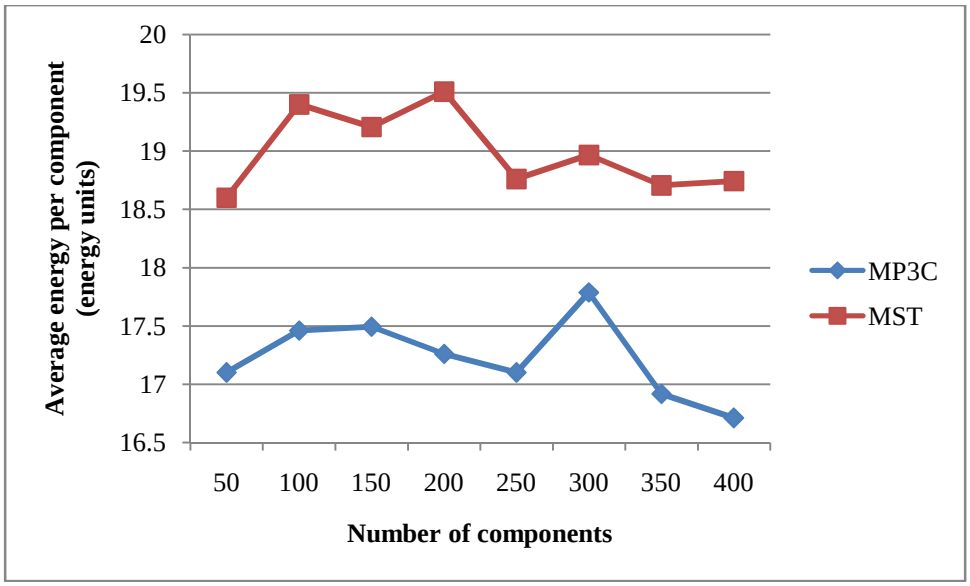


Figure 5.8 Average case performance of different MPKC solutions evaluated for K = 3

5.9 Related works

Multiple radios have many benefits, and they have been exploited for different functions in earlier works. In [74], the authors use dual radios to provide improved data transmission scheduling. In [55], the authors use dual radios with the low power radios exchanging pulse messages which synchronize the other radio used for data communication. In [79], the authors present a hierarchical power management scheme involving radios on sensor motes, PDAs and laptops (the three together is considered as

a single device). In this work, the low power radios wakeup the high power radios in such a way as to elongate the lifetime of the network.

Zhu et al. [91] propose wireless protocols that exploit the knowledge of coexistence of multiple radios to improve the system throughput. Bilstrup et al. [11] divide the network into clusters and use multiple radios for inter-cluster scheduling to break the dependence between local medium access schedules of adjacent clusters. Here the use of multiple radios helps improve the system throughput, while also providing better network connectivity. In [77], the authors propose using one of the multiple radios for resource discovery, which allows a low power radio to be always on for the sake of discovering network resources. The high power radios are turned on only when a wake up message is received. In [67], the authors use multiple radios in a hierarchical radio structure for providing mobility support. In [80], the authors use the low power radios to find suitable end to end paths satisfying certain quality guarantees. Recently, dual radio testbeds are also being increasingly evaluated for their performance benefits [22], [83], [71]. The improvement in system performance measured along various different parameters has shown the feasibility and utility of multi-radio networks. The above works assume that the network remains connected even when all the sensor nodes activate only their lowest power radio. The raison-d'etre for low power radios in most of the existing research is to turn the high power radio on when required to create alternate paths. However, in a general setting, such a requirement on the connectivity cannot be guaranteed. Radios with higher power and longer-range may have to be activated even to make the network connected necessitating the proposed work. While our research may seemingly appear similar to other works that adjust the power levels

of radios for performing topology control [85], [73], it differs from such works in a fundamental way: we have multiple radios at each sensor node (as opposed to single radio in these works) and we selectively activate one or more of them at each sensor; the individual power levels of the radios are not adjusted. In [14], the authors consider linear networks where a random fraction of the nodes in the network have dual-radio functionality and use probabilistic techniques to describe the connectivity of such networks. Our work provides deterministic solutions, is applicable to any given network topology, and also considers the general case of K radios.

5.10 Summary

In this chapter, we studied energy efficient radio activation in wireless sensor networks where each node has $K > 1$ different radios. We showed that achieving optimal radio activation that minimizes the total energy spent by the active radios across all nodes while maintaining network connectivity is NP-Complete for $K > 1$. We proposed four different polynomial time approximation algorithms solving the optimal radio activation problem. Our analytical and experimental studies reveal that the proposed greedy algorithm and the minimum spanning tree solution have the best *worst case* performance while the greedy algorithm has the best *average case* performance.

Chapter 6

6 Energy aware network decomposition techniques

6.1 Introduction

Wireless communication consumes a significant amount of energy, and it is important to minimize the energy costs for communication as much as possible by practicing energy aware routing strategies. This is very important for sensor networks where the energy is an important non-replenishable resource. Routing strategies can increase the network lifetime. Network lifetime is quantified as the number of packets that can be transferred in the network before the source and destination get disconnected from each other [48, 65]. A suitable energy-aware routing strategy for wireless networks is to use those wireless nodes with high energy levels and avoid those with low energy levels. The routing strategies on sensor networks involve the following general steps, a) find routes; b) perform routing; c) update network values and perform step a).

Consider a centralized algorithm wherein a single node (call it *central* node) keeps track of the topology information. The central node will determine the routes (step a) by executing a local algorithm. When a source node requires a message to be routed to the destination, it sends a request to the central node which will provide the entire route to the destination. After the receipt of the information from the central node, the source node can perform routing (step b). Assuming that the source node follows the exact route provided by the central node, the central node can determine the energy changes of the intermediate node (without the intermediate nodes explicitly informing the

central node) and re-compute the routes locally for the next route request. In addition to the energy consumed when packets are routed along the route path, energy is also consumed at intermediate nodes along the path from source to central site and vice-versa for route request and response. This straight-forward algorithm has all the weaknesses of any centralized algorithm such as lack of fault-tolerance and problems associated with hot spots created by request/response information travelling to and from the central node. In fact with repeated route requests it is easy to observe that the neighbors of the central site may quickly lose energy thereby making the central node unreachable and consequently decreasing lifetime. One could choose a new central node and use a simple distributed algorithm such as the distributed depth first search [76] to learn the topology of the network including the node and link information.

Yet another weakness of the centralized algorithm is that for large resource limited sensor networks a single central node may have neither the space capacity to store the entire network nor the computation power to compute the paths in a short period of time, or even enough energy to perform the computation. In this “utopian” model of centralized algorithms all nodes behave perfectly as instructed by the central node and the central node is able to compute real time and accurate information about the energy levels of the nodes in the network. Even if such a utopian model where possible, the reliability of the central node itself (being a sensor, it is equally prone to die, or experience lossy links to neighbors) as well as the lack of scalability of this approach (as more sensors are added, the rapid rise in communication costs are felt acutely at the neighbors of the central node) makes the centralized algorithm nearly impractical for implementing in a sensor network.

Given a distributed system consisting of computational nodes, a distributed algorithm solves a particular problem of interest by exchanging messages among the nodes. In the distributed system each node knows its neighbors by their unique identities and the total number of nodes in the distributed system. A distributed algorithm is evaluated based on the total number of messages exchanged (*message-complexity*) and the time-taken for the completion of the distributed algorithm (*time-complexity*). Depending on the problem to be solved the distributed algorithm must be rerun after a node or link update either on the entire network or a portion of the network. Distributed algorithms are scalable as it does not require a single node to keep track of the entire topology information. The fundamental weakness of the distributed algorithms for sensor networks stems from the fact that after step b) of the routing strategy is completed, the intermediate nodes have new energy levels and now the distributed algorithm to determine routing paths (step a) has to be re-executed. That is after each route request is complete the distributed algorithm is re-run and thereby the message complexity is overwhelmed by the number of route requests that have been completed.

From the above discussion it is clear that the centralized algorithm is message efficient, but ineffective on lifetime as a result of hot spots and other issues of relating to a centralized site. The distributed algorithm addresses the deficiencies of the centralized algorithm but is ineffective in terms of lifetime due to large number of messages required to recompute the routing paths after a completion of a route request. Our goal in this chapter is to introduce a network decomposition approach that will combine both the centralized and distributed approached described above by

decomposing the network into smaller networks (referred to as *clusters*). The centralized algorithm is executed on each cluster and the distributed algorithm is executed on the central nodes (referred to as *cluster head*) of each cluster.

Network decomposition to akin to divide-and-conquer approaches to problem solving wherein, a larger problem is broken into smaller sub-problems and solutions of the smaller sub-problem are combined to arrive at the solution to the larger problem. Network decomposition has been effectively used to solve many problems in sequential, parallel, and distributed environments [21]. Network decomposition techniques have shown to reduce the message complexity of distributed algorithms by (i) decompose the network into a set of connected components, (ii) run a pseudo-distributed algorithm on each connected component (we will call this a *cluster*), and (iii) solve the optimization problem by executing a distributed algorithm involving cluster heads of each cluster. A node that is along the path connecting two cluster heads will only forward messages.

Using network decomposition approaches one can alleviate the problems resulting in having central site. Updates in each cluster are sent to its cluster head. The cluster heads perform a local computation using the topology information as in the case of centralized algorithm. The cluster heads communicate using “meta” data and execute a distributed algorithm to solve the problem at hand. Conceptually since the number of cluster heads is smaller and that fewer nodes will participate in the distributed algorithm the message complexity could be smaller. The above idea has been used to solve many distributed algorithms effectively [4].

Awerbuch and Peleg [5, 6, 7, 8] have published a series of seminal works in the area of distributed algorithms that uses the concept of network decomposition. These works and the work by Linial [51] and Naor and Stockmeyer [64] exploit the concept of "locality" in distributed computations. The concept of locality is that certain functions when locally computed do not affect the global solution. For certain problem the solutions of the local computation can be cleverly combined to obtain global solution. Considering network problems on networks that have been decomposed, certain coloring problem instances can be solved efficiency for the entire network by cleverly stitching together solutions for each cluster.

Several wireless sensor network routing schemes that utilize the concept of "hierarchy" and "clusters" have been proposed before in literature. Our work differs from the existing body of literature in several ways. In existing works, the clusters are defined based on geographical location and/or radio reachability alone. Cluster formation in such works is not directly related to the objective of maximizing the network-life time as done in our work. Also, very few of the existing hierarchical routing schemes have their objective as network life-time maximization. Further, the benefits of using hierarchical routing and intelligent cluster formation on network life-time have not been clearly and quantitatively documented in existing works.

The rest of the chapter is organized along the following lines. In section 6.2 we introduce the widest path problem and its application to improving network lifetime. We present an algorithm to perform widest path routing (or called the maximum residual energy path routing) given a set of clusters. Ideal network decomposition for suits better for network lifetime is described in section 6.3 and a decomposition

algorithm for such decomposition is also presented. In section 6.4, we experimentally validate that network decomposition is the ideal approach to improving network lifetime in sensor networks. We conclude in section 6.5.

6.2 The Widest Path Problem: A Network Decomposition Approach

In this section, we first describe the relevance of the widest path problem to lifetime aware routing on sensor networks in section 6.2.1. Given the need to frequently compute the widest path (or a variant) on the network, section 6.2.2 then provides a network decomposition approach for doing the same. The energy costs involved in the network decomposition approach are discussed in section 6.2.3.

6.2.1 Relevance of widest path to lifetime aware routing

In developing energy aware routing techniques, wireless networks are modeled as graphs wherein, the vertex represents a wireless device and an edge between two vertices indicates that they are in direct communication range of each other. The weight on a vertex indicates the residual energy available at that wireless node and the weight on an edge (u, v) represents the amount of energy required by node u (resp. v) to transmit one unit of data to node v (resp. u). The *residual energy of a path* is defined as the minimum energy level of any node in the path. The max-min routing paradigm suggested in the literature [1, 48, 81] aims to maximize the network lifetime by finding the path where the residual energy is the maximum and forwards packets through this path termed as the *maximum residual energy path*.

Let $G = (V, E)$ represent a wireless network with nodes V and edges E. Let $w(u)$, $u \in V$, represent the available energy at node u. Let c (u, v), $(u, v) \in E$, be the energy

required to transmit a packet from node u to node v. We assume that $c(u, v) = c(v, u)$, for all $(u, v) \in E$.

Let $P(v_0, v_k) = v_0, v_1, \ldots, v_k$, be a path in G. The energy of the path $P(v_0, v_k)$ denoted $e(P(v_0, v_k))$ is given by

$$e\big(P(v_0, v_k)\big) = \sum_{i=0}^{k-1} c(v_i, v_{i+1}) \quad (1)$$

The *residual energy of a path* $P(v_0, v_k)$ denoted $r(P(v_0, v_k))$ is defined as

$$r(P(v_0, v_k)) = min_i(w(v_i)\text{-}c(v_i,v_{i+1})),\ 0 \leq i < k. \quad (2)$$

When a packet is sent along $P(v_0, v_k)$, we need to perform the following *energy decrease operation* on each node along the path except on the node v_k: $w(v_i) = w(v_i) - c(v_i, v_{i+1})$, $0 \leq i < k$. That is, after the packet is sent by a node, the energy level of the node is decremented by the amount of energy required to send the packet.

Let G_0 be set to the initial network G. Assume that $P_0(s, t)$ is a path in G_0. Now after routing a single packet along the path $P_0(s, t)$ and applying the decrease operation we obtain a new network G_1. In the network G_1 the edge weights are the same as in G_0 but the node energy levels are different. If a node u's energy level becomes 0 after the decrease operation, node u and its associated edges $(u, v) \in E$ as well as $(v, u) \in E$ are removed from the network. For the second packet we can again find a path $P_1(s, t)$ in G_1 and the process continues until there exists no path between s and t in some network G_k. That is, we can send at most k packets from s to t before the network is disconnected. The goal of the *network lifetime problem* with respect to a source s and

destination t is to find paths $P_0(s, t)$, $P_1(s, t)$,… $P_{k-1}(s, t)$, such that the value of k is maximized. Here we would like to point out that while our goal is to maximize the network lifetime, it has been shown that computing the value of k is NP-hard [65]. The work by Mohanoor et. al [62] provide an algorithm for computing the widest path and investigate prior work in the area of lifetime for sensor networks. The algorithm described in [62] converts the original network into residual energy network as given below.

We will begin by outlining a description of the widest path which is used in the solution of Mohanoor et al [62]. The graph G is modified into an energy graph $EG = (V, E')$ as follows. We leave the vertices intact but replace each single undirected edge in G with two directed edges. The weight of a directional edge in EG is made equal to the difference between the originating node's energy level and the transmission cost along the edge. This is also the residual energy of a node as defined in Li et al [48]. In Figure 6.1 (a) we have shown an example wireless network and in Figure 6.1 (b), the corresponding energy graph.

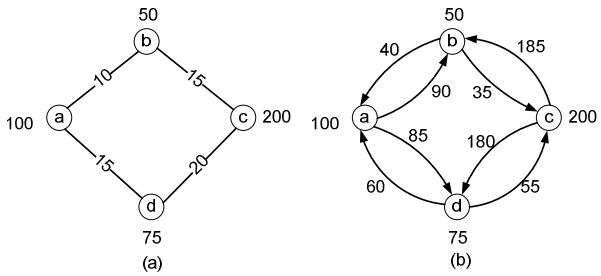


Figure 6.1 (a) A graph showing energy levels at nodes and energy required to transmit at each edge. Figure 6.1 (b) shows the corresponding energy graph.

After obtaining the energy graph an algorithm similar to the shortest path algorithm is executed to obtain the maximum residual path.

6.2.2 A network decomposition approach to distributed widest path problem

Our goal in this section is to describe an algorithm that takes advantage of the clusters that are formed as a result of the network decomposition. In the next section, we will construct the clusters taking into account lifetime issues and impacts. The following algorithm provides a way to find the widest path on a network. As we will describe later, what we compute here is an approximation of the widest path – which we term as a 'weak' widest path.

a) Form a set of clusters $C_1, C_2, \ldots, C_k$ from a given network $G=(V, E)$. Let $h_i \in C_i$ be the cluster head, $1 \leq i \leq k$. With each h_i as the root construct a spanning tree distributedly that includes only the nodes in C_i. Learn the topology of the network induced by the nodes in the cluster C_i. Let the network learned be $G(C_i)$. The network $G(C_i)$ is stored at h_i. Compute all-pairs widest path on $G(C_i)$ at h_i.

b) For each cluster C_i find all its neighboring clusters. Two clusters C_i and C_j are neighbors if there exists a node $v_i \in C_i$ and $v_j \in C_j$ such that $(v_i, v_j) \in E$. Let $N[C_i]$ be the neighbors of C_i and let $d(C_i, C_j) = \max_{(v_{ij}, v_{ji}) \in E} d(v_{ij}, v_{ji})$ where $v_{ij} \in C_i$ and $v_{ji} \in C_j$, and $d(v_{ij}, v_{ji})$ represents the weight of edge (v_{ij}, v_{ji}). (In this case, the weight is the residual energy of the edge in the energy graph as calculated in Figure 1(b)). Both $N[C_i]$ and $d(C_i, C_j)$ can be determined during the learning step as mentioned in step a). Let $D^i[]$ be a distance vector stored at cluster head h_i. Initially, for each neighbor $j \in N[C_i]$, $D^i[j].d = d(C_i, C_j)$, otherwise it is set to zero.

c) Execute a distance-vector distributed algorithm using the distance vectors $D^i[]$ stored at cluster heads h_i. Let us denote $D^i[k].nh = C_j$ to be a next-hop neighbor of cluster C_i along the widest path to cluster C_k. Let $D^i[k].n_0 = v_{ij}$ and $D^i[k].n_1 = v_{ji}$ be the nodes identified in step b). A message from one cluster head h_i travels to a neighboring cluster head h_j along precomputed paths between the two cluster heads. The vector $D^i[]$ is sent from h_i to v_{ij} along this precomputed path.

Since only the weight on the edges joining two clusters (i.e. the bridges) is considered for computing this widest path, it is quite easy to see that what we are computing here is only an estimate of the widest path between two nodes belonging to different clusters. Hence we refer to it as a 'weak' widest path. However, Mohanoor et al.[62] have shown that while it is important to choose high residual energy paths, it is not necessary to use the 'widest' path to extend the lifetime – a sufficiently 'wide' path will suffice, a fact which is also borne out by other works in the literature ([1], [48], [81]).

Once the distance vector algorithm has been executed distributedly, we would then use the results of this algorithm for the energy aware routing. The following steps show how the actual routing is performed.

Let $s \in C_i$ and $t \in C_j$.

1) If C_i and C_j are neighbors, then follow steps below:

    a. Node s will ask h_i the widest path from s to v_{ij}.

b. Node s will next send the packet along the widest path to v_{ij} indicating to it that it should send it to v_{ji} and it is meant for t which is in C_j.

c. The node v_{ji} upon receiving the message will request h_j for a widest path to t.

d. Upon receiving the widest path v_{ji} will send the message to t.

2) If C_i and C_j are not neighbors, then let $D^i[j].nh=C_k$. If $D^i[k].n_0 = s_k$ and $D^i[k].n_1 = t_k$, route the packet from s to s_k and make $s = t_k$, and $i = k$. If i and j are now neighbors, execute steps 1(a)-1(d), else execute step 2.

6.2.3 Energy costs

The energy costs incurred by using the network decomposition technique is two-fold: the cost of updating the energy levels of all nodes within a cluster to the clusterhead, and the cost of executing the distributed algorithm among the clusterheads. Furthermore, the cost of updating the energy levels of all nodes within a cluster is assumed to be a one-time cost. Since we presume that a sender will use the route suggested by the cluster-head, it is possible to compute the decreased energy levels of all the nodes along the route. This is true of all the intermediate nodes which are involved in forwarding packets – our scheme ensures that a clusterhead is aware of the instantaneous energy level of every node within its cluster which is used for any given *s-t* request. As we have precomputed routes along which a cluster node will communicate with its clusterhead, the energy cost involved at each node in this request-reply process is also known to the clusterhead. The energy cost of executing the distributed algorithm is not a one time cost, as we can clearly see that the bridge with

the highest energy level would keep changing as we keep forwarding packets. When a clusterhead becomes aware of the fact that the bridge link chosen as the highest energy edge is no longer the one with the highest energy, it initiates the distributed algorithm between the clusterheads once again. The final energy cost incurred by the network decomposition technique must take into account both these costs.

6.3 Network Decomposition for Improving Lifetime

In the previous section we have presented an algorithm to compute widest path given the arbitrary network decomposition. The following are some of the desirable properties of any decomposition.

a) The diameter of each cluster should be smaller. The diameter of a cluster *S* denoted diam(*S*) is the longest shortest path value in the subgraph induced by nodes in *S*. Given that some node in the cluster will be chosen as the cluster head, the number of hops required to communicate with the cluster head should be smaller. Reducing the number of nodes in the node-cluster head path will result in energy saving thereby increasing lifetime.
b) The number of nodes in each cluster should be bounded. Clearly, the size of the cluster and the number of clusters is a tradeoff between the energy consumption as part of the intra-cluster centralized and inter-cluster distributed computations.
c) The number of clusters should be smaller. Keeping the number of clusters smaller would reduce the inter-cluster communications required by the distributed computation. Finding a minimum number of clusters satisfying the diameter constraint is same as the partition into cliques problem [23] which is NP-complete.

d) The number of bridges from each cluster should be high. If there are more bridges then the inter-cluster distributed computations can use more paths to the cluster containing the destination node and thus improving lifetime. It can also be said that the anchor nodes connecting the bridges should have high residual energy since it is going to see more traffic as part of the distributed computation and widest path routing.

A distributed algorithms along the lines described in [8] can be used for obtaining a desirable decomposition. The distributed algorithm presented in [8] starts constructing a breadth-first search tree in a distributed fashion. As the breadth-first search tree grows we can stop further exploration if either the desired depth bound or size bound has been reached. The choice of which node to explore (or to be added next) will provide a tradeoff between a)-c) of the desirable properties.

A formal centralized algorithm for the network decomposition taking into account the desirable properties above is given in Figure 6.2. In the algorithm Decompose, the parameters k_1 and k_2 specify bounds on the size and diameter of each cluster, respectively. The parameter f defines a fraction that is used to select the set of high degree nodes (to increase the number of bridges). For notational purposes let $\rho(v)$ denote the either the degree or residual energy of a node v in the network. It can be shown that the time-complexity of the above algorithm for a n node and m edges network to be $O(m\log n)$.

The basic idea behind the algorithm is to start with an arbitrary vertex and add nodes with low value of ρ one at a time. As we add nodes to a cluster we have to make

sure that the size and diameter constraints are satisfied. After we add a sufficient number of nodes with low value of ρ we have to start adding nodes with high value of ρ. We will be adding nodes to the cluster in such a way that the cluster is a connected. It is quite possible that after adding the nodes with low values of ρ, the diameter of the cluster could reach the desirable diameter, thereby preventing the addition of nodes with high values of ρ. Clearly, the order in which nodes are added is going to dictate the size and diameter of the cluster and eventually decide the number of clusters that are formed. Since the partition problem is NP-complete (as observed previously), our goal is to provide a better heuristic. Among the nodes that can be connected to the existing connected cluster, we will choose low or high ρ value nodes, depending on the current size of the cluster. One could also choose high ρ value nodes after the diameter has reached a particular value.

Algorithm Decompose (G, k_1, k_2, f)
begin
1. Set $P \leftarrow \Phi$
2. Select an arbitrary node $v \in V$ and $V \leftarrow V - \{v\}$
3. **while** ($V \neq \Phi$) **do**
4. Set $S \leftarrow \{v\}$
5. $L \leftarrow$ construct a min-max heap with ρ values of nodes which are immediate neighbors of v ($N(v)$)
6. **while** ($|S| < k_1$) **and** (diam(S) < k_2) **do**
7. **if** ($|S| \times f < k_1$) **then**
8. $u \leftarrow L$.deleteMin()
9. $S \leftarrow S \cup \{u\}$ and $V \leftarrow V - \{u\}$
10. **for** each node $x \in V$ and $x \in N(u)$ **do**
11. L.insert(x) based on ρ (x)
12. **end for**
13. **else**
14. $i \leftarrow k_1 - |S|$
15. **while** ($i > 0$) **and** (diam(S) < k_2) **do**
16. $u \leftarrow L$.deleteMax()
17. $S \leftarrow S \cup \{u\}$, $V \leftarrow V - \{u\}$, and $i \leftarrow i - 1$
18. **for** each node $x \in V$ and $x \in N(u)$ **do**
19. L.insert(x) based on ρ (x)
20. **end for**

```
21.          end while
22.      end if
23.      end while
24. P ← P ∪ S
25. Select a node v ∈ V such that it has maximum number of neighbors in G(P) (the
26.  graph included by  the vertices in P)
27.      end while
28.      return P
end
```

Figure 6.2 Network decomposition algorithm

The algorithm Decompose described in Figure 6.2 provides a suitable network decomposition strategy that results in clusters which satisfy the multiple and sometimes conflicting constraints – for example, we would like to bound the size of the cluster but also restrict the total number of clusters. Clearly, some tradeoffs are involved in the construction of suitable network decompositions. Besides, one may also add additional hierarchical strategies to find a balance between the number of clusters (and thus the number of messages used for the distributed algorithm) and the number of nodes per cluster (update costs after routing). It must be noted that the Decompose strategy may produce paths (for example widest paths) quite different from a centralized strategy. However, our argument here still holds since the network lifetime is decided by a combination of the paths computed and the strategy used for computing the paths (e.g. centralized vs Decompose). The possible inaccuracy of the paths is compensated by a reduction in the amount of messages used in its computation.

6.4 Experimental Evaluation

We performed preliminary evaluations of our idea by using the LEDA graph algorithms library [31]. The main objective of our simulation study is to show the feasibility of the concept and especially to compare the performance of the purely distributed, centralized and network decomposition schemes.

We generated a random graph with 250 and 500 nodes and created components (clusters) by assigning both a low power as well as a high power radius to each node. When only the low power radius is used to create edges, we get a set of disconnected components. Now we turn on all the high power edges also, and take care to remove multiple edges between nodes by removing the high power edge between two nodes if a low power edge already exists between them.

All the nodes which are boundary nodes, i.e. which have an edge to a different cluster if their high power edges are used, are used as anchor nodes. All internal nodes are assigned low energy levels of 20 units and all the anchor nodes are assigned high energy levels of 50 units each. Note that when the network decomposition algorithm is executed, it will produce clusters with similar properties after its completion. For consistency, the energy required to communicate between two neighboring nodes is set as 0.001 x r^3 where r is the high power radius value.

When we ran the purely distributed algorithm, the amount of energy required to simply exchange sufficient number of messages for algorithm termination, which is $2|V||E|$ in the worst case, leads to energy depletion at such a rapid rate that the algorithm itself cannot be executed more than a handful of times. Hence the results of the purely distributed scheme are omitted.

Our preliminary results for the two different graph sizes n = 250 and n = 500 given in Figure 6.3 show that the network decomposition scheme indeed outperforms the utopian centralized scheme in terms of the number of packets (lifetime) which can be routed before network disconnection.

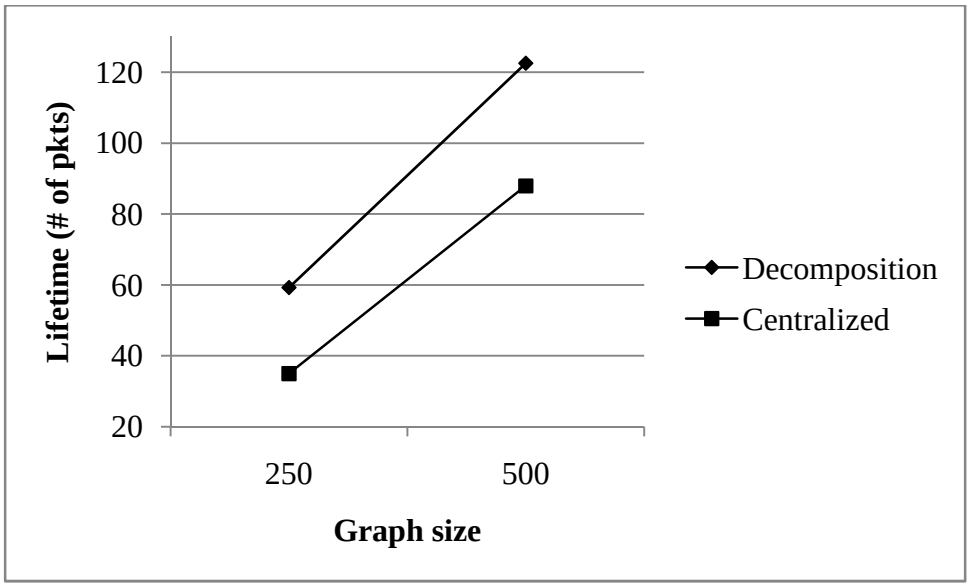


Figure 6.3 Lifetime of decomposition vs centralized algorithm

6.5 Summary

We have presented a lifetime aware network decomposition approach for executing distributed algorithms on wireless sensor networks. The utopian centralized model suffers from the problems of reliability and scalability, as observed earlier. In addition, in an energy constrained sensor network, it also leads to the creation of hotspots and hence leads to smaller lifetimes for the sensor network. By decomposing the network into clusters and executing the distributed algorithm among the cluster heads, we avoid hotspot creation and show how such a technique can lead to improved lifetimes for sensor networks.

Chapter 7

7 Quality of Information (QoI) metrics for knowledge centric sensor networks

7.1 Introduction

A primary necessity for sensor network deployments is to be able to collect data about environmental (and other) phenomena under observation and transform it into useful, actionable knowledge. However, sensor networks due to their resource constrained nature have some key differences from general communication networks (such as the World Wide Web, and corporate intranets). The differences are fundamental, and hence we use the term message centric networks to refer to big, powerful networks such as the WWW and corporate intranets, and knowledge centric networks to refer to sensor networks which usually lie on the other end of the spectrum in terms of scale.

Unit of atomicity – The unit of atomicity can be defined as the 'indivisible' unit of information which still retains semantics. In a knowledge centric network, where combining information is encouraged and loss of information is tolerated, the unit of information is the aggregate knowledge rather than the individual message.

Resource assumptions – the simple act of resending a message is commonplace (and even vital for everyday tasks such as browsing the internet) on a message centric network, where we can make assumptions of virtually unlimited resources. Resending a single message would require careful planning on a resource constrained sensor network, where minimal assumptions are made about the availability of resources.

Data gathering - Nearly all messages generated can and is usually stored or collected in a message centric network, while that is neither a requirement nor a prudent choice on a knowledge centric and resource poor sensor network.

Data dispersal – Data dispersal refers to the replication of the same data and its dispersal over multiple media and devices (such as backing up important files on to a USB drive, a backup disk, and online storage). In a message centric network such as the internet, data dispersal is common and quite useful. Such data dispersal would be costly on a typical sensor network.

Search techniques – this is perhaps a vital difference and a key motivation for the QoI strategy. Any data collected, in order to be made useful, needs to be analyzed and processed. This would usually require doing a search over the data at some point in time. In a message centric network, due in large part to the resource rich nature, the exponential growth of data is tolerated and search techniques evolve to deal with the rate of data generation. We call this the “store and search” strategy. In a fundamentally resource constrained sensor network, the rate of data generation and transmission is controlled by using a top down strategy where the search comes first - in terms of usefulness of data collected, i.e. the “Quality of Information” . Here we first search for what needs to be stored - and hence we “search and store”.

Hence the transformation of data into information (or knowledge) requires a more top down approach which can balance information needs and resource utilization rates. If we begin by defining our information needs (i.e. specifying the Quality of

Information requirements), we would be able to better utilize the often non-renewable resources of a sensor network.

7.2 Motivation for QoI-aware data collection strategies

We believe that adding the Quality of Information (QoI) as another dimension will greatly benefit the knowledge which can be extracted from a sensor network. Mapping the aspects of QoI to different kinds of sensor network applications will allow the user to more clearly specify what he or she wants from the sensor network deployment. By providing a framework to deliver what the user wants, we give more flexibility to the user for defining his/her needs and to understand and analyze the tradeoffs involved.

The most important benefit of the QoI approach to routing on sensor networks is the explicit knowledge of the various tradeoffs involved, which leads to higher quality of data collected from the sensor network. The explicit use of QoI attributes provides a considerable variety of options for data collection.

A second benefit of the QoI approach is a better utilization of network resources. In many scenarios, the use of QoI for specifying the requirements for the data collection process will actually allow for better utilization of network resources than the case where QoI is not considered. For example, we may wish to collect information from highly relevant sensor nodes. We expect to find a fair degree of redundancy in the network; so many sensors could possibly satisfy the relevance constraint. We may choose only a few sensors among them for the data collection task. The sensors chosen may have higher residual energies, and thus we could perform the data collection in an energy-balanced fashion. We could also select sensors which are closer to the sink,

hence reducing the latency of data collection. We can clearly see that using the QoI approach (in this case, specifying that the user is interested in the 'relevance' of the data) allows us to utilize network resources far more effectively while also satisfying the end user requirements.

Some questions could be raised as to what can be categorized as a QoS attribute and what can be categorized as a QoI attribute. We propose a rule of thumb that QoS refers to "objective" attributes whose value remains independent of the interpretation. As an example, the bandwidth of a link has a standardized definition so that given the unit of measurement, everyone will measure the same value for the bandwidth. On the other hand, QoI refers to "subjective" attributes whose value would depend on the specification as well as the interpretation. Some QoI attributes could be defined as graph problems (e.g. our definition of density) which will be objective, but there are also other attributes (for e.g. the relevance of data) which will be fairly subjective and application specific.

7.3 A brief review of QoS models on sensor networks

We now present a brief overview of the literature which pertains to several qualitative issues that occur in data collection for sensor networks. The primary goal for the works we cite here is to satisfy QoS requirements under an additional constraint on the resource of the network (usually energy). However, we argue that they cannot be termed as QoS since they are more dependent on the information than the service offered by the network. Though the works cited call these as QoS problems, they are in reality QoI problems or a combination of QoI and QoS. In Table 7.1, we present some sensor network QoS problems considered in the literature.

In their paper on finding a predictive quality control strategy for wireless sensor networks, Liang et al. [49] use the number of active sensors as a measure of the QoI, since it dictates the spatial resolution of the sensed parameters. They optimize for the number of active sensors so as to achieve optimal lifetime (energy) for the network. Perillo and Heinzelman [66] aim for a reliable description of the environment as the QoI attribute, while simultaneously making the network energy efficient and ensuring that it meets the bandwidth constraints. Gundappachikkenahalli and Ali [26] propose the AdProc framework, where the criticality of information (QoI) is traded off against the latency and the energy requirements. Mingming Lu et al. [58] address the problem of maximizing the network lifetime while maintaining target coverage (QoI).

Wu et al. [88] attempt to turn off sensors making redundant measurements (redundancy is the QoI) while they attempt to reduce the network energy consumption. Kay and Frolik [43] optimize for the network spatial resolution (QoI) and control the network so that sensors participate equally so as to conserve energy. Jin Zhu and Papavassiliou [90] propose a framework called Resource Adaptive Information Gathering (RAIG) that can aggregate data on the fly by making suitable tradeoffs among latency, energy and quality. They do not define the quality explicitly, and their analysis is based on sensors having knowledge of the quality of the data which is transmitted or forwarded. Delicato et al. [20] describe a strategy for managing the duty cycle of sensors, by selecting different sets of nodes to be active at different times. The nodes with higher residual energy and greater relevance (QoI) to the application are kept awake.

Reference	QoI attribute(s)	QoS attribute(s)	Resource optimized
Liang[90]	Number of active sensors as a measure of QoS	N/A	Maximize network lifetime
Perillo[42]	Reliable description of environment	Bandwidth	Maximize network lifetime
Gundappachikkenahalli[26]	Criticality of information	Latency	Better energy distribution
Mingming Lu[58]	Coverage of target	N/A	Maximize network Lifetime
Wu[88]	Degree of redundancy	N/A	Maximize network Lifetime
Kay[43]	Spatial resolution	N/A	Conserve total Energy used
Jin Zhu [90]	Sink specified quality	Latency	Energy savings per Sensor due to aggregation
Delicato [20]	Target coverage, aggregation	Data Acquisition Rate	Increase network Wide residual energy

Table 7.1 An overview of QoS approaches for sensor networks

7.4 Benefits of QoI aware data collection

Not all information is equally important, and *preferential treatment of this information can help us use resources optimally*. The knowledge which can be extracted determines the information gathering process. However, knowledge is subjective, and information which is useful for some applications may not really be very useful for others. For

example, we broadly classify data delivery models on sensor networks as query driven, event driven and continuous delivery [17]. In the continuous delivery model, we are interested in data which is spatially and temporally diverse, since we do not always know exactly what we are looking for. In the event based model, we seek information which is at least sufficient to track an event, and preferably allows a deep analysis of the event. This information may not be spatially or temporally diverse, but we may expect quick notification and perhaps use techniques like multipath routing for maximizing the effective bandwidth. With the query based model, the information which is processed is dependent on the kind of query which is asked. Such a model might support specific techniques like aggregation which is a resource friendly approach when answering queries.

Here we consider the density, a Quality of Information attribute representing the spatial resolution of the sensors reporting their data, and show how we can improve the utilization of network resources by careful choices of the sample set. In their paper describing a vision for the worldwide sensor web [9], the authors mention the following as important outstanding issues for data management in such a network:

a) Data ingest – the calibrate, gap-fill and regrid process which would allow easier multi-dimensional querying of sensor data
b) Data exploration, analysis and visualization – for these advanced usage scenarios, ascertaining and ensuring data quality is a major problem facing embedded sensing
c) Statistical modeling of sensor data – this is one of the most ubiquitous processing tasks performed on sensor data

Suppose we consider these as rough guidelines for choosing a set of sensors which need to report their data to a sink in the network. Let us also assume the sink has knowledge of the network topology. So we would like to sample data from sensors which would satisfy the following requirements

1) Provide adequate coverage of the network – this would help in tasks b) and c) mentioned above
2) Avoid holes in the data – this would help us with the regrid process of task a)
3) Have a way to verify the accuracy of data collected by reconfirming with data values from the neighbors – this would improve the results in tasks b) and c)
4) Provide for sufficient redundancy of data in case step 3) above indicates a faulty sensor - but do not oversample

We can see straightaway that reducing the number of sensors sampled will bring us cost savings in terms of energy cost and thus sensor lifetime. Besides, the redundancy of data collection will help avoid going back and repeating the data collection (in case some readings are faulty) and the adequate coverage of the network may also help with future data collection requests ("I have data collected from region A which is only 5 minutes old, and I could reuse that information for my new query"). Thus we can see that using a QoI approach to data collection provides manifold benefits in terms of information utility obtained per unit resource spent.

We provide the following definition of the density (a subjective definition, as one would expect) – choose a set of sensors such that every sensor has at least *x* % of its *k*-hop neighbors chosen in the sample set (the sensor itself will also be included in this

list of neighbors). We can observe that this provides us with a way of increasing and decreasing the value of x according to requirements and the needs of the end user application.

As a graph problem, this translates to the minimum multi-dominating set problem discussed in [37]. Suppose the neighborhood includes all nodes within k hops. We add edges between each node in the graph and all its neighbors up to k hops to create the modified graph G_1. On this modified graph, we can find the minimum multi-dominating set by setting $r(v)$ to be the $x \times \deg(v)$ on the modified graph G_1.

7.5 Summary

Sensor networks differ in some fundamental respects from Internet model networks. A QoI approach towards data collection would serve to maximize (or at least improve) the resource utilization as a function of the knowledge extracted from these networks.

This strategy could also have some downsides. A QoI approach seems to require more centralized strategies for data collection or decentralized data collection mechanisms which still satisfy QoI definitions (which would be harder to implement than less selective data collection strategies). Another issue to consider is that collecting data based on some particular QoI attribute may turn out to be a poor choice if the application requirements change.

The QoI approach would need further refinements for developing into a science. The new perspective on QoS is likely to lead to new work on redefining QoS parameters, such as collective and aggregate QoS parameters as mentioned in [17]. A major issue is network learning – how to obtain knowledge of global state (if

necessary), who (and how many) need to obtain this information and the frequency of such network learning – are all important questions in this context.

Chapter 8

8 Conclusion and future work

The primary purpose of a wireless sensor network deployment could be stated as the collection of timely, actionable data about the phenomenon under observation. The resource constraints in a wireless sensor network make the following question highly relevant: how can we gather this data from sensor networks in such a way as to maximize the utilization of the networks' resources? It is clear that the utilization is a fairly subjective concept and varies according to the application scenario. However, this question helps us focus on the process of data collection in a more systematic way. In this dissertation, we discussed questions on resource utilization along multiple dimensions.

By solving a series of graph and path problems, we initially find optimal and near optimal methods of resource utilization independent of any subjective constraints. A result of our findings is a set of core principles which lead to an improvement in resource utilization – these principles can then be utilized and combined in various different ways to collect the data needed and to specify the constraints in a resource friendly manner.

The core principles arising from this work can be summarized as follows

- If our goal is to extend the lifetime of a sensor network to collect the largest amount of data possible, using a multi-metric shortest path called as the shortest widest path, as well as close derivatives, is crucial.
- If our goal is to optimally utilize bandwidth available in a multi-hop wireless network, we must focus on a strategy of finding paths which are 'interference aware'. Unlike earlier work which primarily concentrated on link and node scheduling for this problem, our path scheduling approach produces superior throughput at very reasonable computational costs.
- If our goal is to execute distributed algorithms for finding paths in wireless sensor networks, we must be respectful of the typical packet size in a wireless sensor network, which is currently of the order of tens of bytes. This puts an impediment on developing distributed path algorithms which transmit large sized messages. Exploiting the work of low bit complexity distributed algorithms provide a way around this impediment.
- As individual sensors become more powerful and start carrying multiple radios, the problem of activating the radios in an energy aware fashion will turn out to be critical. Our work on radio activation provides additional insight into this problem and shows that the essential question is one of finding such topologies where high power radios form high degree clusters so that the number of nodes connected per high power radio activated is fairly high.

Our goal is to maximize the extraction of knowledge by intelligent resource utilization. This in turn requires a specification of subjective constraints which need to be applied during the data collection such that the data will yield good 'knowledge'. An

important issue which arises in this context is the level of abstraction presented to the end user of the network. The end users are generally specialists with perhaps a good amount of domain specific knowledge but not necessarily trained in the nuances of technical details of wireless sensor devices. However, extracting knowledge from these networks in a resource efficient manner requires considerable amount of such technical knowhow. While early work in this field used lower level abstractions with considerable technical expertise expected from the end user, the need of the hour is to find ways to push this level of abstraction up – in other words to provide a higher level of abstraction and shield the end user from requiring intricate knowledge of wireless sensor devices.

The identification of the core principles described earlier represents an important step forward in pushing the abstraction of the network representation towards higher levels. This frees the end user to make decisions based on application requirements without getting mired in the technical details. We also described a set of quality measures (used during data collection) which can be applied in monitoring scenarios so as to allow us to obtain a high amount of ‘knowledge’ with efficient resource utilization.

Bibliography

[1] Banerjee, S. and Misra, A., "Energy Efficient Reliable Communication for Multi-hop Wireless Networks," to appear in the Journal of Wireless Networks (WINET).

[2] "LEDA," URL - http://www.algorithmic-solutions.com/leda/index.htm

[3]"Towards 2020 Science," located at http://research.microsoft.com/towards2020science/downloads/T2020S_ReportA4.pdf as on 22 February 2007.

[4] Afek, Y.and Ricklin, M. 1993., "Sparser: a paradigm for running distributed algorithms,". J. Algorithms 14, pp. 316-328, 2 (Mar. 1993).

[5] Awerbuch, B. and Peleg, D., "Efficient distributed construction of sparse covers," Technical report CS90-17, Weizman Institute of Science, Dept. of Computer Science, July 1990

[6] Awerbuch, B. and Peleg, D., "Network synchronization with polylogarithmic overhead," In Proc. Of the 31st IEEE Annual Symp. On Foundation of Computer Science, pp. 514-522, Oct 1990

[7] Awerbuch, B. and Peleg, D., "Sparse partitions," In Proc. Of the 31st Annual Symp. On Foundation of Computer Science, pp. 503-513, Oct 1990

[8] Awerbuch, B., Goldberg, A., Luby, M. and Plotkin, S., Network decomposition and locality in distributed computation. In Proc. Of the 30th IEEE Annual Symp. On Foundation of Computer Science, pp. 364-369, Oct 1989

[9] Balazinska, M.; Deshpande, A.; Franklin, M.J.; Gibbons, P.B.; Gray, J.; Nath, S.; Hansen, M.; Liebhold, M.; Szalay, A.; Tao, V., "Data Management in the Worldwide Sensor Web," Pervasive Computing, IEEE , vol.6, no.2, pp.30-40, April-June 2007

[10] Bienstock, D., "On the complexity of testing for odd holes and induced odd paths," Discrete Math., v. 90, no.1, pp.85-92, Jun.1991

[11] Bilstrup, U.; Bilstrup, K.; Svensson, B. and Wiberg, P. A. "Using dual radio nodes to enable quality of service in a clustered wireless mesh network," in IEEE ETFA, pp. 54–61, 2006

[12] Buragohain,C., Suri, S., Tóth,C.D., Zhou,Y.: "Improved Throughput Bounds for Interference-Aware Routing in Wireless Networks," COCOON 2007, pp. 210-221, 2007

[13] Burrell, J.; Brooke, T.; Beckwith, R., "Vineyard computing: sensor networks in agricultural production," Pervasive Computing, IEEE , vol.3, no.1, pp. 38-45, Jan.-March 2004

[14] Cavalcanti, D.; Gossain, H. and Agrawal, D. "Connectivity in multi-radio, multi-channel heterogeneous ad hoc networks," in IEEE PIMRC, vol. 2, 2005, pp. 1322–1326 Vol. 2.

[15] Chang, J. and Tassiulas, L., “Maximum lifetime routing in wireless sensor networks,” IEEE/ACM Transactions on Networking, v.12, no.4, pp. 609-619, 2004.

[16] Charikar, M.; Chekuri, C.; Cheung, T-Y.; Dai, Z.; Goel, A.; Guha, S.; and Li, M. “Approximation algorithms for directed Steiner problems,” in Ninth annual ACM-SIAM symposium on Discrete algorithms (SODA), pp. 192–200, 1998.

[17] Chen, D. and Varshney, P.K., “QoS Support in Wireless Sensor Networks: A Survey,” Proceedings of the 2004 International Conference on Wireless Networks (ICWN 2004), Las Vegas, Nevada, USA, June 21-24, 2004.

[18] Chlebus, B. S., Gasieniec, L., Gibbons, A., Pelc, A., and Rytter, “Deterministic broadcasting in ad hoc radio networks,” Distributed Computing 15, 1 pp.27-38, Jan. 2002.

[19] Cormen,T. H., Leiserson, C. E. and Rivest, R. Introduction to Algorithms. MIT Press, 1990.

[20] Delicato, C., Pires, F., Rust, L., Pirmez, L., De Rezende, J., “Reflective middleware for wireless sensor networks,” Proceedings of the ACM Symposium on Applied Computing, v.2, Applied Computing 2005, pp.1155-1159, 2005.

[21] Derbel, B.; Mosbah, M.; Zemmari, A., “Fast distributed graph partition and application,” Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International , pp. 25-29 April 2006.

[22] ElRakabawy, S.M.; Frohn, S. and Lindemann, C. “ScaleMesh: A Scalable Dual-Radio Wireless Mesh Testbed,” in SECON Workshops ’08. 5th IEEE Annual Communications Society Conference on, pp.1-6, 2008.

[23] Garey, M. R. and Johnson, D.S. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1990.

[24] Garg, N.; Konjevod, G. and Ravi, R. “A polylogarithmic approximation algorithm for the group Steiner tree problem,” Journal of Algorithms, pp. 66–84, 2000.

[25] Guha, S. and Khuller, S. “Approximation Algorithms for Connected Dominating Sets,” Algorithmica, pp. 374–387, 1998.

[26] Gundappachikkenahalli, K., and Ali, H.H., “AdProc: an adaptive routing framework to provide QoS in wireless sensor networks,” Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks as part of the 24th IASTED International Multi-Conference on Applied Informatics, pp.76-83, 2006.

[27] Gupta, P. and Kumar, P. R. “The capacity of wireless networks,” IEEE Transactions on Information Theory, vol. 46, no. 2, pp. 388–404, Mar. 2000.

[28] Heinzelman, R.; Chandrakasan, A. and Balakrishnan, H., "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," Proceedings of the 33rd International Conference on System Sciences (HICSS '00), January 2000.

[29] Helvig, C.; Robins, G. and Zelikovsky, A. “An Improved Approximation Scheme for the Group Steiner Problem,” Networks, pp. 8–20, 2001.

[30] http://research.cens.ucla.edu/

[31] http://www.algorithmic-solutions.com/index.htm last viewed on 10 september 2007.

[32] Hu,X., Liu,Y., Lee,M., Saadawi, T., “Decoupled Multipath Structure for Throughput Enhancement in Wireless Mesh Networks,” IEEE WCNC , pp. 325-330, April 2006.

[33] Intanagonwiwat, C., Govindan, R. and Estrin, D., “Directed diffusion: A scalable and robust communication paradigm for sensor networks,” Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00), 2000.

[34] Intel, “Intel Stargate,” URL - http://platformx.sourceforge.net, 2002.

[35] Iyer, R. and Kleinrock, L., “QoS control for sensor networks,” 2003 IEEE International Conference on Communications, v.1, pp.517-521, 2003.

[36] Jain, K., Padhye, J., Padmanabhan, V., Qiu, L., “Impact of interference on multi-hop wireless network performance,” Proc. Mobicom, ACM Press, New York, pp. 66-80, 2003.

[37] Jia, L., Rajaraman, R., and Suel, T. 2002., “An efficient distributed algorithm for constructing small dominating sets,” Distributed Computing 15, 4 (Dec. 2002), 193-205, 2002.

[38] Jiménez, V. M. and Marzal, “Computing the K Shortest Paths: A New Algorithm and an Experimental Comparison,” Proceedings of the 3rd international Workshop on Algorithm Engineering, Lecture Notes In Computer Science, vol. 1668. Springer-Verlag, pp. 15-29, 1999.

[39] Jones, E.P.C.; Karsten, M.; Ward, P.A.S., “Multipath load balancing in multi-hop wireless networks,” IEEE Intl. Conf. on Wireless And Mobile Computing, Networking And Communications (WiMob'2005), pp.158-166, 2005.

[40] Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. S., and Rubenstein, D., “Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet”, Proceedings of the 10th international Conference on Architectural Support For Programming Languages and Operating Systems, pp. 96-107, 2002.

[41] Kar, K., Kodialam, M., Lakshman, T.V., and Tassiulas, L., “Routing for Network Capacity Maximization in Energy-Constrained Ad-Hoc Networks,” International Conference on Computer Communications (INFOCOM), pp. 673-681, 2003.

[42] Karl, H., “Quality of Service in wireless sensor networks: Mechanisms for a new concept?”, located at http://typo3.cs.uni-paderborn.de/fileadmin/Informatik/AG-Karl/Publications/Talks/karl.pdf last viewed 22 February 2007.

[43] Kay, J. and Frolik, J., “Quality of service analysis and control for wireless sensor networks,” IEEE International Conference on Mobile Ad-hoc and Sensor Systems 2004, pp.359-368, 2004.

[44] Kazmierczak, A. and Radhakrishnan, S., “Efficient distributed algorithms for disjoint s-t paths and testing k-connectivity in networks,” Proc.of Sixth ISCA Intl. Conf. on Parallel and Dist. Comp. Systems, pp. 236-241, Oct. 1993.

[45] Khuller, S. and Scheiber, B., “Efficient algorithms for testing k-connectivity and finding disjoint s-t paths in graphs,” SIAM Journal on Computing, 20(2), pp.352-375, Apr. 1991.

[46] Krishnamurthy, L., Adler, R., Buonadonna, P., Chhabra, J., Flanigan, M., Kushalnagar, N., Nachman, L., and Yarvis, M. 2005, “Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea,” In Proceedings of the 3rd international Conference on Embedded Networked Sensor Systems (San Diego, California, USA, November 02 - 04, 2005). SenSys '05. ACM, New York, NY, 64-75, 2005.

[47] Li, J., Blake, C., De Couto, D. S., Lee, H. I., and Morris, R., “Capacity of Ad Hoc wireless networks,” Proc. 7th Annual Intl. Conf. on Mobile Computing and Networking, pp. 61-69, 2001.

[48] Li, Q., Aslam, J., and Rus, D., “Online power-aware routing in wireless Ad-hoc networks,” Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom), pp. 97-107, 2001.

[49] Liang, B., Frolik, J. and Wang, X.S., “A predictive QoS control strategy for wireless sensor networks,” 2005 IEEE International Conference on Mobile Adhoc and Sensor Systems, pp.8, 2005.

[50] Liaw, Y.S.; Dadej, A.; Jayasuriya, A., “Characterisation and throughput performance of multipaths in wireless multihop network,” Proc. 12th IEEE International Conference on Networks, vol.2, no., pp. 780-784, vol.2, 2004.

[52] Linial, N. 1992, “Locality in distributed graph algorithms,” SIAM J. Comput. 21, 1 (Feb. 1992), pp. 193-201, 1992.

[52] Lu, G., Krishnamachari, B., and Raghavendra, C.S., “Performance Evaluation of the IEEE 802.15.4 MAC for Low-Rate Low-Power Wireless Networks,” IEEE International Performance, Computing and Communications Conference (IPCCC), pp. 701-706, 2004.

[53] Madan, R. and Lall, S., “Distributed algorithms for maximum lifetime routing in wireless sensor networks,” Global Telecommunications Conference, 2004. GLOBECOM '04, IEEE, pp. 748-753 Vol.2, 29 Nov.-3 Dec. 2004.

[54] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. 2002, “Wireless sensor networks for habitat monitoring,” In Proceedings of the 1st ACM international Workshop on Wireless Sensor Networks and Applications (Atlanta, Georgia, USA, September 28 - 28, 2002). WSNA '02. ACM, New York, NY, pp. 88-97, 2002.

[55] Mangharam, R.; Rowe, A; Rajkumar, R.; Suzuki, R. “Voice over Sensor Networks ,” in 27th IEEE International Real-Time Systems Symposium (RTSS’06), 2006, pp. 291–302, 2006.

[56] Mangharam, R. and Rajkumar,R., “MAX: A Maximal Transmission Concurrency MAC for Wireless Networks with Regular Structure,” IEEE Third International Conference on Broadband Communications, Networks and Systems (IEEE BROADNETS), pp. 1-10, 2006.

[57] McIntire, D.; Hing, K. H.; Yip, B.; Singh, A.; Wu, W. and Kaiser, W. “The low power energy aware processing (leap) system,,” in IPSN SPOTS, pp. 449–457, April 2006.

[58] Lu, M.; Wu, J.; Cardei, M.,and Li, M. “Energy-efficient connected coverage of discrete targets in wireless sensor networks,” Data and Applications Security XIX. 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Proc. of Lecture Notes in Computer Science Vol.3654, pp.43-52, 2005.

[59] Mohanoor, A.B.; Radhakrishnan, S.; Sarangan, V., "On energy aware routing in wireless networks," Broadband Communications, Networks and Systems, 2007. BROADNETS 2007. Fourth International Conference on , vol., no., pp.690-697, 10-14 Sept. 2007.

[60] Mohanoor, A.B.; Radhakrishnan, S.; Sarangan, V., “A distributed algorithm for interference aware routing in wireless networks,” accepted for publication in Sixth IEEE Consumer Communications and Networking Conference (CCNC), 2009.

[61] Mohanoor, A.B.; Radhakrishnan, S.; Sarangan, V., “Interference aware multiple path routing in wireless networks,” accepted for publication in Fifth Intl. Conf. on Mobile Ad hoc and Sensor Systems (MASS), Oct 2008.

[62] Mohanoor, A.B.; Radhakrishnan, S.; Sarangan, V., “Online energy aware routing in wireless networks,” accepted for publication in Elsevier Adhoc Networks, 2008.

[63] Mohanoor, A.B..; Radhakrishnan, S.; Sarangan, V., “Radio activation in multi-radio wireless sensor networks,” submitted to IEEE Conference on Communications (Infocom) 2009.

[64] Naor, M. and Stockmeyer, L., “What can be computed locally?,” In Proceedings of the Twenty-Fifth Annual ACM Symposium on theory of Computing (San Diego, California, United States, May 16 - 18, 1993). STOC '93. ACM, New York, NY, pp.184-193, 1993.

[65] Park, J., and Sahni, S., “An online heuristic for maximum lifetime routing in wireless sensor networks,”, IEEE Transactions on Computers, v.55, no.8, pp. 1048-1056, 2006.

[66] Perillo, M., Mark, A., and Wendi, B., “Sensor management policies to provide application QoS,” Ad Hoc Networks, v.1, n.2-3, pp.235-246, September 2003.

[67] Pering, T., Raghunathan, V. and Want, R. “Exploiting radio hierarchies for power-efficient wireless discovery and connection setup,” in 18th International Conference on VLSI Design, pp. 774–779, 2005.

[68] Peterson, L.L. and Davie, B. Computer Networks: A systems Approach, Morgan Kaufmann, 2003.

[69] Philip, B., Chhabra, J. Krishnamurthy, L. and Kushalnagar, N., "Heavy Industry Applications of Sensornets," Intl. Conf. on Distributed Computing in Sensor Systems (DCOSS) 2005.

[70] Ramarao, K. V. and Venkatesan, S. 1992, "On finding and updating shortest paths distributively," J. Algorithms 13, 2 ,pp.235-257, 1992.

[71] Robinson, J. ; Papagiannaki, K.; Diot, C.; Guo, X. and Krishnamurthy, L. "Experimenting with a Multi-Radio Mesh Networking Testbed," in Proceedings of the First Workshop on Wireless Network Measurements, 2005.

[72] Saha,D.; Toy, S.; Bandyopadhyay, S.; Ueda, T.; Tanaka, S., "An adaptive framework for multipath routing via maximally zone-disjoint shortest paths in ad hoc wireless networks with directional antenna," Global Telecommunications Conference, 2003. GLOBECOM '03 IEEE, vol.1, no., pp. 226-230 Vol.1, 1-5 Dec. 2003.

[73] Santi, P. "Topology control in wireless ad hoc and sensor networks,"ACM Computing Surveys, vol. 37, no. 2, pp. 164–194, 2005.

[74] Schurgers, C.; Tsiatsis, V.; Ganeriwal, S. and Srivastava, M. B. "Topology management for sensor networks: Exploiting latency and density," in MobiHoc, 2002, pp. 135–145, 2002.

[75] Segev, A. "The node-weighted steiner tree problem," Networks, pp. 1– 17, 1987.

[76] Sharma, M. B. and Iyengar, S. S. 1989, "An efficient distributed depth-first-search algorithm," Inf. Process. Lett. 32, 4 (Sep. 1989), 183-186, 1989.

[77] Shih, E.; Bahl, P.; and Sinclair, M. J. "Wake on wireless: An event driven energy saving strategy for battery operated devices," in Mobicom, pp. 160–171, 2002.

[78] Sobrinho, J.L., "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," Networking, IEEE/ACM Transactions on, vol.10, no.4, pp. 541-550, Aug 2002.

[79] Sorber, J.; Banerjee, N.; Corner, M. D. and Rollins, S. "Turducken: Hierarchical power management for mobile devices," in MobiSys, pp. 261–274, 2005.

[80] Stathopoulos, T.; Lukac, M.; McIntire, D.; Heidemann, J.; Estrin, D. and Kaiser, W. J. "End-to-end Routing for Dual-Radio Sensor Networks," in Proceedings of INFOCOM, pp. 2252–2260, 2007.

[81] Toh, C.-K., Cobb, H. and Scott, D.A., "Performance evaluation of battery-life-aware routing schemes for wireless ad hoc networks," IEEE International Conference on Communications (ICC), v.9, pp. 2824-2829, 2001.

[82] Wan, P-J.; Alzoubi, K. M. and Frieder, O. “Distributed construction of connected dominating set in wireless ad hoc networks,” Mobile Networks and Applications, pp. 141–149, 2004.

[83] Wang, X.H.; Iqbal, M.; Zhou, X. “Design and implementation of a dual radio wireless mesh network testbed for healthcare,” in Internation Conference on Technology and Applications in Biomedicine, pp.300-304, 2008.

[84] Wang, Z. and Crowcroft, J., “Quality-of-service routing for supporting multimedia applications,” IEEE Journal on Selected Areas in Communications, v.14, no.7, pp. 1228-1234, 1996.

[85] Wattenhofer, R.; Li, L.; Bahl, P. and Wang, Y. M. “Distributed topology control for power efficient operation in multihop wireless ad hoc networks,” in INFOCOM, pp. 1388–1397, 2001.

[86] Werner-Allen, G.; Johnson, J.; Ruiz, M.; Lees, J.; Welsh, M., "Monitoring volcanic eruptions with a wireless sensor network," Wireless Sensor Networks, 2005.

[87] Wu, K. and Harms, J. 2001, “Performance Study of a Multipath Routing Method for Wireless Mobile Ad Hoc Networks,” In Proceedings of the Ninth international Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp.99-107, 2001.

[88] Wu, K., Gao, Y.; Li, F.; Xiao, Y., “Lightweight deployment-aware scheduling for wireless sensor networks,” Mobile Networks and Applications, v.10, n.6, pp.837-852, 2005.

[89] Zhang, P., Sadler, C. M., Lyon, S. A., and Martonosi, M. 2004. Hardware design experiences in ZebraNet. In Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems (Baltimore, MD, USA, November 03 - 05, 2004). SenSys '04. ACM, New York, NY, 227-238, 2004.

[90] Zhu, J. and Papavasilliou, S., “A resource adaptive information gathering approach in sensor networks,” IEEE/Sarnoff Symposium on Advances in Wired and Wireless Communications, pp.115-118, 2004.

[91] Zhu, J.; Waltho, A.; Yang, X. and Guo, X. “Multi-Radio Coexistence: Challenges and Opportunities,” in ICCCN, pp. 358–364, 2007.

[92] Zussman, G. and Segall, A., “Energy efficient routing in ad hoc disaster recovery networks,” Ad Hoc NetworksVolume 1, Issue 4, November 2003, pp. 405-421, 2003.

Appendix A

IRIS Mote

Processor:

Program flash memory 128 KB

Measurement flash (for sensing) 512 KB

RAM 8K bytes

Processor current draw 8mA active mode, 8μA sleep mode

Radio:

Frequency band 2405 MHz to 2480 MHz

Transmit data rate 250 kbps

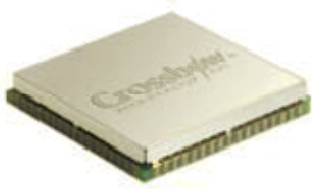
Radio current draw 16mA receive mode, 17mA transmit mode (typical)

Batteries: 2 AA batteries

Source:

http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf

MICAz OEM module



Processor:

Program flash memory 128KB

Measurement flash 512 KB

RAM 4KB

Current draw 8mA active mode, <15µA sleep mode

Radio:

Frequency band 2400 MHz to 2483.5 MHz

Transmit data rate 250 kbps

Current draw 19.7mA receive mode, 17.4mA transmit mode

Batteries: 2 AA batteries

Source:

http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_OEM_Edition_Datasheet.pdf

MICA 2

Processor:

Program memory 128 KB

Measurement flash 512 KB

Configuration EEPROM 4KB

Current draw 8mA active mode, <15µA sleep mode

Radio:

Frequency band 868-916 MHz

Transmit data rate 38.4 Kbaud

Current draw 10mA receive, 27mA transmit

Batteries: 2 AA batteries

Source:

http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf

IMote2

Processor:

SRAM memory 256kB

SDRAM memory 32MB

FLASH memory 32MB

Current draw 66mA active mode, 390μA sleep mode

Radio:

Frequency band 2400-2483.5 MHz

Data rate 250kbps

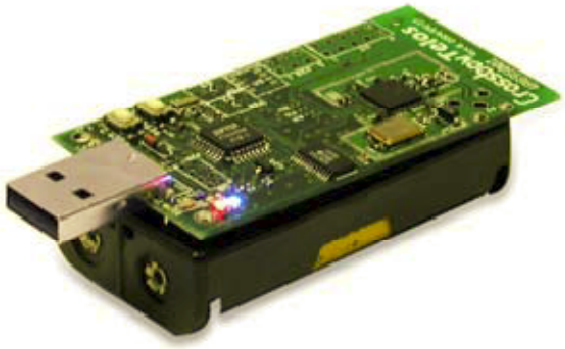
Batteries 3 AAA batteries

Source:

http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf

Note: This sensor can run the .NET micro framework (like a Java Virtual Machine for mobile devices)

TelosB



Processor:

Program flash memory 48KB

Measurement flash 1024 KB

RAM 10KB

Radio:

Frequency band 2400-2483.5 MHz

Transmit data rate 250kbps

Batteries 2 AA batteries

Source:

http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf

Appendix B

Here we list some real world sensor network deployments.

Volcano monitoring

From http://fiji.eecs.harvard.edu/Volcano

"This interdisciplinary project is investigating the use of wireless sensor networks for monitoring eruptions of active and hazardous volcanoes. Wireless sensor networks have the potential to greatly enhance our understanding of volcanic activity. The low cost, size, and power requirements of wireless sensor networks have a tremendous advantage over existing instrumentation used in volcanic field studies. This technology will permit sensor arrays with greater spatial resolution and larger apertures than existing wired monitoring stations.

We have deployed three wireless sensor networks on active volcanoes. Our initial deployment at Tungurahua volcano, Ecuador, in July 2004 served as a proof-of-concept and consisted of a small array of wireless nodes capturing continuous infrasound data. Our second deployment at Reventador volcano, Ecuador, in July/August 2005 consisted of 16 nodes deployed over a 3 km aperture on the upper flanks of the volcano, and measured both seismic and infrasonic signals with high resolution (24 bits per channel at 100 Hz). Our third deployment at Tungurahua in August 2007 tested the [[Lance]] utility-driven data collection system that we developed to enhance data fidelity."

Vineyard monitoring

From http://camalie.com/WirelessSensing/WirelessSensors.htm

“Camalie vineyards currently has one of the most advanced soil moisture monitoring systems in operation in U.S. agriculture today. It uses the wireless sensor networking technology developed by UC Berkeley in collaboration with Intel Corp and commercialized by Crossbow Inc. It is used for optimization of irrigation; reducing water consumption and associated pumping energy costs as well as increasing grape quality without sacrificing yield. It also serves as a monitor of the irrigation system, failure of which can cause substantial long term impact on the large capital investment in the vines. This monitoring system, Camalie Net, as such provides a form of risk management.

Mark Holler, owner of Camalie Vineyards, developed interface circuitry and adapted Crossbow hardware and associated TinyOS firmware for the Watermark soil moisture tensiometer, soil temperature probe and water pressure sensors used in the first version of this system which was deployed in the summer of 2005. Mark is also the grower who looks at the data and makes the irrigation decisions. Ramon Pulido vineyard manager for Camalie Vineyards brings 32 years of growing experience on Mt. Veeder to the vineyard. He manages all of the cultural practices, spraying for powdery mildew, and erosion control using cover crop in the vineyard.

Soil moisture sensing began at Camalie Vineyards in 2003 using a Davis weather station with three Watermark Sensors. This data was found to be quite useful but, it was clear data from other irrigation blocks was needed to irrigate optimally in them as well. The lack of a scalable solution to gather data from more locations was the driving force behind the development of Camalie Net.

Camalie Net was used during the 2005, 2006 and 2007 growing seasons to guide irrigation decisions in the 4.4 acres of Camalie Vineyards. Yield per vine in 2005 was double that of the 2004 yields for same age vines yet water consumption was constant. Water consumption normally goes up with canopy size which more than doubled for these 2.5 year old vines in 2005. Grape quality was excellent. Some of this success was due to generally better than average weather in 2005 but, we believe our visibility of the soil moisture played a significant role. Extra drippers were added to some areas of the vineyard based on the soil moisture data. Also irrigation intervals were shortened based on sensor data to reduce the amount of water that penetrated below the root zones where it would be wasted.

In 2006, the third year for our vines, the yields again doubled from 4 tons to 8 tons.

In the 4th year, 2007 the yield again doubled to 16 tons of fruit that was sold and another 1.5 tons that we made home wine from. The yield was 3.97 tons per acre which is very rare on Mt. Veeder especially with water limited due to less than half the normal rainfall in the winter of 06/07. Water had to be purchased but thanks to our precision irrigation we minimized water purchasing and still had great yields. Fruit quality was again excellent although early rains near harvest bloated some of the first fruit harvested, reducing Brix levels to the 25-26 range. Some of the last fruit harvested after a week or two of dry weather, however, had a late in the day Brix average of 29.7!

At the end of the 2007 growing season the network was upgraded from my 433MHz prototypes which use Crossbow mica2dot radios to eKo Pro series alpha units

opperating in the 2.4GHz band. The network was scaled up at this time from 10 sensing sites in one vineyard to 17 sites covering Camalie Vineyards and the vineyards of two neighbors and the Mt. Veeder Irrigation Co-op was formed. On March 3, 2008 it was scaled up again to 25 sites. In May six of the nodes at Camalie Vineyards were fitted with third soil moisture sensors at 36" depth. In April 2008 Camalie Networks LLC was formed to sell, customize and service this technology."

Zebra monitoring

From http://www.princeton.edu/~mrm/zebranet.html

"Funded by a research grant from the National Science Foundation through their Information Technology Research (ITR) initiative, ZebraNet is a project to explore wireless protocols and position-aware computation from a power-efficient perspective.

The Princeton ZebraNet Project is an inter-disciplinary effort with thrusts in both Biology and Computer Systems. On the computer systems side, ZebraNet is studying power-aware, position-aware computing/communication systems. Namely, the goals are to develop, evaluate, implement, and test systems that integrate computing, wireless communication, and non-volatile storage along with global positioning systems (GPS) and other sensors. On the biology side, the goal is to use systems to perform novel studies of animal migrations and inter-species interactions.

As a computer systems research problem, ZebraNet is compelling because the needs of the biological researchers are stringent enough to require real breakthroughs in wireless protocols and in low-power computer systems design and computer systems power

management. These breakthroughs can be leveraged into other (non-wildlife-oriented) fields of research; essentially ZebraNet is a power-aware wireless ad hoc sensor network, but with more serious bandwidth and computational needs than most prior sensor networks research problems. As a biology research problem, ZebraNet allows researchers to pose and to answer important long-standing questions about long-range migration, inter-species interactions, and nocturnal behavior."

Storm Petrel habitat monitoring

From http://www.coa.edu/html/motes.htm

"This project represents a collaboration between College of the Atlantic and the Intel Research Laboratory at Berkeley. Through a combination of funding from the Henry Luce Foundation and in-kind donation of equipment and expertise from the Intel Research laboratory students under the direction of John Anderson and Dr. Alan Mainwaring engaged in the active demonstration of wireless sensor network technology as applied to micro-habitat monitoring on Great Duck Island.

Intel-developed "motes" consisting of micro-processors containing temperature, humidity, barometric pressure, and infra-red sensors were deployed within Leach's Storm Petrel nesting habitat, and linked to a computer base station in the Eno Research Station. This in turn fed into a satellite link that allowed researchers to download real-time environmental data over the internet.

The ultimate goal was to enable researchers anywhere in the world to engage in non-intrusive monitoring of sensitive wildlife and habitats."