ROTEAMENTO MULTICAMINHO PARA REDES DE SENSORES SEM FIO COM DOIS RÁDIOS

NILDO DOS SANTOS RIBEIRO JÚNIOR

ROTEAMENTO MULTICAMINHO PARA REDES DE SENSORES SEM FIO COM DOIS RÁDIOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Marcos Augusto Menezes Vieira
Coorientador: Luiz Filipe Menezes Vieira

Belo Horizonte

Outubro de 2017

NILDO DOS SANTOS RIBEIRO JÚNIOR

MULTIPATH ROUTING FOR DUAL-RADIO WIRELESS SENSOR NETWORKS

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Marcos Augusto Menezes Vieira
Co-Advisor: Luiz Filipe Menezes Vieira

Belo Horizonte
October 2017

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Ribeiro Júnior, Nildo dos Santos

R484m Multipath routing for dual-radio wireless sensor networks / Nildo dos Santos Ribeiro Júnior. — Belo Horizonte, 2017
xxiv, 57 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.
Orientador: Marcos Augusto Menezes Vieira
Coorientador: Luiz Filipe Menezes Vieira

1. Computação – Teses. 2. Redes de sensores sem fio. 3. Roteamento (Administração de redes de computadores). I. Orientador. II. Coorientador. III. Título.

CDU 519.6*22(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Multipath Routing for Dual-Radio Wireless Sensor Networks

NILDO DOS SANTOS RIBEIRO JUNIOR

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

Marcos Augusto Menezes Vieira
Prof. Marcos Augusto Menezes Vieira - Orientador
Departamento de Ciência da Computação - UFMG

Luiz Filipe Menezes Vieira
Prof. Luiz Filipe Menezes Vieira - Coorientador
Departamento de Ciência da Computação - UFMG

[illegible]
Profa. Olga Nikolaevna Goussevskaia
Departamento de Ciência da Computação - UFMG

[illegible]
Prof. Omprakash Gnawali
Department of Computer Science - University of Houston

Belo Horizonte, 6 de Outubro de 2017.

To everybody.

Acknowledgments

The 25 years I have lived has led me to write this note of gratitude as a final touch of my dissertation. I know none of my accomplishments are a result of my pure merit. I have been lucky in many aspects. I am very grateful to have many special people in my life. I consider them my greatest privilege. I would like to thank you all.

First of all, my family. To my parents, Nildo and Olinda, who have been there for me every single day. To my sister and brother, Patrícia and Marcos, and to my aunt Lia. Thank you all for the support and unconditional love that was so important to me not only in the recent years, but throughout my whole life.

I would like to thank my academic advisor, Marcos Augusto Vieira, and co-advisor Luiz Filipe Vieira, for their guidance. Thank you for the great opportunities you provided me and for all the valuable lessons I have learned over these five years we have been working together.

I would also like to thank my dear friends from UFMG, who have also being so supportive all these years. To my good friends Gabriel Poesia, Victor Rodrigues, Luiz Santos, Jocasta Cunha, Édson Silva, Henrique Alves, Rafael Moraes, Diego Moreira, Raquel Aoki, and every friend working at LECOM with me, thank you for making my time at the university the best experience it could be.

Finally, I would like to thank every teacher in my academic life, always studying at public schools. You were always great professionals working with very few resources and most of the time not being recognized as you deserve. Every teacher who has taught me something at some point were very important to me, and for that I am very grateful.

"If I am not in the world simply to adapt to it, but rather transform it, and if it is not possible to change the world without a certain dream or vision for it, I must make use of every possibility there is not only to speak about my utopia, but also to engage in practices consistent with it."

(Paulo Freire)

Resumo

O projeto de uma Rede de Sensores Sem Fio depente muito da aplicação. Para aplicações tradicionais, como monitoração ambiental, *smart buildings* ou na agricultura, foram sempre priorizadas a redução do custo das plataformas, o gasto de energia e o uso de memória, em troca de ter uma vazão mais baixa. Aplicações modernas, como sistemas de segurança ou monitoramento de tráfego, geralmente exigem o uso de câmeras e transmissão de dados de vídeos pela rede. Para aplicações como essas, plataformas de dois rádios foram desenvolvidas, onde o seu projeto prioriza alcançar uma vazão maior e conservar a eficiência energética da rede. Nesse trabalho é proposto um algoritmo de roteamento multicaminho para encontrar dois caminhos disjuntos com a mesma paridade entre um par de nós da rede. O novo algoritmo de roteamento, combinado com um esquema de encaminhamento de pacotes que alterna os rádios ao longo do caminho, permite que todos os nós usem os dois rádios que possuem em paralelo a todo o tempo, dobrando a vazão em comparação com um esquema de um único caminho. Nós avaliamos nosso algoritmo e esquema de encaminhamento em um testbed físico contendo 100 nós da plataforma Opal, que possui dois rádios. Nosso protocolo conseguiu dobrar a vazão em comparação com o FastForward, o protocolo estado-da-arte para plataformas com dois rádios, e conseguiu atingir até 96% do limite teórico.

Palavras-chave: Roteamento Multicaminho, Redes de Sensores Sem Fio, Dois Rádios.

Abstract

The design of a Wireless Sensor Network depends significantly on the application. For the traditional applications, such as environmental monitoring, smart buildings or agriculture, the design prioritized reducing the cost, the energy expenditure and memory usage at the expense of not having a higher throughput. Modern applications, such as surveillance or traffic monitoring, usually require using cameras and transmitting video data through the network. For such applications, dual-radio platforms were developed where their design prioritizes achieving higher throughput and conserve the energy efficiency of the network. In this work, we propose a multipath routing algorithm to find two disjoint paths with the same parity between a pair of nodes in the network. The new routing algorithm, combined with a forwarding scheme that alternates the radios throughout the paths, allows all nodes in the paths to use both radios in parallel all the time, doubling the throughput in comparison with a single path routing scheme. We evaluated our algorithm and forwarding scheme in a real world testbed with 100 nodes of the Opal dual-radio platform. Our scheme were able to double the throughput when compared with FastForward, the state-of-the-art protocol for dual-radio platforms, and achieve up to 96% of the theoretical limit.

Keywords: Multipath Routing, Wireless Sensor Networks, Dual-radio.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

A Wireless Sensor Network (WSN) consists of several sensor nodes, which act as both data generators and network relays. Each node have one or more sensors, a micro-processor and a radio transceiver. According to Akyildiz and Vuran [2010], wireless communication provides ease of deployment, and the distributed sensing capabilities makes WSNs an important component of our daily lives. They have many applications, such as environmental monitoring, agriculture, health care, and smart buildings.

The design of a WSN depends significantly on the application. Yick et al. [2008] state that the environment, the application's design principle, the hardware and systems constraints might be very different in each scenario. For instance, the environment is important to determine the size of the network, the deployment scheme and even the network topology. But when we are talking about the sensor nodes' hardware, there are two key issues taken into account in their design: cost and power consumption.

Since sensor networks consist of a large number of sensor nodes, the cost of a single node is very important to justify the overall cost of the network. This constraint makes most sensor networks platforms have very limited processing power and memory, so the cost of each node can be as low as possible. As a wireless sensor node is usually powered with batteries, power consumption must be minimized. Because of that we are always trying to minimize the use of the radio transceiver, since the wireless communication consume a large amount of power when compared to the rest of the components of the hardware architecture.

Minimizing cost and power consumption is achieved, but at the expense of the network's performance. This is a very important tradeoff in the design of WSNs. According to Jurdak et al. [2011], the design of traditional sensor network platforms has favored low power operation at the cost of communication throughput or range. This made sense in the context of the beginning of the development of WSNs, where most applications would be to collect small pieces of data, such as temperature or lighting values.

Applications are now also being deployed to gather acoustic and visual data, which have a high demand for communication throughput. So, though energy remains a key consideration in the design of WSNs, throughput is gaining more importance in these new kinds of applications. As new techniques for energy harvesting have been developed, the design of the platforms could switch to prioritize other factors instead of the total energy consumption. An example of how these priorities can be changed is to prioritize the energy efficiency, which is the amount of energy used to transmit a certain amount of data, instead of the total energy consumption. This way we can have applications with higher throughput, which would spend more energy overall, but conserving the energy efficiency of the system.

This change of paradigm motivated researchers to develop new platforms for WSNs aimed to deliver higher throughput while still being energy efficient. A technique that allows us to achieve this goal is radio diversity, which means having more than one radio in each WSN mote. To be able to improve throughput, these radios should be able to operate at the same time. It is also desirable that these radios operate in different frequency band, so they will not interfere with each other while communicating, reducing packet loss due to interference and allowing simultaneous transmissions without interference in the same sensor node. According to Kusy et al. [2011], radio diversity can significantly improve end-to-end delivery rates, network stability and transmission costs at only a small increase in energy cost over a single radio.

Due to the development of multi-radio platforms aiming to provide radio diversity, the research community started working on adapting the network protocols to explore all the advantages brought by radio diversity. The ability to simultaneously transmit and receive packets through different radios was exploited by Ekbatanifard et al. [2013] in the design of FastForward, a high-throughput transport protocol for dual-radio WSNs. In FastForward, each forwarder receives packets through one radio and sends them though the other one, as it is shown in Figure 1.1a. This scheme allows all the intermediate nodes in the path to be receiving and transmitting packets at the same time, achieving an end-to-end throughput equal to the channel capacity. This result is twice the maximum throughput it is possible to achieve with single-radio platforms, because the radio would have to spend half of its time receiving packets and the other half forwarding them.

The problem with FastForward's scheme is that the source and the destination nodes, despite having two radios available on them, they only use one radio each, since it uses only one path. The main idea of this dissertation is to fully utilize all

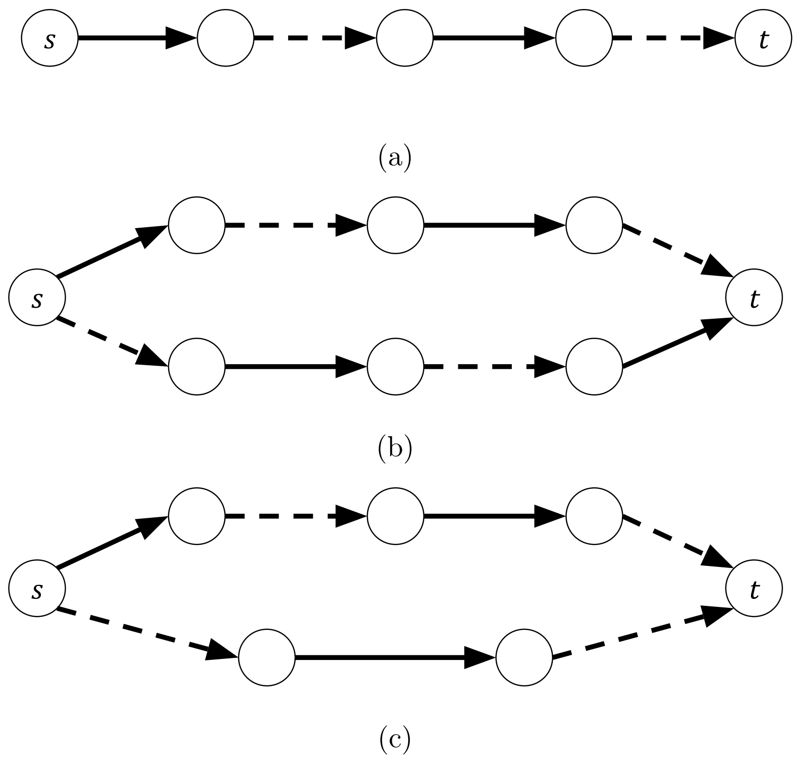


Figure 1.1: Examples of a forwarding scheme using a single path, two paths with the same parity and two paths with different parities.

the available radios, even those in the source and destination nodes, to achieve the maximum theoretical throughput using dual-radio platforms. In order to accomplish this goal, we calculate and use two disjoint paths, where each intermediate node will receive packets through one radio and retransmit them through the other one, just as the previous scheme, but the source node will be using both radios for transmissions and the destination node will be using both for receiving packets, as shown in Figure 1.1b.

Using two disjoint paths to transmit packets potentially doubles the throughput compared with the single path scheme, theoretically achieving twice the channel capacity. However, it adds a new constraint to the routing problem. The two paths must have lengths of the same parity. If this is not the case, there will be a bottleneck in the network, where the destination node would be receiving packets coming from two different paths through the same radio, and the result would be a maximum throughput equal to the one we would achieve using a single path. This situation is illustrated in Figure 1.1c.

The parity restriction defines a new graph problem of finding two vertex disjoint simple paths between a source and a destination with lengths of the same parity. The problem arises from the practical application in dual-radio WSNs described here, and does not have a solution in the literature. In this dissertation, we discuss the nature of this problem for directed and undirected graphs, present a practical solution and evaluate the gains in throughput in the network in a physical testbed with dual-radio platforms.

1.1 Motivation

The recent availability of inexpensive hardware that are able to ubiquitously capture multimedia content from the environment, such as CMOS cameras and microphones, has led to the development of Wireless Sensor Networks that interconnect devices that allow retrieving video and audio streams, still images, and scalar sensor data [Gürses and Akan, 2005; Misra et al., 2008]. A single sensor device can be equipped with audio and visual information collection modules because of the rapid improvements and miniaturization in hardware. An example is the Cyclops image capturing and inference module presented by Rahimi et al. [2005], which is designed for extremely light-weight imaging and can be interfaced with a host mote.

Transmitting multimedia content through WSNs will not only enhance existing sensor network applications such as tracking, home automation, and environmental monitoring, but they will also enable several new applications. For example, Teixeira et al. [2006] shows that there is a particular interest in the possible use of WSNs in monitoring the elderly aging at home. Processing video with sensor networks eliminate the need for elderly to remember to wear cumbersome instrumentation and sensors. There is a whole potential for integrating multimedia networks to provide ubiquitous health care services. Patients can carry medical sensors to monitor parameters such as body temperature, blood pressure, breathing activity, among others. In addition of that, remote medical centers could be able to perform advanced remote monitoring of their patients via video and audio sensors, location sensors, motion or activity sensors, which can also be embedded in wrist devices, as in Hu and Kumar [2003].

There is also a potential application in surveillance systems, since wireless video sensor networks will be composed of interconnected, battery-powered miniature video cameras, each packaged with a low-power wireless transceiver that is capable of processing, sending, and receiving data. Video and audio sensors can be used to enhance and complement existing surveillance systems against crime and terrorist attacks. Large-scale networks of video sensors can extend the ability of law enforcement agencies to monitor areas, public events, private properties and borders. Some research have been recently made for video surveillance sensor networks as in Durmus et al. [2012].

Another potential application is environmental monitoring. Several projects on habitat monitoring that use acoustic and video feeds are being envisaged, in which information has to be conveyed in a time-critical fashion. For example, arrays of video sensors are already used by oceanographers to determine the evolution of sandbars via image processing techniques [Holman et al., 2003]. A recent example of video sensor network being developed is presented by Zhang et al. [2014], where they study coastal erosion. Recording the physical erosion events during extreme high waves is significant to evaluate the dynamics of bluff erosion and to document these short-term processes. Still and motion imagery are important media to observe rare and extreme events in ecology, geology, and environmental condition. This study requires recording devices for these modalities capable of long-term, low-cost, low-power operation with low maintenance, and with the ability to support a large dynamic range in both time and space. So, they development of a wireless video camera network for studying coastal erosion and implemented a video camera network on Thompson Island, Boston Harbor.

With multimedia content being transmitted by WSNs, traffic avoidance, enforcement and control systems will be able to monitor car traffic in big cities or highways and deploy services that offer traffic routing advice to avoid congestion. Smart parking advice systems based on WSNs will allow monitoring available parking spaces and provide drivers with automated parking advice, thus improving mobility in urban areas. Moreover, multimedia sensors may monitor the flow of vehicular traffic on highways and retrieve aggregate information such as average speed and number of cars. Sensors could also detect violations and transmit video streams to law enforcement agencies to identify the violator, or buffer images and streams in case of accidents for subsequent accident scene analysis.

Most of the above applications require the sensor network paradigm to be rethought in view of the need for mechanisms to deliver multimedia content with a certain level of quality of service. As we know, the need to minimize the energy consumption has driven most of the research in sensor networks so far, mechanisms to efficiently deliver application level quality of service, and improvements in the network throughput have not been primary concerns in mainstream research on classical sensor networks.

1.2 Objective

The objective of this dissertation is to provide a method to increase the throughput achievable in end-to-end data transmissions in WSNs. The proposed approach is using multipath routing combined with dual-radio platforms that can operate both radios at the same time to achieve the highest possible throughput, which has a theoretical limit of twice the channel capacity.

1.3 Contributions

The main contributions of this work are:

- The introduction of a new graph problem of finding two vertex disjoint simple paths with lengths of the same parity. We show that the decision version of the problem in undirected graphs is polynomial, but we proof the same problem in directed graphs to be NP-complete.
- We present an optimal centralized routing solution for the minimization version of the problem in directed graphs, modeling it as integer linear programming.
- We present the design and implementation of a distributed heuristic solution for the minimization problem in directed graphs.
- Experiments evaluating different models for real world dual-radio wireless sensor networks.
- Experiments with the proposed multipath routing scheme in a real world testbed with 100 dual-radio nodes, showing that it is possible to achieve a throughput of up to 96% of the theoretical limit.

1.4 Organization

The organization of this dissertation is as it follows: In Chapter 2 we discuss the related work. In Chapter 3 we present the theoretical basis and theoretical contributions of this work. We formalize the problem and detail the complexity of its different versions. In Chapter 4 we present the solutions we developed for the problem. In Chapter 5 we summarize the experiments we did to evaluate the performance of our protocol and compare with the state-of-the-art protocol. Finally, in Chapter 6 we present the conclusions and list many possible future work.

Chapter 2

Bibliography Review

In this chapter we present previous work related to the two main ideas we use in this dissertation: radio diversity and multipath routing. Though we apply these ideas with the objective of increasing the throughput in sensor networks, many of the works we present are not directly related to WSNs, but understanding them provided insights to this work. We present an overview of multi-radio platforms and protocols, multipath algorithms and other techniques used to improve throughput in WSNs.

2.1 Multi-radio Platforms and Protocols

The first sensor node platform that employ a multi-radio feature we have found is the BTnode platform, developed by Beutel [2006]. This platform incorporates a low-power radio operating in ISM band 433-915 MHz and a Bluetooth radio. The author claims that it is the first dual-radio platform for sensor networks, and that it provides opportunities to investigate the trade-offs between bandwidth, power consumption and latency.

Another proposal to use more than one radio in WSNs is the work of Kohvakka et al. [2006]. They present the design and implementation of a WSN platform that have four independent radio transceivers, which allow simultaneous reception and transmission. The authors claim that the benefits of their platform are high interference tolerance, low latency, and high mesh-networking performance. The architecture they present uses four equal radios, therefore, they all operate in the same frequency band. But the authors say that the hardware reconfigurability allows to adapt their platform with different radios. In this work, they also propose an application with high data rate by the use of multiple parallel routes, but with no implementation or experiments to evaluate the performance of this proposed scheme. Another drawback is that there is also no evaluation of how much energy the platform spends.

The Shimmer mote proposed by Burns et al. [2010] is another example of a two-radio platform for WSNs. It is specifically designed to support wearable applications, where keeping the size as small as possible is a very important and must be carefully balanced against other features in the design. This mote features a 802.15.4 and a Bluetooth radio. The authors say that Shimmer has been actively used for kinematic, physiological and ambient sensing applications in the context of healthcare systems.

There is also in the literature the s-Mote, developed by Popovici et al. [2011]. The authors present it as a versatile heterogeneous multi-radio platform for WSNs. One of the goals for the design of this platform is to achieve maximum interoperability between various networks. It can communicate concurrently using a variety of combinations of ISM bands. So, the platform does not specify which radios it uses, instead, it provides two slots that can accommodate ISM radio transceivers.

Then, the most recent work about a multi-radio platform for WSNs is the Opal mote, presented by Jurdak et al. [2011]. Opal includes two onboard 802.15.4 radios operating in the 900 MHz and 2.4 GHz bands to provide radio diversity. What makes Opal a more appealing platform is that the authors present experiments that evaluate Opal's throughput, range and energy consumption, comparing their results with single-radio platforms. They showed that Opal increases throughput by a factor of 3.7 when compared with platforms with just one radio with the same data rate. The authors claim that their architecture allows to achieve an aggregate transfer rate of 3 Mbps. To compare the energy efficiency between the platforms, they introduced a new metric, called spatial energy cost, which measures the energy to transfer one bit of information in a unit area.

Between all of these platforms, Opal is the only one that has a large-scale sensor network testbed for public access. Twonet is a testbed that has 100 Opal nodes, presented by Li et al. [2013] and located at the University of Houston, in the USA. Although testbeds generally are limited to a few tens or hundreds of nodes and a fixed network topology, they provide means to realistic, moderate-scale evaluation of sensor network algorithms, being an important tool for the development of new protocols. Having a testbed available enables us to research about new algorithms and evaluate their performance on the real world.

As we have seen, multi-radio platforms for WSNs started to be proposed quite a while ago, but they are gaining popularity only in the recent years. When we search for multi-radio protocols in the literature, the protocols we find are not specifically for WSNs, but they provide inspiration for what can be achieved when designing for multi-radio platforms.

The first work on the subject is a link layer protocol called the Multi-radio Unification Protocol or MUP, presented by Adya et al. [2004]. MUP coordinates multiple 802.11 radios operating over multiple channels. It improves performance using multiple radios, but on a single node, assuming shortest-path routing and homogeneous radios. Therefore, MUP does not consider characteristics of the overall path in the network, making its decisions using only locally available information. Through experiments, the authors showed that a network using their protocol achieve significant improvements on throughput and delay.

Another approach in routing in multi-radio wireless networks is presented by Draves et al. [2004]. In their work, the authors present a new metric for routing in multi-radio, multi-hop wireless networks, called Weighted Cumulative Expected Transmission Time or WCETT. This metric is calculated based on the Expected Transmission Time (ETT), assigning weights to individual links and then combining them. They incorporated the proposed metric in a protocol that they called Multi-radio Link-quality Source Routing or MR-LQSR. Unlike the MUP protocol, their work considers global characteristics of the network.

Subramanian et al. [2006] proposed an enhanced AODV routing protocol that they called AODV-MR. In their work, they present a new routing metric that they called iAWARE, that aids in finding paths that are better in terms of reduced interference. They assume that each node in the network is equipped with multiple radio interfaces. They showed that in contrast to existing link and path metrics at the time, their metric iAWARE tracks changes in interfering traffic, deliver higher throughput and finds paths with good channel diversity.

There is also the work of Shin et al. [2009], where they present an interface-aware multi-radio routing protocol that dynamically reconstructs a source-initiated path when radical link deterioration happens. It is based on MR-LQSR, but adds new features to make it adaptive to interference.

2.2 Multipath Routing Algorithms

Various multipath routing algorithms for WSNs in the literature aim to solve different problems. Some routing algorithms consider the problem of finding exactly two disjoint paths in a network. The algorithms presented by Ishida et al. [1995] and Ogier and Shacham [1989] are designed to find a pair of disjoint paths from each node to a single destination. Lee and Reddy [2010] present an algorithm to find two paths that are as disjoint as possible and the second path is as short as possible. Griffin and Korkmaz [2011] present an algorithm to check if the network contains at least two disjoint paths between all node pairs.

Another problem in many multipath routing algorithms is to find k disjoint paths between a source and a destination node, with a general k. Beginning with the centralized algorithms to approach this problem, there are those that will try to find any set of k disjoint paths and guarantee correctness [Khuller and Schieber, 1991; Iwama et al., 1997; Chen et al., 2004]. Other centralized algorithms find the k disjoint paths with minimum total length and guarantee correctness and optimality [Srinivas and Modiano, 2003; Bhandari, 1997; Suurballe, 1974; Hashiguchi et al., 2011]. The main problem of using such centralized algorithms in WSNs is the necessity of collecting every node's neighbor list to be aware of the whole topology, making the communication cost very high for large networks.

Distributed algorithms for a general k were also proposed in the literature [Ganesan et al., 2001; Li and Wu, 2005; Baek et al., 2007; Sidhu et al., 1991; Fang et al., 2009; Omar et al., 2011; Kumar and Varma, 2010]. These algorithms are efficient, but do not try to minimize the length of the routing paths. They find the paths one by one, and once a path is found, it is fixed and its internal nodes are removed from the network, so they cannot be selected to another path. On the other hand, Zhang et al. [2016] present a distributed algorithm for finding k disjoint paths with minimum length in WSNs called OFDP. Unlike the other decentralized algorithms, OFDP does not fix the paths it finds. Instead, it marks the nodes as occupied and consider the edges to be only backwards and with negative weights. Then, in each iteration it searches for augmenting paths in the network. The pseudo code for the main procedure of OFDP is shown in Algorithm 1. OFDP finds k disjoint paths between a source and a destination with minimum total length in the k^{th} round, if they exist. But none of the algorithms cited in this section consider the problem of finding disjoint paths with the same parity.

Algorithm 1 Optimally-Finding-Disjoint-Paths (OFDP)

```
   Input: two nodes $s$ and $t$; a positive integer $k$
   Output: min$k, k*$ disjoint $s \sim t$ paths with minimum total length
1: repeat
2:     Each node executes Procedure Finding-Shortest-Augmenting-Path;
3:     if cannot find an augmenting path then
4:         Stop and output all the found paths;
5:     end if
6:     Each node executes Procedure Tracing-Shortest-Augmenting-Path;
7: until Finding $k$ disjoint $s \sim t$ paths;
```

2.3 Achieving High Throughput in WSNs

Protocols developed for traditional applications in WSNs do not achieve high throughput. Their design prioritized minimizing the total energy consumption in each node of the network, to guarantee a longer lifetime of these nodes using a single battery. When high throughput stated to be a necessity in the traditional scenario with single radio platforms, many protocols specific for bulk data transfer were developed, such as Flush [Kim et al., 2007], FlushMF [Tavares et al., 2016], and PIP [Raman et al., 2010]. These protocols use a single path route between a source node and a destination, disable duty-cycling and reserve the radio channels until the transfer is over. They try to achieve high throughput by enforcing an end-to-end schedule of packets transmissions across each hop. These protocols consider the problem of achieving high throughput to be orthogonal to the low-power operation. A technique called Burst Forward presented by Duquennoy et al. [2011] allows the intermediate nodes to keep their duty-cycle and achieve the same throughput as the previous protocols. It achieves its goal by using a combination of high-throughput bursts and a store-and-forward method.

However all of these protocols are limited to a maximum theoretical end-to-end throughput of 50% of the channel capacity, which is a fundamental problem of single radio platforms. Furthermore, according to Raman et al. [2010],to eliminate self-interference these protocols use channel hopping, and the overhead of changing channels decreases the throughput such that the best performing protocols only achieve one quarter of the channel capacity.

With the development of dual-radio platforms for WSNs, such as Opal [Jurdak et al., 2011], this could be improved and theoretically reach 100% of the channel capacity. FastForward, presented by Ekbatanifard et al. [2013], is the only protocol in the literature to explore the advantages of using dual-radio platforms in order to maximize throughput in a WSN. FastForward transmits packets from a source node to a destination through a single path. The radio used to transmit the packets along the path are alternated in each hop. It assumes that one radio does not interfere with the other, which already helps to solve the problem of intra-path interference. Furthermore, it uses multiple fixed channels for each radio in each hop, reducing even more the effect of interference and eliminating the overhead of switching channels along the way. In Figure 2.1 we can see how FastForward uses the communication resources in dual-radio networks.

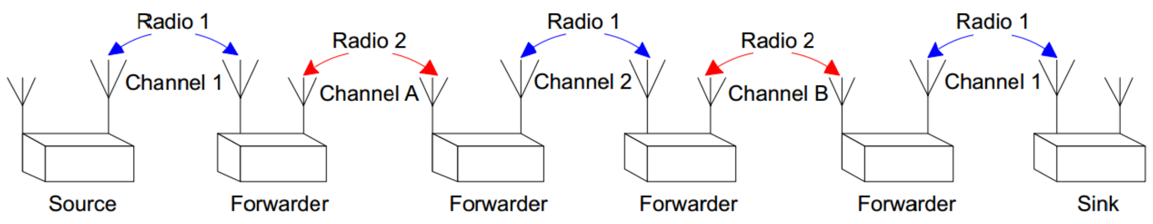


Figure 2.1: Example of how the FastForward protocol works

Chapter 3

Theoretical Basis

In this chapter, we discuss the theoretical basis of this work. First we formalize the notations and definitions used throughout the text. Then we explain the possible ways of modeling dual-radio wireless network and the related assumptions in each model. Finally, we state the problem for different modelings of the network and give our conclusions on the complexity of each version of the problem.

3.1 Preliminary Definitions

Here we present the notation, terms and its formal definitions used throughout this chapter. We will present the problem in four different types of graphs: undirected graphs, directed graphs, undirected multigraphs and directed multigraph. We assume all of these types of graphs to be weighted. From now on, we refer to them just as *graphs*, *digraphs*, *multigraphs* and *multidigraphs*, respectively.

A graph is represented as $G = (V, E)$, where V is a set of vertices and E is a set of *edges* each of which is a set of two vertices (u, v). A digraph is represented as $D = (V, A)$, where V is also a set of *vertices* and A is a set of *arcs* each of which is an ordered pair of vertices (u, v). A multigraph is represented as $M = (V, E, r)$, where V is a set of vertices, E is a set of edges and $r \colon E \to \{\{u, v\} \colon u, v \in V\}$ is a function assigning each edge to an unordered pair of endpoint vertices. A multidigraph (also called *quiver* in the literature) is represented as $Q = (V, A, s, t)$, where V is a set of vertices, A is a set of arcs, $s \colon A \to V$ is a function assigning each arc to its source node and $t \colon A \to V$ is a function assigning each arc to its target node. To denote different edges between the same vetices u and v, we use the notation $(u, v)_1$ and $(u, v)_2$. To denote the weight of an edge, we use $w(u, v)$ rather than $w((u, v))$, and $w(u, v)_1$ rather than $w((u, v)_1)$, for simplicity.

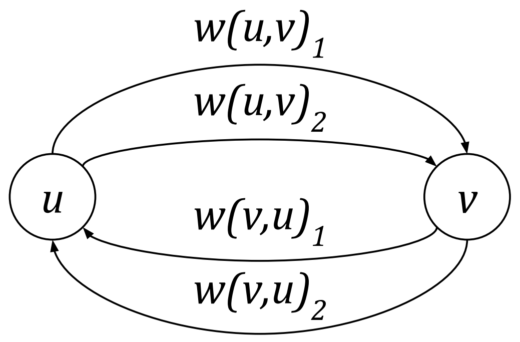
A *path* P in G is a subgraph that can be expressed as a sequence of vertices $P = (v_1, ..., v_m)$, where $v_i \in V$ and $(v_i, v_{i+1}) \in E$. We will refer to the number of edges in a path as the *number of hops* and represent it as $|P|$. The *parity* of the path is the rest of the division of its number of hops by 2. A path is *even* if $|P| \bmod 2 = 0$ and *odd* if $|P| \bmod 2 = 1$. The length of the path $l(P)$ is the sum of its edges' weights, i.e., $l(P) = \sum_{e \in E} w(e)$. A path is *simple* if no vertex appear on it more than once, i.e., $v_i \neq v_j$ if $i \neq j$. We will be always referring to simple paths, unless otherwise noted. Two paths $P_1 = (v_1^1, ..., v_m^1)$ and $P_2 = (v_1^2, ..., v_n^2)$ are *internally vertex disjoint* if every vertex in P_1, except the first and the last one, is different from the vertices in P_2. For now on, the term *disjoint* refer to internally vertex disjoint paths, for simplicity. All of these path definitions in graphs are the same for digraphs, multigraphs and multidigraphs.

3.2 Network Models

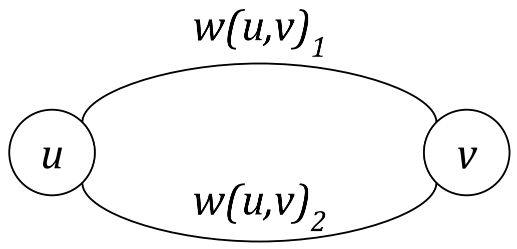
In a dual-radio WSN, there are four possible ways a message can be transmitted between a pair of nodes u and v: i) u sends a message to v via radio 1, (ii) u sends a message to v via radio 2, (iii) v sends a message to u via radio 1 or (iv) v sends a message to u via radio 2. If we want to represent each of the possible links between the two nodes, we need to model the network as a multidigraph, where each vertex represent a sensor node, and each pair of vertices may have up to four arcs between them, one for each case from (i) to (iv), and each vertex having its own weight. This is the most general case, where we make no additional assumptions about the communication links. This modeling is show in Figure 3.1a. There two arcs going from u to v, $(u, v)_1$ and $(u, v)_2$ and two going from v to u, $(v, u)_1$ and $(v, u)_2$, each one assigned with its own weight.

In wireless networks it is common to make an assumption that, if u can communicate with v via a certain radio, then v can also communicate with u through the same radio, and both links have the same weight. We say that the communication between u and v is *symmetric*. If we make this assumption, we can model the network as a multigraph, where each pair of vertices have at most two edges, each one representing a symmetric communication through each radio. This model can be seen in Figure 3.1b.

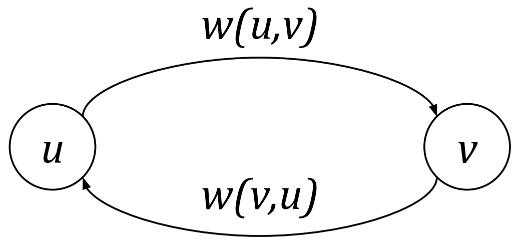
Another assumption it is possible to make about the network is that if u communicates with v via radio 1, then u also communicate with v via radio 2, and both links have the same weight. It is possible that a dual-radio platform has two radios of the same type, and in this case, this assumption is reasonable. Let's call this assumption



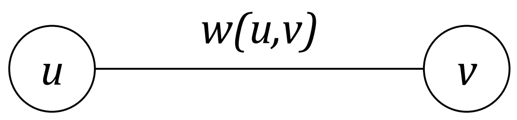
(a) Multidigraph



(b) Multigraph



(c) Digraph



(d) Graph

Figure 3.1: Modeling for different sets of assumptions in the network

a *homogeneous* communication between radios, when this assumption is true. If we consider that the links are homogeneous, but are not symmetric, we can represent the network as a digraph, as shown in Figure 3.1c.

Finally, if we assume that the communication between the sensor nodes are both symmetric and homogeneous, then we can represent the network as a graph, the simplest possible representation, as shown in Figure 3.1d.

Notice that if we can solve the problem for the more general representation, the multidigraph, then we can solve it for every other representation. A multigraph $M = (V, E, r)$ is a specific instance of a multidigraph $Q = (V, A, s, t)$ where if $(u, v)_r \in A$, $r = \{1, 2\}$, then $(v, u)_r \in A$ and $w(u, v)_r = w(v, u)_r$. A digraph $D = (V, A)$ can be represented as a multidigraph where if $(u, v)_1 \in A$, then $(u, v)_2 \in A$ and $w(u, v)_1 = w(u, v)_2$. And finally, a graph $G = (V, E)$ can be represented as a multidigraph where if $(u, v)_1 \in A$, then $(u, v)_2, (v, u)_1, (v, u)_2 \in A$ and $w(u, v)_1 = w(u, v)_2 = w(v, u)_1 = w(v, u)_2$.

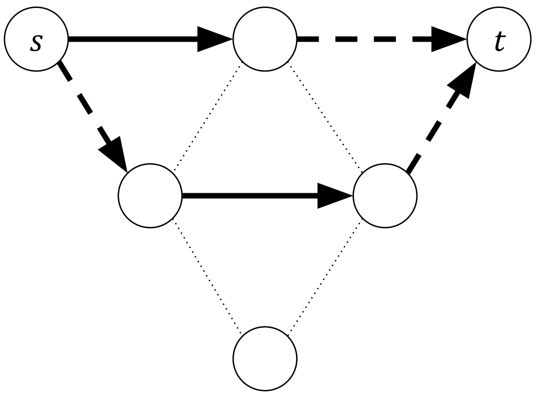
In the next subsection we will discuss our theoretical results for the problem applied in each one of these possible models for the network, and in Chapter 4 we present experiments showing the results when making each of these assumptions in a real world testbed topology.

3.3 Parity Disjoint Paths Problems

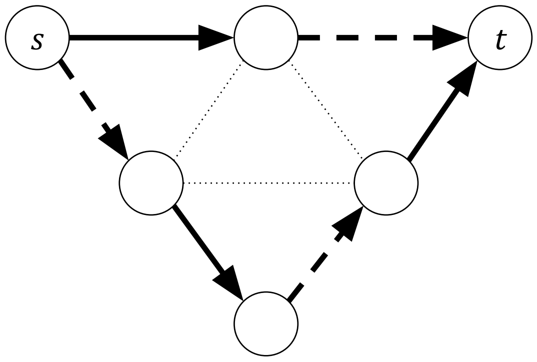
In our scenario, each node has two distinct radios that can operate simultaneously. Utilizing both radios in each node, it is possible to maximize data transmission from source to destination finding two paths in the network. Those paths should meet three constraints. First, the paths must be simple, and second, the paths must be disjoint. These constraints are intuitive, because if a node is chosen twice for one path, or is chosen to be a part of the two paths, it will not be able to receive and transmit simultaneously the required flow, creating a bottleneck in the network and losing the benefits of using two radios.

The third constraint is that both paths should have the same parity. We need two even paths or two odd paths. This is due to the fact that the source needs to transmit packets by different radios and the destination needs to receive the packets also by different radios, so that both can operate simultaneously. Figure 3.2 shows the reasoning for this constraint. In it, we have a network with several nodes, and possible two disjoint paths between the source node s and the final destination t. The paths chosen in Figure 3.2a have different parity, one has odd number of hops and the other has even. Since the source node needs to transmit by different radios and the intermediate nodes need to alternate their radios to send, the destination ends up receiving packets from both packets in the same radio, which is impossible to occur simultaneously. Differently, the path chosen in Figure 3.2b have the same parity. For this reason, the destination node ends up receiving the packets by two distinct radios, which can happen at the same time.

In the next subsections we study different versions of this problem. The problem vary in the type, which can be a decision or an optimization problem, and in the type of the graph, which can be a graph, digraph, multigraph or multidigraph. In the decision versions, which we are calling *parity disjoint paths problem* (or PDPP), we are only concerned in the existence of a pair of paths with lengths of the same parity, whereas in the optimization versions, which we are calling *minimum parity disjoint paths problem* (or MPDPP) we want to find among these existing pair of paths, one with the minimum total weight. Table 3.1 shows the results we got for each of these problems. We proof the decision and optimization problems in directed graphs to be NP-complete, and we



(a) Different parity



(b) Same parity

Figure 3.2: Example of two disjoint paths in a network.

proof the decision problem in undirected graph to be polynomial. The optimization problem in undirected graphs remains open, but we provide some insights with related problems.

	Decision Problem (PDPP)	**Optimization Problem (MPDPP)**
Graphs	Polynomial	?
Multigraphs	?	?
Digraphs	NP-complete	NP-complete
Multidigraphs	NP-complete	NP-complete

Table 3.1: Known complexities of the versions of the problem.

3.3.1 NP-Complete Problems

We start our analysis with the versions of the problem in digraphs and multidigraphs. Digraphs are a subset of multidigraphs, where there are no parallel arcs, therefore, if a problem is NP-complete for digraphs, it is also NP-complete for multidigraphs in general. Furthermore, if the decision problem is NP-complete, then its optimization problem is also NP-complete. Next we prove that the decision problem in digraphs is NP-complete, which implies that all of the problem's versions in digraphs and multi-digraphs are also NP-complete.

Given a digraph $D = (V, A)$ and two vertices $s, t \in V$, the decision version of the parity disjoint paths problem (or PDPP) is to verify if there are two simple paths P_1 and P_2, starting at the same source node s and ending in the same destination node t, with all intermediate nodes disjoint, with both paths having the same parity, i.e., $|P_1| \bmod 2 = |P_2| \bmod 2$.

We prove this is a NP-complete problem from the *even path in digraphs problem*, which is stated as follows: Given a digraph $D = (V, A)$ and s, $t \in V$, is there an even-length simple path from s to t? This problem is shown to be NP-complete by LaPaugh and Papadimitriou [1984]. Using this result, we can show the following:

Theorem 1. *Given a digraph $D = (V, A)$ and s, $t \in V$, it is NP-complete to decide whether there are two simple paths from s to t, with all intermediate vertices disjoint, and number of hops of the same parity.*

Proof. Fist of all, the parity disjoint paths problem is in NP because we can check a solution for the problem by counting the number of arcs in each path, comparing their parity and verifying that each intermediate vertex appears only once, and this can be done in polynomial time.

Once we know the problem is in NP, we will reduce the even path in digraphs problem to the disjoint parity paths problem. Given a digraph $D = (V, A)$ and s, $t \in V$, we construct a new digraph $D' = (V', A')$ as follows:

$$V' = V \cup \{s', t', n\}$$

$$A' = A \cup \{(s', s), (t, t'), (s', n), (n, t')\}$$

Basically, we add three new vertices and four new arcs, as shown in Figure 3.3, and this transformation is $O(1)$. We can see that D' is constructed in a way that it already has one simple path of even length from s' to t' going through n. The other path must include the arcs (s', s) and (t, t'), and it will be of even length if, and only if, there is a path of even length from s to t. Therefore, D' will have two disjoint simple paths of the same parity from s' to t' if, and only if, D has a even-length simple path from s to t.

$\square$

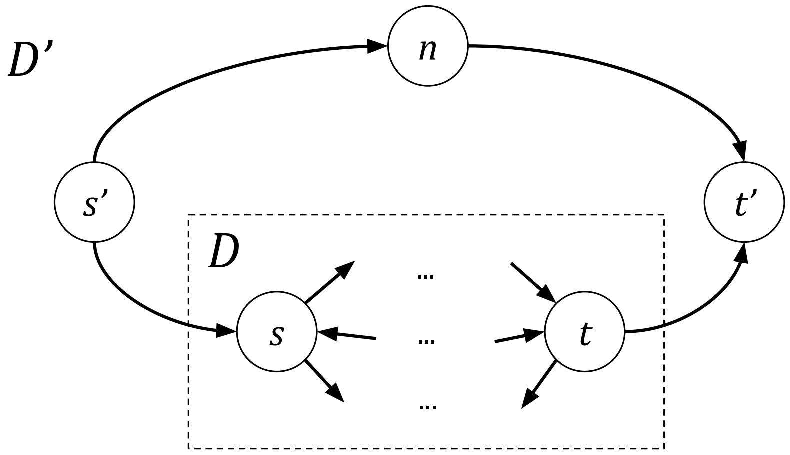


Figure 3.3: Reducing the even path in digraphs problem to the disjoint parity paths problem.

3.3.2 Polynomial Problem

In this subsection, we prove that the decision problem for graphs can be solved in polynomial time. Our problem is stated as follows: Given a graph $G = (V, E)$, $s, t \in V$ the decision version of the parity disjoint paths problem (or PDPP) is to verify if there are two simple disjoint paths P_1 and P_2 from s to t, with all intermediate nodes disjoint, with both paths having the same parity.

There is a more general problem in the literature, the *k parity disjoint paths problem* (k-PDPP), presented and proved to be polynomial by Kawarabayashi et al. [2011]. This problem is stated as follows:

Given a graph $G = (V, E)$ and k triples (s_i, t_i, p_i), $1 \leq i \leq k$, where s_i and t_i are vertices of G, $p_i \in \{0, 1\}$, and $\{s_i, t_i\} \cap \{s_j, t_j\} = \emptyset$, the *k parity disjoint paths problem* (or k-PDPP) is to determine whether there exist disjoint paths $P_1, ..., P_k$ such that the ends of P_i are s_i and t_i and the parity of P_i is p_i.

This is a more general problem in three aspects. The first one is that it considers the problem of finding k paths, while the PDPP concern is to find only 2 paths. The second one is that k-PDPP considers distinct sources and destinations for each path, which means that all the paths must be completely vertex disjoint, instead of only internally vertex disjoint. And third, is that k-PDPP allows for specifying any parity combinations for each paths, including combinations with different parities, whereas PDPP only wants the combinations with the same parity.

Kawarabayashi et al. [2011] prove that there exists an algorithm with time complexity equals to $O(nm\alpha(m, n))$ for k-PDPP, where n is the number of vertices, m is the number of edges and α is the inverse of the Ackermann function, which is an extremely slow growing function [Tarjan, 1983]. With this result, we can show the following:

Theorem 2. *There exists an $O(nm\alpha(m,n))$-time algorithm for the parity disjoint paths problem (PDPP) where n is the number of vertices, m is the number of edges, and α is the inverse of the Ackermann function.*

Proof. We now describe an algorithm to solve PDPP using the algorithm to solve k-PDPP. The pseudo code for the algorithm is presented in Algorithm 2. The first thing we need to do is a transformation of the input graph. Given a graph $G = (V, E)$ and $s, t \in V$, we construct a new graph $G' = (V', E')$ as follows:

$$V' = V \cup \{s', t'\}$$

$$E' = E \cup \{(s', v) \mid (s, v) \in E)\} \cup \{(t', v) \mid (t, v) \in E\}$$

An example of this transformation is shown in Figure 3.4, and an algorithm to perform this transformation is in the beginning of Algorithm 2, between lines 2 and 9. Basically we add two new vertices do the graph, s' and t', we add an edge between s' and all of the neighbors of the original s, and an edge between t' and all of the neighbors of the original t. The transformation in place, without copying the entire graphs, has time complexity $O(n)$, being the worst case when s and t are connected to all the vertices in the graph, in which case we need to add $(n-1)+(n-2) = 2n-3$ edges.

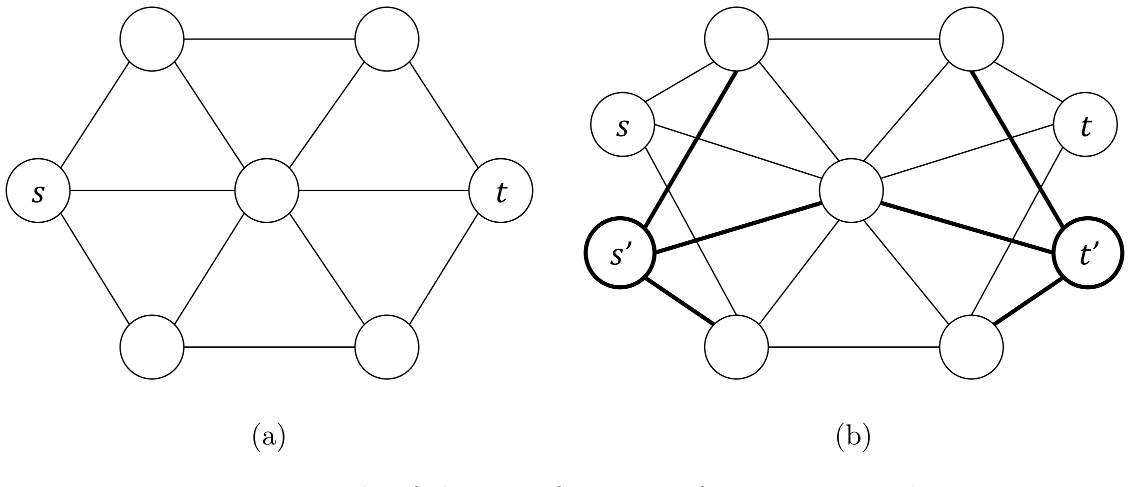


Figure 3.4: Example of the transformation from PDPP to k-PDPP

After the transformation, we need to use the algorithm to solve k-PDPP at most two times. The input for the fist execution is G', $k = 2$ and the two triples $\{(s, t, 0), (s', t', 0)\}$. If the algorithm returns true, then there exists two even paths

Algorithm 2 Solve PDPP using k-PDPP

```
1: **function** PDPP($G, s, t$)
2:     Add vertex $s'$ to $G$
3:     Add vertex $t'$ to $G$
4:     **for** each vertex $v$ adjacent to $s$ **do**
5:         Add edge $(s', v)$ to $G$
6:     **end for**
7:     **for** each vertex $v$ adjacent to $t$ **do**
8:         Add edge $(t', v)$ to $G$
9:     **end for**
10:    **if** K-PDPP($G, 2, \{(s, t, 0), (s', t', 0)\}$) **then**
11:        **return** TRUE
12:    **end if**
13:    **if** K-PDPP($G, 2, \{(s, t, 1), (s', t', 1)\}$) **then**
14:        **return** TRUE
15:    **end if**
16:    **return** FALSE
17: **end function**
```

between s and t in the original graph. If it returns false in the first execution, we execute it again, with inputs G', $k = 2$ and the triples $\{(s, t, 1), (s', t', 1)\}$. If this second execution returns true, then exists two odd paths between s and t in the original graph. If it also returns false, then there are no two paths between s and t with the same parity. Since there exists an algorithm to solve k-PDPP with time complexity $O(nm\alpha(n, m))$, and the described algorithm need to execute it at most two times and a $O(n)$ transformation described above, its complexity is also $O(nm\alpha(n, m))$.

□

3.3.3 Open Problems

The optimization problem for undirected graphs, and the decision and optimization versions for multigraphs remains open. For now, there are no polynomial algorithm to solve these problems in the literature, and there are no proof that they are NP-complete either. In this subsection, we give some insights about the optimization problem in undirected graphs.

The problem is stated as follows: Given a graph $G = (V, E)$ and two vertices $s, t \in V$, the *minimum parity disjoint paths problem* (or MPDPP) is to find the two simple disjoint paths P_1 and P_2, starting in s and ending in t with the same parity and the minimum total length $l(P_1) + l(P_2)$.

Since both paths must be disjoint and have the same parity, the union is the shortest cycle with an even number of edges that includes the vertices s and t. Thus, the MPDPP can also be viewed as the problem of finding the shortest even simple cycle that contains both s and t nodes. There is a corner case, when s and t are adjacent. The solution for MPDPP is $P_1 = P_2 = \{s, t\}$, whereas a single edge is not an even cycle in a undirected graph. Other than in this situation, the two problems are equivalent for a given graph G.

Other possibility is to reduce MPDPP to the *simple even path via a node problem*, which is given a graph $G = (V, E)$ and three vertices s, t and m, find the shortest simple even path with endpoints in s and t via m. We can transform an instance of MPDPP into an instance of the even path via a node by creating a new graph $G' = (V', E')$, duplicating the source node s into s and s' and connecting s' to all of the neighbors of s. Then solve the even path via a node to find the shortest even path from s to s' via t. Figure 3.5a depicts a solution for MPDPP and Figure 3.5b a corresponding solution for the even path via a node problem in G'.

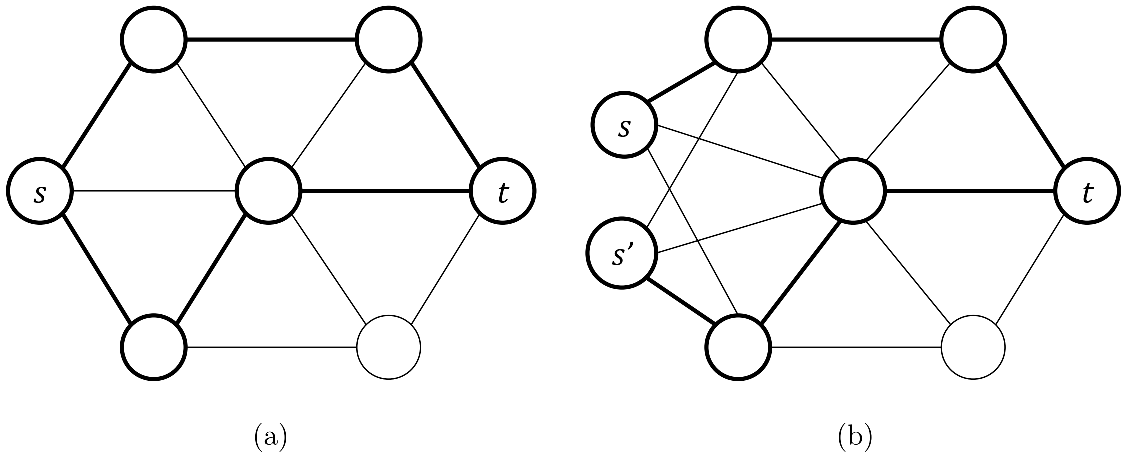


Figure 3.5: Example of the transformation from PDPP to the even path via a node problem.

The most general problem would be the *minimum k parity disjoint paths problem* (k-MPDPP), which is the optimization version of the k parity disjoint problem presented by Kawarabayashi et al. [2011]. We can reduce MPDPP to k-MPDPP the same way we described how to reduce the decision problem in Section 3.3.2, as shown in Figure 3.4. We didn't find k-MPDPP stated or solved in the literature.

Finally, the decision and optimization versions of the problem for multigraphs also remains with unknown complexity. It is worth noting that our models of the network have a limit of at most two parallel edges in the graph, but general multigraphs may have an arbitrary number of parallel edges. This general model can represent a multi-radio network with an arbitrary number of radios per node, instead of only two. If a polynomial solution is found for our specific dual-radio models, it doesn't mean that the general problem is polynomial. However, if the general case is polynomial, our particular case would be polynomial as well. These remain open problems.

Chapter 4

Development

In this chapter, we present the solutions we developed for the parity disjoint paths problem. First, we present a centralized solution as a model or integer linear programming and the implementation of a centralized routing scheme using it. Then, we present a distributed heuristic algorithm to solve the problem and the implementation of a routing protocol for dual-radio WSNs based on the algorithm.

4.1 Centralized Solution

To solve the problem with the path parity constraint, we model the problem using integer linear programming. The complete model is shown in Figure 4.1.

We represent the network as a multidigraph. For each arc (i, j) of the graph, we associate two different weights, c^1_{ij} and c^2_{ij}, each represents the cost to send one message from vertex i to vertex j. c^1_{ij} is the cost to send a message via radio 1 and c^2_{ij} is the cost to send via radio 2. For each arc, we also define two binary variables, x^1_{ij} and x^2_{ij}, that have value 0 if the arc (i, j) is not chosen for a path, or value 1, in case that arc is chosen. x^1_{ij} represents that radio 1 is chosen and x^2_{ij} represents that radio 2 is chosen. The goal to minimize the total cost, that is mapped by the sum of the cost of each radio multiplied by the correspondent x variable.

To model the constraints we need to formalize some definitions: we denote s the source node and t the destination node. For each vertex $i \in V$, we define two sets of arcs: $S(i)$ is the set of arcs leaving node i, and $E(i)$ is the set of arcs entering node i.

The constraints (i) and (ii) guarantee that the radio alternates between the chosen paths. In constraint (i), for each vertex i, the sum of the arcs leaving radio 1 minus the sum of the arcs entering by radio 2 must be 1 if the node is the source, -1 if it is the destination, or 0 for all the other cases. This guarantees that the source node will

$$\text{minimize} \sum_{(i,j)} \left(c^1_{ij} x^1_{ij} + c^2_{ij} x^2_{ij} \right)$$

subject to

$$\text{(i)} \quad \sum_{(i,j)\in S(i)} x^1_{ij} - \sum_{(j,i)\in E(i)} x^2_{ji} = \begin{cases} 1, \text{ if } i = s \\ -1, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases}$$

$$\text{(ii)} \quad \sum_{(i,j)\in S(i)} x^2_{ij} - \sum_{(j,i)\in E(i)} x^1_{ji} = \begin{cases} 1, \text{ if } i = s \\ -1, \text{ if } i = t \\ 0, \text{ otherwise} \end{cases}$$

$$\text{(iii)} \quad \sum_{j\in E(i)} x^1_{ji} + \sum_{j\in E(i)} x^2_{ji} \leq 1, \quad \text{if } i \neq t$$

$$\text{(iv)} \quad \sum_{(i,j)\in A} x^1_{ij} - \sum_{(i,j)\in A} x^2_{ij} = 0$$

$$\text{(v)} \quad x^1_{ij}, x^2_{ij} \in \{0, 1\}$$

Figure 4.1: Integer linear programming model to solve the shortest dual path parity routing problem.

not receive by radio 2 and for sure will send through radio 1. We also guarantee that the destination will receive by radio 2 and will not send by radio 1. For all the other nodes, if it receives by radio 2, it will definitely send by radio 1.

Similarly to previous constraint, in constraint (ii) for each vertex i, the sum of the arcs that leave radio 2 minus the sum of the arcs that enter by radio 1 must be 1 for the source, -1 for the destination and 0 for the other cases. This guarantee that the source node will not receive by radio 1 neither and for sure will send by radio 2. The destination node will for sure receive by radio 1 and will not send by radio 2. For all the other nodes, if it receives by radio 1, it will send via radio 2.

Constraint (iii) guarantees that no intermediate node will be chosen for two paths, making the sum of arcs entering in all node i, except the destination node, be less or equal to 1. As the variables are binary, either radio 1 is chosen and radio 2 is not, or radio 2 is chosen and radio 1 is not, or none of them is chosen. For the destination node, two entering arcs are chosen, one for each radio, as guarantee constraints (i) and (ii). Constraints (i) and (ii) also guarantee that we do not need to create a constraint for the number of leaving arcs for each node, as it must be equal to the number of entering arcs in each intermediate node.

Finally, constraint (iv) is the one that guarantees that both paths will have the same parity. It says that the number of arcs where radio 1 is chosen is equal to the number of arcs where radio 2 is chosen. Since the nodes always alternate the radios in the paths, it enforces them to have the same parity. Constraint (v) determines that variables x_{ij}^1 and x_{ij}^2 are binary.

4.2 Distributed Algorithm

Our distributed algorithm works by finding a shortest path between s and t of a certain parity, and then trying to find an augmenting path of the same parity of the one that was found before. A pseudo code of the main procedure is shown in Algorithm 3. It has two main procedures, called FINDPATH and TRACEPATH and two phases.

Algorithm 3 Main procedure

```
 1: procedure PDPP($Q, s, t$)
 2:                                            ▷ First phase
 3:     FINDPATH($s, t, 0$)                    ▷ Path starting through radio 0
 4:     TRACEPATH($s, t, 1$)                   ▷ and ending through radio 1
 5:     if a path was found then
 6:         FINDPATH($s, t, 1$)
 7:         TRACEPATH($s, t, 0$)
 8:         if two paths were found then
 9:             paths0 ← GETPATHS($Q$)
10:         end if
11:     end if
12:                                            ▷ Second phase
13:     FINDPATH($s, t, 0$)                    ▷ Path starting through radio 0
14:     TRACEPATH($s, t, 0$)                   ▷ and ending through radio 0
15:     if a path was found then
16:         FINDPATH($s, t, 1$)
17:         TRACEPATH($s, t, 1$)
18:         if two paths were found then
19:             paths1 ← GETPATHS($Q$)
20:         end if
21:     end if
22:
23:     return MINIMUM(paths0, paths1)
24: end procedure
```

In the first phase, the algorithm tries to find a path and an augmented path of even length. It does that by enforcing the source node to transmit its first FIND message through a specific radio, and the destination node to accept paths that result in a

FIND message received through the other radio. Intermediate nodes always alternate the radios they receive and send messages. We use the notation $\overline{r}$ to indicate the opposite radio, so, if a node receives through r, it sends through $\overline{r}$.

Similarly, in the second phase, the algorithm tries to find a path and an augmented path of odd length, enforcing the destination node to only accept FIND messages from the same radio r that the source node used to transmit its first FIND message. Trying to find a path in the network that has a certain parity induce an odd cycle in some path. To avoid cycles, the FIND packets include the path until node v, and if v is in the path, it just ignores the message.

A node may be marked as free or occupied. Being marked as a free node means that it is not a part of any current paths. At the beginning, all nodes start marked as free nodes, and becomes occupied if it is chosen to be a part of a path in a TRACE step. If a node is occupied, it knows the nodes that are its predecessor and the successor in the path, in variables we called $prev_r(v)$ and $next_r(v)$, where r indicates which is the radio used to receive or send messages from the predecessor or the successor.

Let v be any node in Q. We assume that v knows all its neighbors and the weights of the incident arcs. The header of each message contains its sender's ID, so the receiver is able to know where an incoming message is coming from. In the next subsections we describe the algorithms for each procedure of the algorithm.

4.2.1 Finding an augmenting path

The pseudo code for the FINDPATH procedure is presented in Algorithm 4. Each node v maintains variables $d_r(v)$ and $p_r(v)$. $d_r(v)$ records the length of the shortest path from s to v such that the FIND message was received by v through radio r, and $p_r(v)$ records the associated path.

The nodes try to find an augmenting path by exchanging FIND messages. Each FIND message sent by u via radio r has a field $d(u)$, which is the length of the shortest path from s to u that arrived in u via radio $\overline{r}$ found until now. The other field in the FIND message, $p(u)$, is the node list that represents the shortest path associated with $d(u)$.

At the beginning of a FINDPATH procedure, s broadcasts a FIND message with $d(s) = 0$ and $p(s) = \{\}$ via radio r, which is defined in the main procedure of the algorithm, to start the process. When a node $v \neq s$ receives a FIND message $\{d(u), p(u)\}$ from u via radio r, there are five possible cases we discuss next:

Case 1: v is in the received path $p(u)$. Ignores the message received in this condition because it indicates a loop in the path.

Algorithm 4 FindPath procedure

```
1: procedure FindPath($s, t, r$)
2:     $s$ broadcasts {0, {}} via radio $r$
3:     while $v$ receives $\{d(u), p(u)\}$ via $r$ do
4:         case $v$ is in $p(u)$:
5:             Ignore message
6:         case $u$ is $prev_r(v)$ or $u$ is $prev_{\overline{r}}(v)$ or $u$ is $next_{\overline{r}}(v)$:
7:             Ignore message
8:         case $v$ is a free node:
9:             $dist(v) \leftarrow d(u) + l_r(u, v)$
10:            if $dist(v) < d_r(v)$ then
11:                $d_r(v) \leftarrow dist(v)(v)$
12:                $p_r(v) \leftarrow p(u) \cup \{u\}$
13:                Broadcast $\{d_r(v), p_r(v)\}$ via $\overline{r}$
14:            end if
15:        case $v$ is an occupied node and $u$ is not $next_r(v)$:
16:            $dist(v) \leftarrow d(u) + l_r(u, v)$
17:            if $dist(v) < d_r(v)$ then
18:                $d_r(v) \leftarrow dist(v)$
19:                $p_r(v) \leftarrow p(u) \cup \{u\}$
20:                if $v \neq t$ then
21:                    Send $\{d_r(v), p_r(v)\}$ to $prev_r(v)$ via $r$
22:                end if
23:            end if
24:        case $v$ is an occupied node and $u$ is $next_r(v)$:
25:            $dist(v) \leftarrow d(u) - l_r(v, u)$
26:            if $dist(v) < d_r(v)$ then
27:                $d_r(v) \leftarrow dist(v)$
28:                $p_r(v) \leftarrow p(u) \cup \{u\}$
29:                Send $\{d_r(v), p_r(v)\}$ to $prev_{\overline{r}}(v)$ via $\overline{r}$
30:                Broadcast $\{d_r(v), p_r(v)\}$ via $r$
31:            end if
32:    end while
33: end procedure
```

Case 2: u is the predecessor of v in a already established path or u is the successor of v through the other radio. The only possible communication between nodes that are in a established path is from its successor through the related radio, indicating a backward segment in the path. In any other situation it just ignores the message.

Case 3: v is a free node. The algorithm uses a variable $dist(v)$ to record the length from s to v via radio r. Let $dist(v)$ be $d(u) + l_r(u, v)$, where $l_r(u, v)$ is the cost of sending a message from u to v via radio r. If $dist(v) < d_r(v)$, it means that the path that just arrived via radio r is shorter than the shortest path found until now, then it lets this be the new shortest path from s to v via radio r. Also, if the path is updated, v broadcasts a FIND message $\{d_r(v), p_r(v)\}$ via radio $\overline{r}$, to alternate the radios.

Case 4: v is an occupied node and u is not $next_r(v)$. This means v received a message from a free node, thus the arc (u, v) is not part of an established path. Let $dist(u)$ be $d(u) + l_r(u, v)$, and compare with the current shortest path. If the shortest path is updated and $v \neq t$, v sends a FIND message $\{d_r(v), p_r(v)\}$ just to $prev_r(v)$, otherwise it would allow an augmenting path to cross an occupied node.

Case 5: v is an occupied node and u is $next_r(v)$. The later condition means that the arc (v, u) is part of a selected path using radio $\overline{r}$. Then, it lets $dist(v)$ be $d(u) - l_{\overline{r}}(u, v)$, to allow our path to go through this back arc, and eliminate it if this augmenting path is chosen. It needs to store the distance and the path in the variables $d_r(v)$ and $p_r(v)$, respectively. Then, it needs to make two transmissions: one to $prev_{\overline{r}}(v)$ via radio $\overline{r}$ and the other broadcasting a FIND message $\{d_r(v), p_r(v)\}$ via radio r. The first transmission is to find the cases where the augmenting path continues going through the back edges, and the second transmission is to find the case where the backward segment ends in node v, returning to pass through free nodes. This second message didn't have to be sent to $prev_{\overline{r}}(v)$, but we choose to make a broadcast instead of various unicasts to simplify and reduce the number of transmissions. It ignores the extra message received in case 2.

4.2.2 Tracing an augmenting path

The TRACEPATH pseudocode is shown in Algorithm 5. This procedure is started by t. When t receives its first FIND message in the previous FINDPATH procedure, it waits for a long enough time to start the execution of TRACEPATH. Each TRACE message has one field $p_r(t)$, which is the list of nodes in the shortest augmenting path that arrived in t via radio r. The radio r is a parameter defined by the main procedure of the algorithm.

Node t starts by sending a TRACE message to its predecessor in the chosen path. When a node v receives a TRACE message from u, we define two new variables $succ(v)$ and $pred(v)$, that receives the successor and the predecessor of v in the received path $p(t)$.

Case 1: v is a free node. This is a trivial case. It just marks itself as an occupied node and sets its $prev(v)$ and $next(v)$ to $pred(v)$ and $succ(v)$, respectively.

Case 2: v is an occupied node and $succ(v)$ is not $prev(v)$. This means that node v is the last hop of a backward segment. It is the case of node a in the execution

Algorithm 5 TracePath procedure

```
 1: procedure TRACEPATH($s, t, r$)
 2:     $pred(t) \leftarrow$ predecessor of $t$ in $p_r(t)$
 3:     $t$ sends $\{p_r(t)\}$ to $pred(t)$ via $r$
 4:     while $v \neq s$ receives a new $\{p(t)\}$ via $r$ do
 5:         $succ(v) \leftarrow$ successor of $v$ in $p(t)$
 6:         $pred(v) \leftarrow$ predecessor of $v$ in $p(t)$
 7:         case $v$ is a free node:
 8:             Mark itself as an occupied node
 9:             $prev_{\overline{r}}(v) \leftarrow pred(v)$
10:             $next_r(v) \leftarrow succ(v)$
11:             Send $\{p(t)\}$ to $pred(v)$ via $\overline{r}$
12:         case $v$ is an occupied node and $succ(v)$ is not $prev_r(v)$:
13:             Still mark itself as an occupied node
14:             $next_r(v) \leftarrow succ(v)$
15:             Send $\{p(t)\}$ to $pred(v)$ via $r$
16:         case $v$ is an occupied node and $succ(v)$ is $prev_r(v)$ and $pred(v)$ is $next_{\overline{r}}(v)$:
17:             Mark itself as a free node
18:             Send $\{p(t)\}$ to $pred(v)$ via $\overline{r}$
19:         case $v$ is an occupied node and $succ(v)$ is $prev_r(v)$
                    and $pred(v)$ is not $next_{\overline{r}}(v)$:
20:             Still mark itself as an occupied node
21:             $prev(v) \leftarrow pred(v)$
22:             Send $\{p(t)\}$ to $pred(v)$ via $r$
23:     end while
24: end procedure
```

example of Figure 4.2b. In this case, node v changes its $next(v)$ variable to $succ(v)$, and keep its old $prev(v)$.

Case 3: v is an occupied node and $succ(v)$ is $prev(v)$ and $pred(v)$ is $next(v)$. It means that node v is in the middle of a backward segment, illustrated by node b in Figure 4.2b. v marks itself as a free node, since it will not be a part of any path anymore.

Case 4: v is an occupied node and $succ(v)$ is $prev(v)$ and $pred(v)$ is not $next(v)$. In this case, v is the first node of a backward segment, as node c in the example of Figure 4.2b. In this case, node v needs to change its $prev(v)$ variable to $pred(v)$.

4.2.3 An example of the algorithm's execution

Figure 4.2 shows an example execution of the distributed algorithm. Consider that each arc without a value assigned has weight 1. Also, all the arcs have the same weight

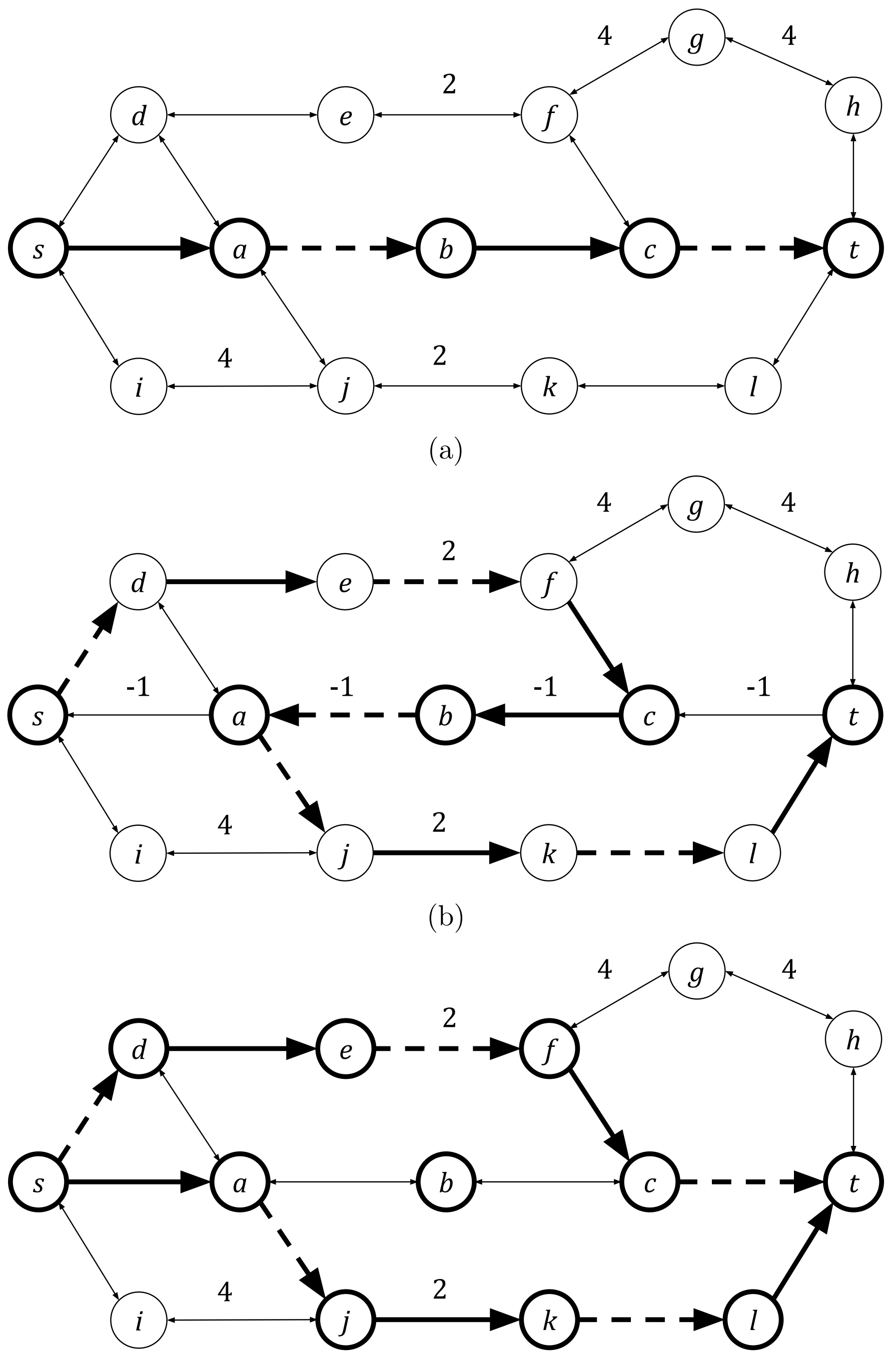


Figure 4.2: Example of the distributed algorithm's execution

in both ways.

Figure 4.2a shows the result of the first round of FINDPATH and TRACEPATH procedures. It finds a path starting through radio 0 (continuous arrow) and ending through radio 1 (dashed arrow). Now each of these arcs (u, v) will be removed, and the back arc (v,u) through the opposite radio will be assigned with a negative weight equal of the removed arc.

In Figure 4.2b is the result of the second round of FindPath procedure. As in the previous round s started sending through radio 0 and t received through radio 1, in this round, s starts sending through radio 1 and t accepts paths received through radio 0. The first path t will receive through radio 0 will be $\{s, d, e, f, g, h, t\}$, with total weight of 13. But when c receives a FIND message from f, it will follow case 4 and send a message to b. Then b will be in case 5, store the path coming from a back arc and broadcast a path with length $5 - 1 = 4$. Node a will receive the message through another back arc, also fall in case 5, and broadcast a path with length $4 - 1 = 3$. It will be received by j, and from it it will continue until it gets to t through radio 0 with length 8. Then t will update its path to the path $\{s, d, e, f, c, b, a, j, k, l, t\}$, with contains two back arcs.

Finally, in Figure 4.2c is the result of the second round of TracePath procedure. The TRACE message will make the free nodes l, k, j, f, e and d fall in case 1, which is straightforward. Node a will execute case 2 and change $next(a)$ to j in the new path. Then node b will execute case 3 and mark itself as a free node and not be part of any path anymore. Node c will be on case 4, and change $prev(c)$ to f in the augmenting path. We will finish the algorithm with two same parity paths with total length of 12.

4.2.4 Complexity Analysis

We analyze the complexity of our distributed heuristic algorithm in two aspects: the number of messages exchanged and the time it takes to complete each procedure. For the message complexity, we assume that each unicast counts as 1 message sent, and each broadcast also counts for only one message sent. For the time complexity, we assume that event processing takes no time and that a message is received at most one time unit after it is sent.

Fist, let's analyze the FindPath procedure in terms of number of messages. Each node makes a transmission when either $d_0(v)$ or $d_1(v)$ is updated. Every node in the network will update each variable at least once, so the number of messages sent is at least $2n$, where n is the number of nodes in the network. The maximum number of messages exchanged is related to the maximum number of times each variable in a node can be updated in the process. Each variable $d_r(v)$ always contains the length of a simple path in the network, and only decrease it's value to another value that is also the length of another simple path in the network. The maximum number of nodes in a simple path in the network is n, therefore, each variable will be updated at most n times. Therefore, an upper bound on the messages sent in this process is $2n \times n = 2n^2$.

The message complexity of the find message is $O(n^2)$.

Now, analyzing the time complexity, each time a node v broadcasts a message with a path including n hops, its neighbors will have a path of $n + 1$ hops. The maximum number of hops of an augmenting path will be three times the diameter of the network in number of hops, which would be a case where the path goes back almost all the back arcs. Therefore, the time complexity of this procedure is $O(diam)$, where $diam$ is the diameter of the network.

After evaluating and executing the proper case, every node in the path that received a TRACE message must retransmit the message to its predecessor. In the standard case, the message complexity of the TracePath procedure will be $O(diam)$ messages, where $diam$ is the diameter of the network. But the TRACE message has to be delivered to every node in the selected path. When we do not assume that all communication links are at least partially symmetric, any arc in the selected path might not have a back arc. When an arc in a selected path (u, v) doesn't have a back arc (v, u), the simplest way to work around it is v starts broadcasting the TRACE message until it gets to u. If this is necessary, the message complexity of TracePath will be $O(n)$ in number of messages in the worst case. The time complexity will be always $O(diam)$.

4.3 Distributed Implementation

The distributed algorithm was implemented using TinyOS 2.1.2 for the Opal platform [Jurdak et al., 2011]. This platform has two 802.15.4 radio transceivers that operate in different bands: 900 MHz and 2.4 GHz. The two radios share the same SPI bus, which creates a bottleneck for data transfer between the radios and the microcontroller. However, data transfer on the bus is much faster than data transmission over the radio. On the Opal platform, sending an SPI packet to the transmission buffer takes less than 10% of the time it takes to transmit the packet over the radio [Ekbatanifard et al., 2013]. As the protocol seeks to keep the radios always busy, the two radios will be operating at the same time most of the time.

Figure 4.3 shows an overview of the architecture implemented for the distributed algorithm. It was implemented over the communication protocol stack for TinyOS: the radio driver, and the implementation of the MAC and link layers. It offers options for configuring the modulation used by radios, transmit power, Clear Channel Assessment (CCA) on/off, and random backoff parameters used. There is still the possibility of enabling or disabling the use of acknowledgment packets provided by the link layer.

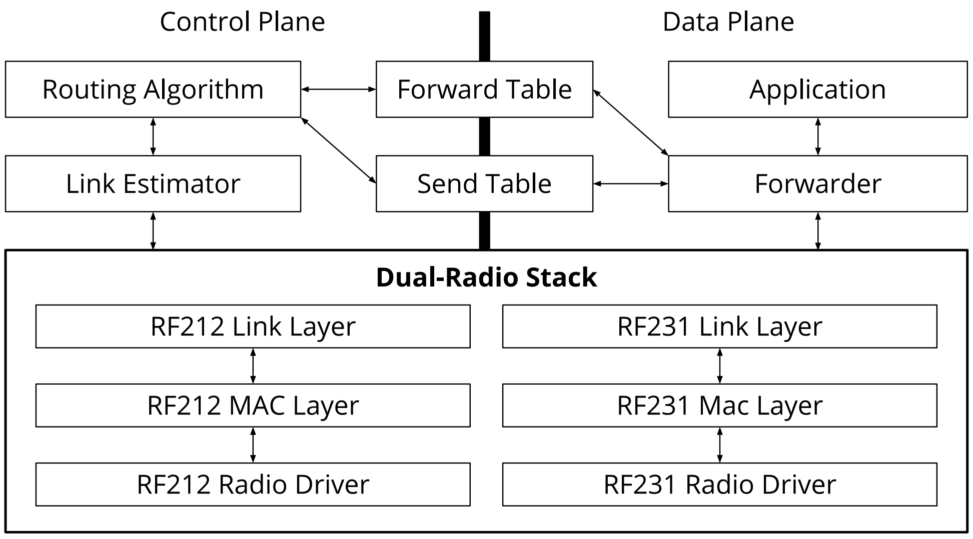


Figure 4.3: Overview of the implemented architecture.

Our current implementation is specific for point-to-point multipath routing, and the fist to address this problem. Existing routing protocols for wireless sensor networks, such as RPL and CTP prioritize data collection, which means communicating from every node to a single source node. However, a network application, for instance, a surveillance system, could integrate both our protocol and the traditional ones for different tasks. We can define the packet error rate as the routing metric for both protocols, and use the traditional protocols for collecting video meta data informing if anything different is happening and use our protocol when transmitting the whole video is necessary.

Chapter 5

Experiments and Results

In this chapter, we present the experiments we did to evaluate the network models trade-offs based on a real-world testbed, to evaluate the heuristic algorithm and to measure the throughput improvement when using multipath routing in data streaming, comparing it with FastForward, the state-of-the-art protocol for high throughput in dual-radio WSNs.

5.1 Model Evaluation

We presented in Section 3.2, four different possible network models for dual-radio WSNs. To recap, the most general is a multidigraph with at most four parallel arcs between each pair of nodes. The representation of the network as a multigraph assumes the communication of each radio to be symmetric, the representation as a digraph assumes the communication to be homogeneous among the radios, and the representation as a graph assumes it to be symmetric and homogeneous.

Our fist set of experiments evaluate the symmetry and homogeneity of the wireless links on a large-scale dual-radio wireless sensor network testbed called Twonet [Li et al., 2013]. It contains 100 dual-radio sensor nodes of the Opal platform [Jurdak et al., 2011]. First, we describe how we collected the topology of the testbed and the metric used to estimate the weights in the multidigraph model. Then we analyze the symmetry and homogeneity in this model. We calculate the routes assuming each of these properties and compare the results with the original model.

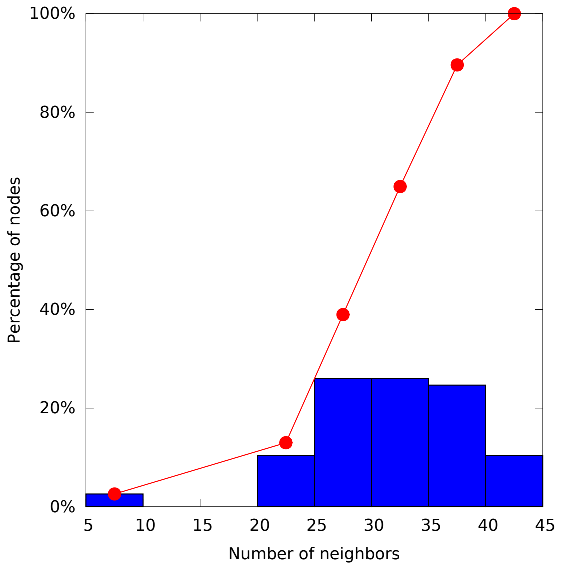
5.1.1 Testbed Topology Characterization

The first thing we did was to collect the complete topology of the testbed and represent it in a multidigraph, where $w(u,v)_r$ represents the cost of transmitting a packet from u to v through radio r. We uploaded a program where each node broadcast a total of 100 packets through each radio, with a random interval between 1 and 2 seconds to each transmission. When a node v receives a packet from node u through radio r, it adds u to a list of neighbors where each entry has two counters, one for each radio, and increments the counter for radio r. At the end of the experiment, each node v has a counter $c(u,v)_r$ for the number of messages received from each neighbor u through radio r. The metric we use to assign the weights of each link in the testbed topology is the Packet Error Rate (PER), which is the amount of packets lost in each link. u will only be in the neighbors' table of v if at least one packet is received. If no message was received through a certain link, the multidigraph model of the network does not contain the related arc.

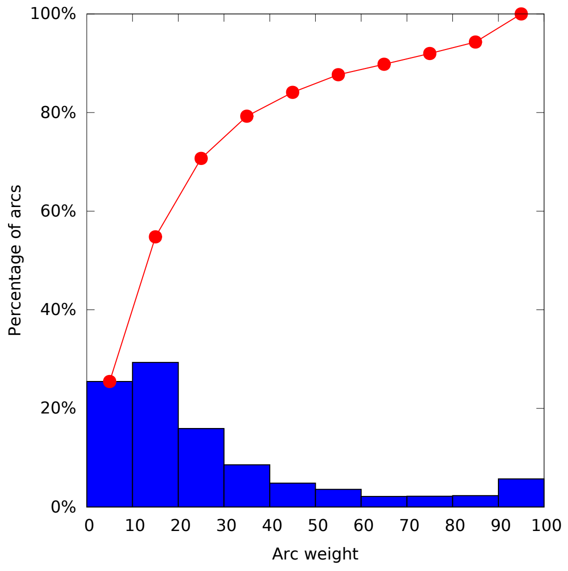
Though the testbed contains 100 sensor nodes, only 77 nodes returned the results of the topology collection, which includes messages received by that node. The remaining nodes were not listed as messages senders in any log, which means that the remaining nodes were not working at the time, and it might have been caused by an error in those nodes. Therefore, our topology actually has 77 nodes, and since between each pair there could be up to 4 arcs, the maximum number of arcs in the network would be 11704 arcs. In our model, there are 4030 arcs, which means that the density of the testbed is approximately 34.4%.

Figure 5.1a shows the distribution of nodes in the testbed by the number of neighbors they have, and the cumulative distribution. The average number of neighbors for each node in the testbed is 32. Only two nodes have between 5 and 10 neighbors. Less than 13% of the nodes have less than 25 neighbors. The majority of the nodes, about 76.6% have between 25 and 40 neighbors in the testbed. The most connected nodes, having between 40 and 45 neighbors are 10.3% of the total.

Figure 5.1b shows the distribution of arcs by weight and the related cumulative distribution. The average arc weight in the testbed is 28. We can see that 25% of the total arcs have its weight between 1 and 10, which are the best quality links. The majority of the arcs have weight less or equal to 30%, and only about 12.4% of the nodes have its weight greater than 50.



(a) Neighbors distribution. Total: 77 nodes.



(b) Arc weight distribution. Total: 4030 arcs.

Figure 5.1: Distribution of the testbed's nodes by number of neighbors and arcs by weight.

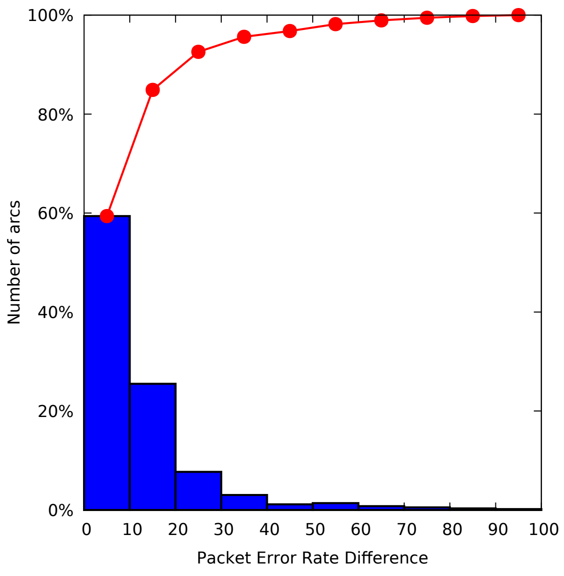
5.1.2 Symmetry and Homogeneity

Now we analyze the symmetry in the testbed multidigraph model. Starting with general statistics, only 126 arcs $(u, v)_r$ do not have a corresponding arc $(v, u)_r$, which correspond to 3% of the total of arcs. We call these arcs completely asymmetric. About 93% of the arcs in the testbed topology have both arcs $(u, v)_r$ and $(v, u)_r$, but $w(u, v)_r \neq w(v, r)_u$. We call these arcs partially asymmetric. Only 4% of the arcs have $w(u, v)_r = w(v, u)_r$, these are completely symmetric links.

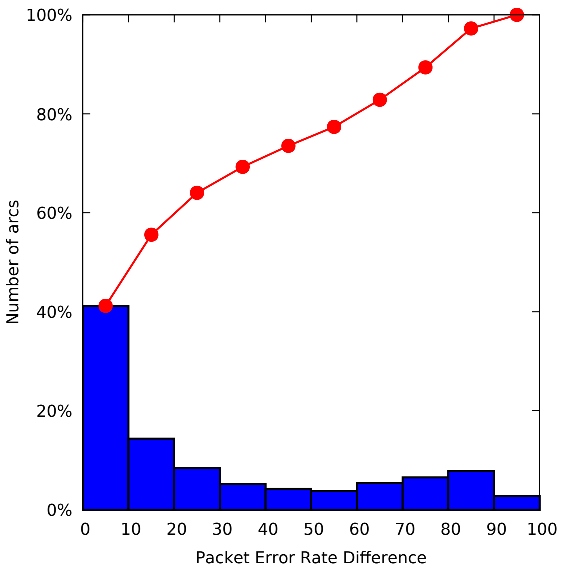
In Figure 5.2a we can see a histogram of the weight differences between each pair of arcs $(u, v)_r$ and $(v, u)_r$. For each of these pair of arcs, we calculate the absolute value of $w(u, v)_r - w(v, u)_r$ and count the number of arcs with weight differences in intervals of 10. We can see that about 60% of the pair of arcs have a weight difference smaller than 10. It means that even if only 4% of the links are completely symmetric, most of the partially symmetric links have a small weight difference. Furthermore, only 7.4% of the arcs have weight difference greater than 30.

Regarding the homogeneity of the testbed model, about 19% of the arcs $(u, v)_r$ are completely heterogeneous, which means that the corresponding arc $(u, v)_{\overline{r}}$ is not present in the network. about 76% of the arcs are partially heterogeneous, which meas that both $(u, v)_1$ and $(u, v)_2$ are present in the network, but $w(u, v)_1 \neq w(u, v)_2$. Only 5% of the arcs are completely homogeneous, i.e, $w(u, v)_1 = w(u, v)_2$

In Figure 5.2b we can see a histogram of the weight difference between each



(a) Symmetry distribution



(b) Homogeneity distribution

Figure 5.2: Symmetry and homogeneity distributions in the testbed.

pair of arcs $(u,v)_1$ and $(u,v)_2$. We can see that about 40% of the pairs of arcs have a weight difference smaller than 10, but about 23% of them have weight difference greater than 50. Therefore, the model of the testbed has a lower level of homogeneity than symmetry, which implies that considering that the network is homogeneous would lead to greater errors in the final results.

The final step in our model evaluation is to quantify the additional costs that assuming the symmetry and homogeneity properties in the testbed would generate. To do that, we converted the multidigraph model of the testbed into three simplified models: a multigraph, a digraph and a graph. We only add an edge to the simplified models if the original model have all of the corresponding edges merged into a single one. So, in the multigraph and digraph models, each edge correspond to two original edges and in the graph model each edge correspond to four original edges. Then we choose the maximum weight among the original edges to be the weight of the new edge.

We ran the ILP model in each of these simplified models for all possible combinations of origin and destination nodes and compared the total weight of the paths obtained from the simplified models with the total weight from the original model. The additional cost is calculated as the difference between these values divided by the value obtained from the original model.

In Figure 5.3 we show the additional cost distribution for each of the simplified models in a boxplot graph. Quartile boundaries are determined such that 1/4 of the points have a value equal or less than first quartile boundary, 1/2 of the point have a value equal or less than the second quartile value, etc. In the boxplot, a box is drawn around the region between the first and third quartiles, with a horizontal line at second, which is the median value. Comparing the additional cost distribution of the multigraph and digraph models, we can see that the additional cost is bigger in the digraph model. Since we have shown that the testbed topology is more symmetric than homogeneous, it was expected that the additional cost in the digraph model is greater than the additional cost in the multigraph model. We can also see that assuming the graph model, the additional cost goes to up to almost 200%, having a median of about 95% increase in the value calculated from the original multidigraph model.

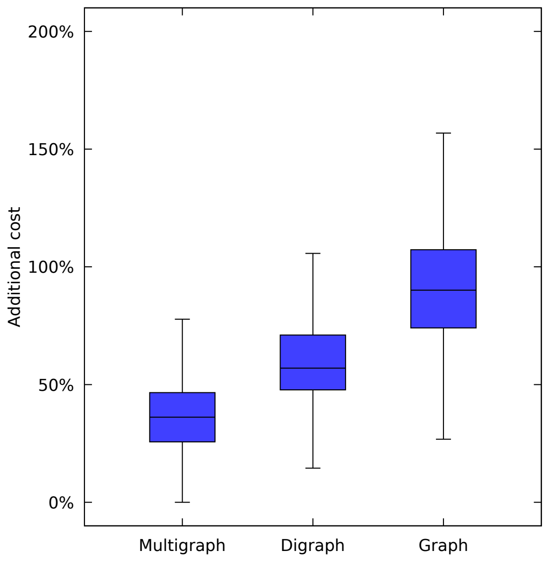


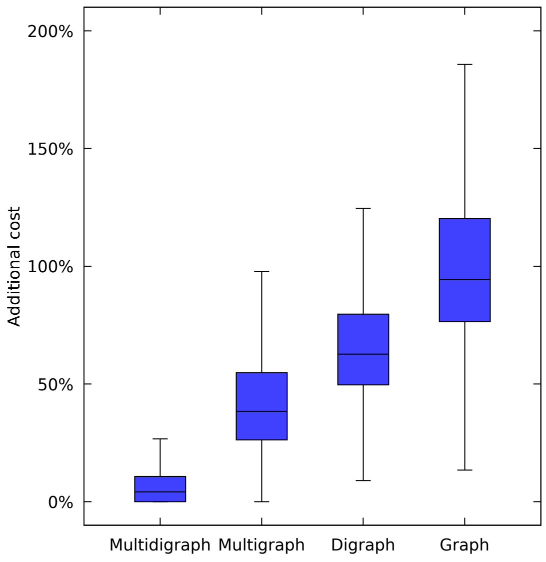
Figure 5.3: Additional cost distribution in the testbed topology assuming different models

5.2 Heuristic Evaluation

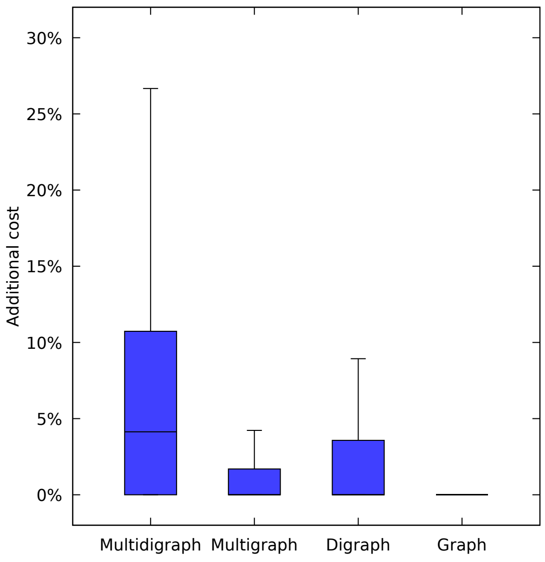
In this section we present experiments to evaluate the performance of our proposed heuristic algorithm in networks with different properties. We start evaluating the performance of our algorithm in the testbed topology we collected, then we evaluate it using randomly generated networks.

5.2.1 Testbed

We ran our heuristic in each of the different models of the testbed topology. In each experiment, we calculate the routes to every possible pair of nodes as source and destination. We make two different comparisons with our results. First, we compare the results we got from our heuristic in each network model with the optimal result in the multidigraph model. This means that we are measuring the cumulative additional cost of the model simplification and the heuristic additional cost. The results for this first set of experiments can be seen in Figure 5.4a. The additional cost in the original multidigraph model goes from 0% to about 27%, with a median value of 4%. Then the additional cost shown for the simplified models increase, because it is the cumulated additional cost from simplifying the model added to the additional cost of the heuristic over that model. For the graph model, the additional cost has a median of about 100%, which does not differ from the model additional cost previously shown in Figure 5.3.



(a) All compared with the original topology



(b) Each compared with the simplified models

Figure 5.4: Algorithm performance in the testbed topology assuming different models

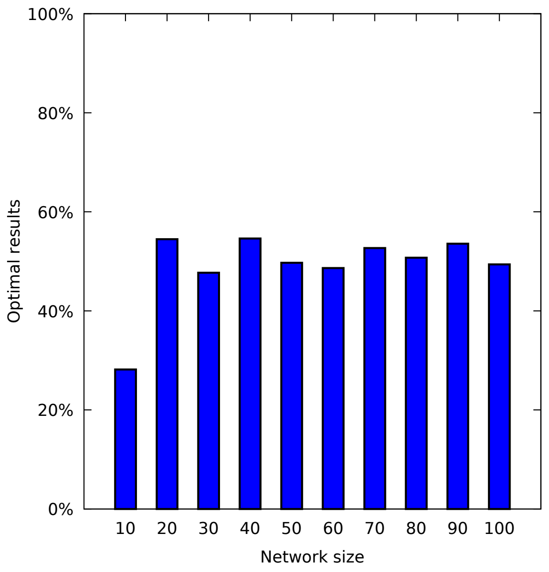
Although the cumulated additional costs are high, it is mostly because of the simplification in the models. In Figure 5.4b we see the representation of only the additional cost caused by the heuristic algorithm, i.e., we compare the result from the heuristic with the optimal result for the same simplified model, and not for the multidigraph model. The result for the first column is the same as the first column of the previous graph, since it was already compared with the optimal results for the same model. But the results for the multigraph, digraph and graph models are much smaller. In all of the three simplified models, the median line is on 0%, meaning that the heuristic achieved the optimal solution in at least 50% of the cases. We can also see that the additional costs in the digraph model were higher than in the multigraph model. This shows that the heuristic performance is better when the network is symmetric than when it is homogeneous. If a link is not symmetric, then an occupied node cannot send a message for its predecessor so that edge could be removed in the second finding phase. Finally, for the graph model, by looking at the boxplot we can see that all the quartile marks are in 0%, which means that over 75% of the cases reached the optimal value. In fact, the heuristic achieved the optimal value in 85% of the cases, and the average additional cost of those that did not reach the optimal was only 2.2%.

5.2.2 Random Multidigraphs

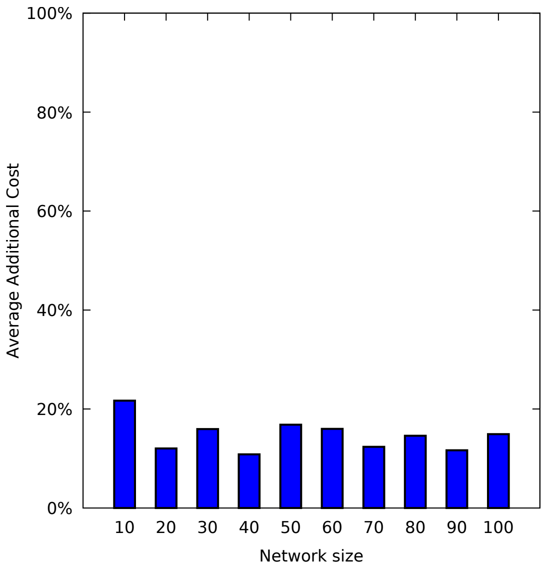
In the following set of experiments of this section, we want to see how close our heuristic comes to the optimal solution in randomly generated multidigraphs, varying in size and density. We solve each instance using the ILP model to get the optimal result, and then solve using our heuristic distributed algorithm to compare how far from the optimal are the results we obtain.

To generate random multidigraphs, we use a tool included in the suite of programs included in the package of *nauty*, a program for computing automorphism groups of graphs and digraphs provided by McKay and Piperno [2014]. It can generate random unweighted digraphs with some specified properties. For each graph generated, we specify the number of vertices and the number of edges to achieve a certain network density. We generate a multidigraph instance by generating two random digraphs and combining them in a single instance. Arcs from the first digraph are labeled with $r = 1$ and represent the links from one of the radios while arcs from the second digraph are labeled with $r = 2$ and represent links from the other radio. After generating this instance, it is still unweighted. We assign weights to the edges by generating random numbers from a exponential distribution with $\lambda = 0.05$. This distribution generates weight values less or equal to 50 with probability 91%, and less or equal to 100 with probability 99%.

Figure 5.5 shows the results we obtained by running our algorithm in randomly generated networks of different sizes in number of nodes, but fixed network density equals to 30% of the possible arcs. In the left, Figure 5.5a shows the percentage of the instances that returned paths with total weight equal to the value returned by the ILP model, reaching the optimal. In the right, Figure 5.5b shows the average additional cost obtained from the instances that did not reach the optimal value. We cannot observe a strong correlation between the size of the network and the optimality of our algorithm. Except for the very small network with only 10 nodes, which reached the optimal only about 30% of the cases and had an average additional cost of about 20%, the other instances had about the percentage of optimality and average additional cost.



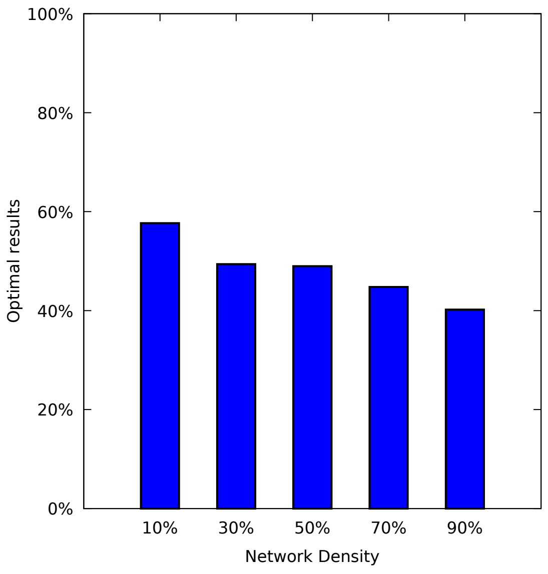
(a) Percentage of optimal results



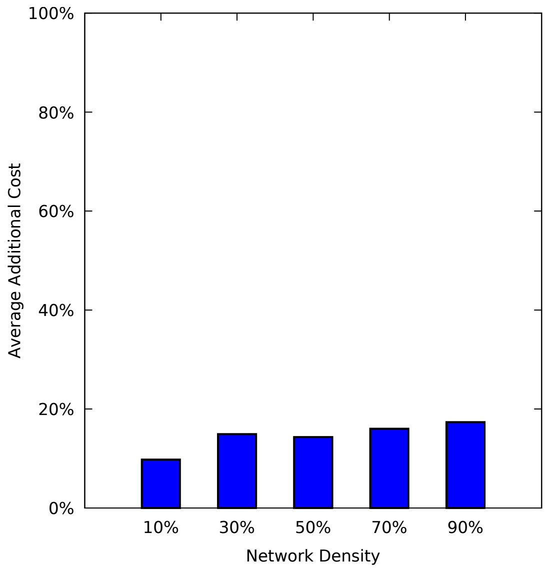
(b) Average additional cost in non optimal results

Figure 5.5: Heuristic evaluation in different network sizes and density = 30%

In Figure 5.6 we see the performance of our heuristic algorithm varying the density of the network from 10% to 90% at steps of 20%. Like the previous figure, in the left we see the percentage of the instances that reached the optimal value, and in the right we see the average additional cost of the instances that did not achieve the optimal. Unlike our analysis of the size of the network, we can see a small correlation of the optimality and the network density. With density equal to 10%, we have the best result, achieving almost 60% of the optimal results and 9% of average additional cost. The optimality and additional cost get slightly worse as the density of the network increases, having about 40% of the optimal and an average additional cost of 17% when the network have density equal to 90%. A possible explanation is that less dense networks have fewer options for disjoint paths. It is more likely that the first chosen path will be closer to the optimal solution. In more dense networks, generally there are much more than two disjoint paths between a source and a destination, and it is more likely that the fist path chosen need to be adapted in the second finding phase. But, since these instances are randomly generated multidigraphs, the chance of not having symmetric links reduce the probability of reaching the optimal. We will see in the next subsection, that having symmetric links improves the optimal results.



(a) Percentage of optimal results



(b) Average additional cost in non optimal results

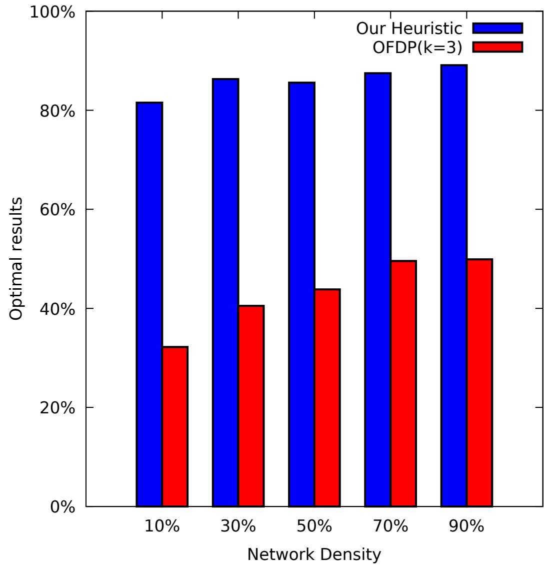
Figure 5.6: Heuristic evaluation in different network densities and size = 100

5.2.3 Random Graphs

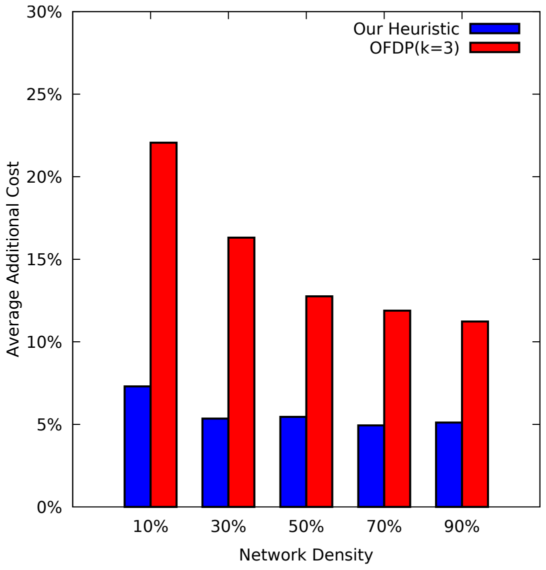
As we mentioned before, there isn't in the literature other work solving the minimum parity disjoint paths problem, so there is no proper baseline to compare our heuristic with. We decided to compare our heuristic to another simple heuristic using an existing distributed algorithm to solve the minimum k disjoint paths problem, OFDP [Zhang et al., 2016]. But since this algorithm is designed to solve the problem in undirected graphs, we can only compare the results running both algorithms in the same instance. Therefore, these experiments assume that the network is symmetric and homogeneous. To find two paths with the same parity with OFDP, we run it setting $k = 3$. The algorithm will find three disjoint paths in the network with the minimum total weight, therefore, it is guaranteed that at least two of them will have the same parity. Then we choose the pair of paths with same parity and smallest total weight. We chose this algorithm to compare our results because it is a trivial way to use an existing distributed algorithm to find paths with the same parity.

We generated five random instances of a network topology in the graph model, each one with different densities, increasing the number of edges in the graph from 10% to 90% at equal steps of 20%. In each instance we ran the ILP solution to find the optimal, and run both algorithms for every combination of two nodes as source and destination, such that source and destination are not adjacent. We excluded the case where source and destination are adjacent because, since we are assuming homogeneity, our solution will always be the trivial solution of only one edge.

In Figure 5.7a we can see the comparison of our heuristic with OFDP($k = 3$) in number of optimal solutions found. Our solution found over 80% of optimal solutions for every tested network density, slightly increasing as the density increases. With OFDP($k = 3$), the optimal solutions goes from 32% to 49%.



(a) Percentage of optimal results



(b) Average additional cost in non-optimal results

Figure 5.7: Comparing the results for our heuristic and OFDP($k = 3$) in random network topologies with different densities

In Figure 5.7b we show the average additional cost from the non-optimal solutions. Our heuristic's greatest additional cost was about 7% of the optimal in less dense networks. The smallest average additional cost with OFDP($k = 3$) was 11% in more dense networks, but reaching up to 22% at 10% density. Comparing the results of our heuristic in this scenario, where every link is symmetric with the results obtained in the last subsection, we can see again that our heuristic performance is greatly improved under the assumption of symmetric and homogeneous links. At 10% density, the optimality is improved from 60% in random multidigraphs to 80% in random graphs, and the average additional cost decreases from 9% to 7%. Another interesting observation is that the optimality increases as the density increases, which is the opposite behavior from the one we observed in multidigraphs. This indicates that, as we stated before, in more dense networks the need of changing the first path chosen is higher, and since having all the back edges allows the algorithm to always do that, the results gets better in more dense networks using the graph model.

5.3 Throughput Experiments

Finally, in this section we present practical experiments in the real-world testbed to evaluate the throughput and data yield using two paths calculated with our algorithm and compare it with with the state-of-the-art protocol for high-throughput in dual-radio WSNs, FastForward, which uses only one path.

Several rounds of experiments were performed, and each round consisted of the following steps: (i) determination of source and destination nodes, (ii) execution of the distributed routing algorithm to establish the two routes and (iii) transmission of data from the source node to the destination.

We configured the two radios to transmit using the O-QPSK modulation at 250 kbps and with transmission power of 3 dBm. In the first experiments, we left enabled, in the MAC layer, the Clear Channel Assessment (CCA) and the random back-offs. Later, these functions were disabled to compare with FastForward results. We also performed the experiments with and without acknowledgment packets, to analyze the compromise between the throughput and the data yield in both cases.

In each experiment, the source node sends 1000 packets. Each packet has 127 bytes, where the payload has 100 data bytes. We define the throughput as the total number of bytes received by the destination node per second, including those not related to the payload in the packet. We define the data yield as the number of unique packets received by the destination node divided by the total number of packets sent by the source node. The experiments were repeated 10 times for each instance, and the values presented are the mean and standard deviation of the results obtained.

The maximum transmission rate, considering an ideal environment, using only one radio is 250 kbps, or 31.25 kBps. We represent this with a green dashed line. The theoretical limit for two radios is twice or 62.5 kBps. This is represented with a continuous green line.

We present results that compare the performance of our protocol against our FastForward implementation. Figure 5.8 shows the result with CCA enabled. Top figure shows the throughput and bottom shows the data yield. We can observe that our solution using two paths achieved higher throughput. On average, we achieved a 60% throughput improvement in this scenario. The data yield of both protocols were similar, achieving 100 % data yield on most cases.

Figure 5.9 shows the performance of the protocols when we disable the CCA in the MAC layer of the radios. In practice, it is not recommended to disable this function because the media can be used by several different networks and one can congestion the other. The test was performed to evaluate the maximum transmission potential of

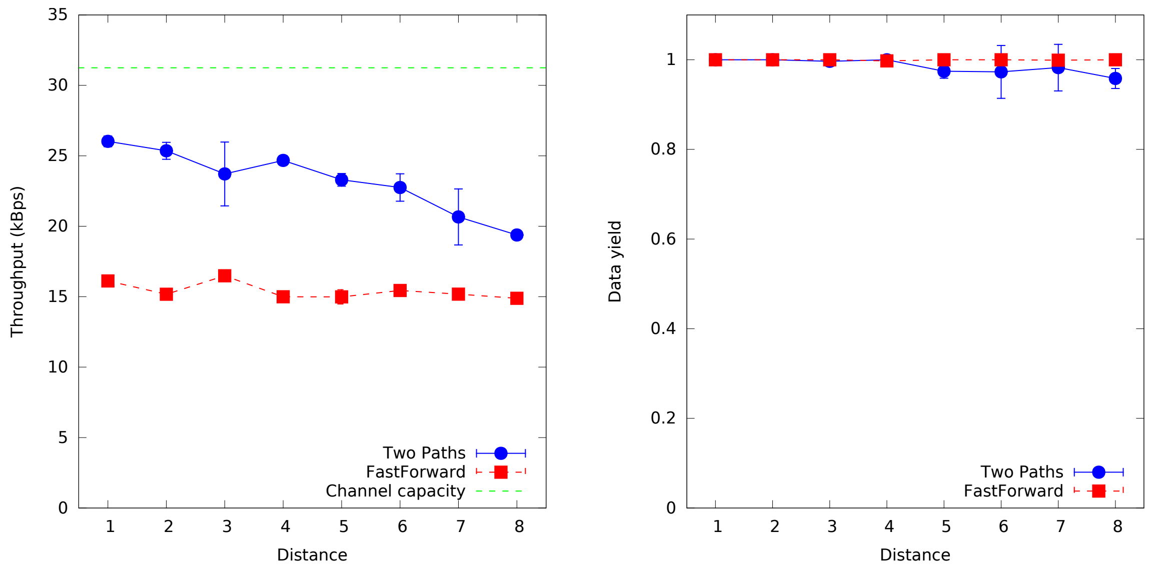


Figure 5.8: Performance when CCA is enabled and ACK is required.

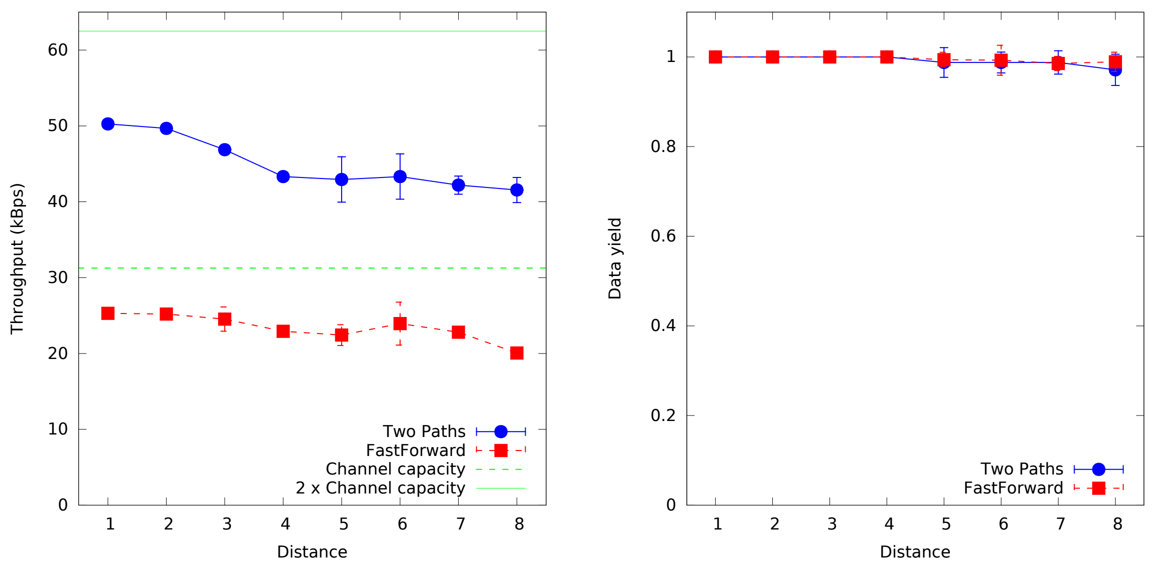


Figure 5.9: Performance when CCA is disabled and ACK is required.

the protocols. Acknowledgment packets are also used in this scenario. We can observe that in the first cases, our solution achieves 50 kBps throughput, while FastForward reaches a maximum of 25 kBps, an improvement of 100%.

Finally, Figure 5.10 represents a scenario where both the CCA and acknowledgment have been disabled. In this scenario, we obtained the maximum throughput of 60 kBps. This value is close to the maximum theoretical throughput limit when using

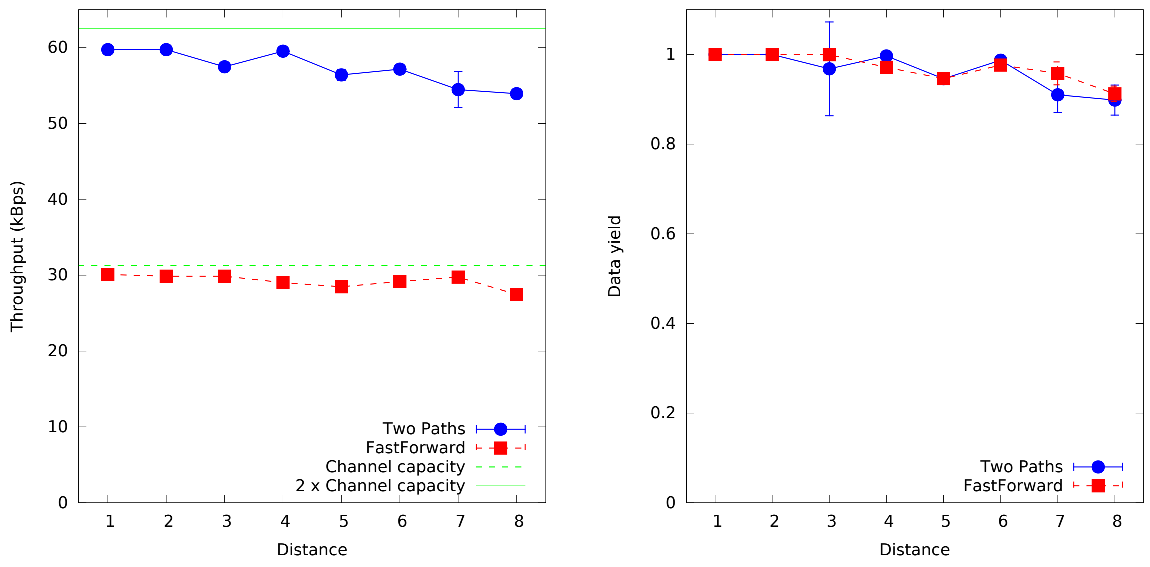


Figure 5.10: Performance when CCA is disabled and ACK is not required.

two radios. Therefore, the value obtained is 96% of ideal theoretical maximum flow and it was obtained in a real environment. In addition, this value is twice the value achieved with FastForward, indicating the maximum use of the two radios in intermediate nodes. The data rate did not reach 100% because of interference and packet error since CCA and ACK were disabled. But this scenario is important to show that our solution could reach 96% of ideal theoretical maximum flow in practice.

About energy consumption, with a larger throughput, it is expected the total energy expenditure to increase. But the energy consumption per transmitted byte becomes smaller. According to Jurdak et al. [2011], the Opal consumes an average of 49 mA of current if both radios are operating simultaneously. As our protocol reached a flow rate of 60 kBps, we have an energy expenditure of 0.82 mA/kB. While with the maximum flow reached by FastForward of 30 kBps, we would have an energy expenditure of 1.64 mA/kB. Thus, our protocol consumes less energy per byte transmitted on each node. Because the number of nodes using two paths does not double, since the source and destination nodes are always the same, our protocol will have a greater overall energy efficiency than FastForward.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this work, we presented and studied the parity disjoint paths problem, which is finding two disjoint paths between the same source and destination with the same parity. Our theoretical results show that the decision and optimization versions of the problem in directed graphs are NP-complete, and the decision version of the problem in undirected graphs is polynomial. The complexity of the optimization problem in undirected graphs remains open. We present an optimal solution for the problem in directed graphs, modeling it as integer linear programming, and a distributed heuristic algorithm.

We evaluated the proposed models for the network and the optimality and error rate for our proposed heuristic algorithm in networks with different properties. Although the heuristic have better performance in homogeneous and symmetric networks, assuming those properties in a real world testbed incur in greater errors than the error from the heuristic algorithm.

In our practical results, we presented a new protocol for bulk-data transfer optimized for dual-radio platforms, based on our heuristic for the parity disjoint paths problem. The main advantage of this new approach is to utilize the two transceivers available on each node in parallel on the source, intermediate, and destination nodes. The protocol aims to meet the demand for high throughput in new applications of wireless sensor networks that need to transmit multimedia data in real time.

Experiments carried out in the physical world show that the proposed approach achieves throughput of up to 60 kBps, which represents 96% of the theoretical maximum limit of 62.5 kBps when using two 802.15.4 radios with O-QPSK modulation at 250 kbps, without checking channel occupancy. For an application that needs to share the communication medium and checks if the channel is busy, our protocol achieves throughput of up to 26 kBps vs. 16 kBps of the current state-of-the-art FastForward protocol, a 60% performance improvement.

6.2 Future Work

There are many theoretical and practical possible future work.

- Figuring out the complexity of the minimum parity disjoint paths problem in undirected graphs. Depending on the result, find an optimal algorithm to solve the problem or a better heuristic or an approximative algorithm, if it is possible to find an approximation factor.
- Study a more general problem, of finding k disjoint paths between a source and destination with number of hops of the same result modulo k. This has a application in maximizing the throughput in a similar multi-radio wireless network scenario, where each node has k distinct radio transceivers.
- Develop a more efficient transport protocol to work with two paths, which distributes better the load between the two paths.

Bibliography

Adya, A., Bahl, P., Padhye, J., Wolman, A., and Zhou, L. (2004). A multi-radio unification protocol for ieee 802.11 wireless networks. In *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on*, pages 344--354. IEEE.

Akyildiz, I. F. and Vuran, M. C. (2010). *Wireless sensor networks*, volume 4. John Wiley & Sons.

Baek, J. W., Nam, Y. J., and Seo, D.-W. (2007). An energy-efficient k-disjoint-path routing algorithm for reliable wireless sensor networks. In *IFIP International Workshop on Software Technolgies for Embedded and Ubiquitous Systems*, pages 399--408. Springer.

Beutel, J. (2006). Fast-prototyping using the btnode platform. In *Design, Automation and Test in Europe, 2006. DATE'06. Proceedings*, volume 1, pages 1--6. IEEE.

Bhandari, R. (1997). Optimal physical diversity algorithms and survivable networks. In *Computers and Communications, 1997. Proceedings., Second IEEE Symposium on*, pages 433--441. IEEE.

Burns, A., Greene, B. R., McGrath, M. J., O'Shea, T. J., Kuris, B., Ayer, S. M., Stroiescu, F., and Cionca, V. (2010). Shimmer^{TM}–a wireless sensor platform for noninvasive biomedical research. *Sensors Journal, IEEE*, 10(9):1527--1534.

Chen, Y.-q., Guo, X.-f., Zeng, Q.-k., and CHEN, G.-h. (2004). Amr: a multipath routing algorithm based on maximum flow in ad-hoc networks. *Acta Electronica Sinica*, 32(8):1297--1301.

Draves, R., Padhye, J., and Zill, B. (2004). Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 114--128. ACM.

Duquennoy, S., Österlind, F., and Dunkels, A. (2011). Lossy links, low power, high throughput. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 12--25. ACM.

Durmus, Y., Ozgovde, A., and Ersoy, C. (2012). Distributed and online fair resource management in video surveillance sensor networks. *IEEE Transactions on Mobile Computing*, 11(5):835--848.

Ekbatanifard, G., Sommer, P., Kusy, B., Iyer, V., and Langendoen, K. (2013). Fastforward: High-throughput dual-radio streaming. In *Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on*, pages 209--213. IEEE.

Fang, X., Shi, S., and Li, J. (2009). A disjoint multi-path routing algorithm in wireless sensor network. *Journal of Computer Research and Development*, 46(12):2053--2061.

Ganesan, D., Govindan, R., Shenker, S., and Estrin, D. (2001). Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(4):11--25.

Griffin, M. and Korkmaz, T. (2011). Distributed verification of global multiple disjoint paths in mobile wireless networks. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 666--670. IEEE.

Gürses, E. and Akan, Ö. B. (2005). Multimedia communication in wireless sensor networks. In *Annales des Télécommunications*, volume 60, pages 872--900. Springer.

Hashiguchi, T., Tajima, K., Takita, Y., and Naito, T. (2011). Node-disjoint paths search in wdm networks with asymmetric nodes. In *Optical Network Design and Modeling (ONDM), 2011 15th International Conference on*, pages 1--6. IEEE.

Holman, R., Stanley, J., and Ozkan-Haller, T. (2003). Applying video sensor networks to nearshore environment monitoring. *IEEE Pervasive Computing*, 2(4):14--21.

Hu, F. and Kumar, S. (2003). Qos considerations in wireless sensor networks for telemedicine. In *ITCom 2003*, pages 217--227. International Society for Optics and Photonics.

Ishida, K., Kakuda, Y., and Kikuno, T. (1995). A routing protocol for finding two node-disjoint paths in computer networks. In *Network Protocols, 1995. Proceedings., 1995 International Conference on*, pages 340--347. IEEE.

Iwama, K., Iwamoto, C., and Ohsawaa, T. (1997). A faster parallel algorithm for k-connectivity. *Information processing letters*, 61(5):265--269.

Jurdak, R., Klues, K., Kusy, B., Richter, C., Langendoen, K., and Brünig, M. (2011). Opal: A multiradio platform for high throughput wireless sensor networks. *Embedded Systems Letters, IEEE*, 3(4):121--124.

Kawarabayashi, K.-i., Reed, B., and Wollan, P. (2011). The graph minor algorithm with parity conditions. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 27--36. IEEE.

Khuller, S. and Schieber, B. (1991). Efficient parallel algorithms for testing k and finding disjoint s-t paths in graphs. *SIAM Journal on Computing*, 20(2):352--375.

Kim, S., Fonseca, R., Dutta, P., Tavakoli, A., Culler, D., Levis, P., Shenker, S., and Stoica, I. (2007). Flush: a reliable bulk transport protocol for multihop wireless networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 351--365. ACM.

Kohvakka, M., Arpinen, T., Hännikäinen, M., and Hämäläinen, T. D. (2006). High-performance multi-radio wsn platform. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality*, pages 95--97. ACM.

Kumar, A. and Varma, S. (2010). Geographic node-disjoint path routing for wireless sensor networks. *IEEE Sensors Journal*, 10(6):1138--1139.

Kusy, B., Richter, C., Hu, W., Afanasyev, M., Jurdak, R., Brünig, M., Abbott, D., Huynh, C., and Ostry, D. (2011). Radio diversity for reliable communication in wsns. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 270--281. IEEE.

LaPaugh, A. S. and Papadimitriou, C. H. (1984). The even-path problem for graphs and digraphs. *Networks*, 14(4):507--513.

Lee, Y. O. and Reddy, A. N. (2010). Disjoint multi-path routing and failure recovery. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1--6. IEEE.

Li, Q., Han, D., Gnawali, O., Sommer, P., and Kusy, B. (2013). Twonet: Large-scale wireless sensor network testbed with dual-radio nodes. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 89. ACM.

Li, S. and Wu, Z. (2005). Node-disjoint parallel multi-path routing in wireless sensor networks. In *Embedded Software and Systems, 2005. Second International Conference on*, pages 6--pp. IEEE.

McKay, B. D. and Piperno, A. (2014). Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112. ISSN 0747-7171.

Misra, S., Reisslein, M., and Xue, G. (2008). A survey of multimedia streaming in wireless sensor networks. *IEEE communications surveys & tutorials*, 10(4):18--39.

Ogier, R. and Shacham, N. (1989). A distributed algorithm for finding shortest pairs of disjoint paths. In *INFOCOM'89. Proceedings of the Eighth Annual Joint Conference of the IEEE Computer and Communications Societies. Technology: Emerging or Converging, IEEE*, pages 173--182. IEEE.

Omar, S., Zoulikha, M., and Cousin, B. (2011). Energy efficiency in ad hoc wireless networks with node-disjoint path routing. In *Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on*, pages 127--130. IEEE.

Popovici, E., Boyle, D., O'Connell, S., Faul, S., Angove, P., Buckley, J., O'Flynn, B., Barton, J., and O'Mathúna, C. (2011). The s-mote: A versatile heterogeneous multi-radio platform for wireless sensor networks applications. In *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pages 421--424. IEEE.

Rahimi, M., Baer, R., Iroezi, O. I., Garcia, J. C., Warrior, J., Estrin, D., and Srivastava, M. (2005). Cyclops: in situ image sensing and interpretation in wireless sensor networks. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 192--204. ACM.

Raman, B., Chebrolu, K., Bijwe, S., and Gabale, V. (2010). Pip: A connection-oriented, multi-hop, multi-channel tdma-based mac for high throughput bulk transfer. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 15--28. ACM.

Shin, B., Ko, Y., An, J., and Lee, D. (2009). Interference-aware routing protocol in multi-radio wireless mesh networks. In *Proceedings of the 4th international conference on future internet technologies*, pages 12--15. ACM.

Sidhu, D., Nair, R., and Abdallah, S. (1991). Finding disjoint paths in networks. In *ACM SIGCOMM Computer Communication Review*, volume 21, pages 43--51. ACM.

Srinivas, A. and Modiano, E. (2003). Minimum energy disjoint path routing in wireless ad-hoc networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 122--133. ACM.

Subramanian, A. P., Buddhikot, M. M., and Miller, S. (2006). Interference aware routing in multi-radio wireless mesh networks. In *Wireless Mesh Networks, 2006. WiMesh 2006. 2nd IEEE Workshop on*, pages 55--63. IEEE.

Suurballe, J. (1974). Disjoint paths in a network. *Networks*, 4(2):125--145.

Tarjan, R. E. (1983). *Data structures and network algorithms*. SIAM.

Tavares, R., Vieira, M. A. M., and Vieira, L. F. M. (2016). Flushmf: A transport protocol using multiple frequencies for wireless sensor network. In *Proceedings of the 13th IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS)*. IEEE.

Teixeira, T., Culurciello, E., Park, J. H., Lymberopoulos, D., Barton-Sweeney, A., and Savvides, A. (2006). Address-event imagers for sensor networks: evaluation and modeling. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 458--466. ACM.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Computer networks*, 52(12):2292--2330.

Zhang, K., Han, Q., Yin, G., and Pan, H. (2016). Ofdp: a distributed algorithm for finding disjoint paths with minimum total length in wireless sensor networks. *Journal of Combinatorial Optimization*, 31(4):1623--1641.

Zhang, Y., Wetherill, B. R., Chen, R. F., Peri, F., Rosen, P., and Little, T. D. (2014). Design and implementation of a wireless video camera network for coastal erosion monitoring. *Ecological Informatics*, 23:98--106.