A link-quality-aware graph model for cognitive radio network routing topology management

Andrew Michael James

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
A Link-quality-aware Graph Model for Cognitive Radio Network Routing Topology Management

by

Andrew Michael James

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Assistant Professor, Dept. of Computer Engineering Dr. Fei Hu
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
October 2007

Approved By:

Dr. Fei Hu
Assistant Professor, Dept. of Computer Engineering
Primary Adviser

Dr. Muhammad Shaaban
Associate Professor, Dept. of Computer Engineering

Dr. Roy Melton
Visiting Assistant Professor, Dept. of Computer Engineering
Title: A Link-quality-aware Graph Model for Cognitive Radio Network Routing Topology Management

I, Andrew Michael James, hereby grant permission to the Wallace Memorial Library reproduce my thesis in whole or part.

Andrew Michael James

Date
Dedication

This research is dedicated to all of my family, friends, and professors who supported me in my academic journey.
Acknowledgments

Thank you Dr. Hu for all of your guidance throughout this research. I would also like to thank my committee members, Dr. Shaaban and Dr. Melton for their assistance in this process.
Abstract

Wireless communications is one of the fastest growing fields in the world; however this creates a problem since all wireless signals are fighting for the same limited amount spectrum in any given space. The underutilization of licensed spectrum has created a need for a new way to use it. Cognitive Radio Networks and Dynamic Spectrum Access are a solution to this problem.

By opportunistically using spectrum, devices can gain access to more wireless bandwidth while not violating FCC regulations. The concepts of Cognitive Radio Networks and Dynamic Spectrum Access are very new topics and have yet to be fully explored. One of the current goals in this area is adapting existing concepts in networking algorithms to be aware of and to take advantage of a Dynamic Spectrum Access environment.

Awareness and using cross-layer design enables opportunistic use of the spectrum and allows devices to take full advantage of the nature of the Dynamic Spectrum Access environment. This thesis explores some existing solutions to the Dynamic Spectrum Access problem, and uses them as inspiration to create a Link-quality-aware Graph Model for Cognitive Radio Network Routing Topology Management.
# Contents

Dedication ................................................................. iii

Acknowledgments ......................................................... iv

Abstract ........................................................................ v

Glossary ......................................................................... x

1 Introduction .................................................................. 1
   1.1 Dynamic Spectrum Access and Cognitive Radio Networks .......... 1
   1.2 Interference Temperature .................................................... 4
   1.3 Cross-layer Design .......................................................... 5
   1.4 Channel Interference ........................................................ 6
   1.5 Routing Styles ............................................................... 7
      1.5.1 Proactive vs. Reactive .................................................. 7
      1.5.2 Centralized vs. Decentralized ....................................... 8
   1.6 Motivation ................................................................. 9
   1.7 Document Structure ..................................................... 10

2 Related Work ................................................................ 11
   2.1 Genetic Algorithms ........................................................ 11
   2.2 Layered Graph Model ..................................................... 15
   2.3 Spectrum Aware On-demand Routing Protocol ....................... 20

3 A Link-quality-aware Graph Model for Routing Topology Management .. 22
   3.1 Algorithm ..................................................................... 22
      3.1.1 Prioritization ............................................................ 22
      3.1.2 Channel Selection .................................................... 23
      3.1.3 Routing ................................................................. 24
      3.1.4 Fitness Function ..................................................... 24
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Spectrum Usage [1]</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>IT Model [1]</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>An Example of a Small Scale of the Layered Graph [14]</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>An Example Network with Routing and Channel Selection</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>Throughput versus Load with a 15 Node Network [14]</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Throughput versus Load with a 30 Node Network [14]</td>
<td>34</td>
</tr>
<tr>
<td>4.3</td>
<td>SORP Results for Increasing SSD [2]</td>
<td>35</td>
</tr>
<tr>
<td>4.4</td>
<td>SORP Results for Increasing Number of Flows [2]</td>
<td>36</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison of Run Times</td>
<td>39</td>
</tr>
<tr>
<td>4.6</td>
<td>Comparison of Avoidance of Channel Interference</td>
<td>40</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison of Average Number of Hops</td>
<td>40</td>
</tr>
<tr>
<td>4.8</td>
<td>The Problem with SeqAssign</td>
<td>44</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Run Time Results</td>
<td>42</td>
</tr>
<tr>
<td>4.2</td>
<td>Diversity Results</td>
<td>42</td>
</tr>
<tr>
<td>4.3</td>
<td>Path Length Results</td>
<td>42</td>
</tr>
</tbody>
</table>
Glossary

A

AODV  Ad-hoc On-demand Distance Vector.

B

BER  Bit Error Rate.

C

CCC  Common Control Channel.

CR  Cognitive Radio.

D

DSA  Dynamic Spectrum Access.

F

FCC  Federal Communications Commission.

G

GA  Genetic Algorithms.
I
IT  Interference Temperature.

L
LGM  Layered Graph Model.
LGMRTM  Link-aware Graph Model for Routing Topology Management.

M
MAC  Medium Access Control.
MANET  Mobile Ad Hoc Network.

P
PU  Primary User.

Q
QoS  Quality of Service.

R
RREP  Route Reply.
RREQ  Route Request.

S
SNR  Signal to Noise Ratio.
SOP  Spectrum Opportunities.
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD</td>
<td>Sparsity of Spectrum Distribution.</td>
</tr>
<tr>
<td>SU</td>
<td>Secondary User.</td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Dynamic Spectrum Access and Cognitive Radio Networks

The area of wireless communications has become huge and continues to grow at a fast pace. Wireless devices are found everywhere and are constantly in use. This mass market for wireless communications has created a problem. Unlike wired devices where the signal is contained in the space that the wire consumes, wireless signals send their signal spreading to the entire surrounding area propagating through the air. All devices within reach of each other have to compete for the finite amount of spectrum that exists because of the fact that every wireless signal in a given area has to travel through the same airspace.

Complicating the issue is that the Federal Communications Commission (FCC) reserves much of this spectrum to auction to companies that provide wireless services. Mobile phone service providers would be an example of a company bidding on this spectrum. When a company licenses a piece of spectrum from the FCC, it has exclusive access to that band, and any unauthorized use of that spectrum that interferes with its primary user violates FCC restrictions. In order for companies to guarantee good service for their customers, the companies typically own the license to more spectrum than is needed on a regular basis. This is beneficial to them for a couple of reasons. First, if their customer base expands,
which any business is hoping for, extra spectrum is already licensed and ready to dedicate to that growing customer base. The other reason is that this can allow the company to provide a higher quality of service. If more than enough spectrum is possessed, then their customers are rarely pushing it to its limit. When customers saturate the spectrum there can be conflicts in serving all users. If the amount of licensed spectrum is always more than sufficient, then these issues are mitigated. A study done by the FCC showed that on average around 50% of the licensed spectrum is unused at any given time or place [1]. This means that while spectrum that is available for unlicensed use has become cluttered with unlicensed devices, there is a large amount of licensed spectrum being underutilized. The problem is that unlicensed devices cannot interfere with licensed devices. The issue of licensed spectrum being underutilized is illustrated in Figure 1.1.

![Figure 1.1: Spectrum Usage](image)

Figure 1.1: Spectrum Usage [1]
A user that is affiliated with the business that owns a licensed piece of spectrum is called a licensed or primary user. A user that is not affiliated with the company who licensed the spectrum being used is called an unlicensed or secondary user. Once the users were divided into two groups, a new concept emerged. The unlicensed devices merely cannot interfere or disrupt licensed communications. If unlicensed devices were created that could take advantage of the unused spectrum without interfering with licensed users then that extra spectrum could be used without violating FCC mandates. This concept is called Dynamic Spectrum Access (DSA). DSA works by detecting unused pieces of spectrum, which are called Spectrum Opportunities (SOP). Once the device knows what spectrum is available it can decide how to use that spectrum. Since the available spectrum is not guaranteed, the device must continuously monitor what spectrum is available. The device must also detect when a piece of spectrum it is using is needed by a primary user, since the prerequisite for using the spectrum is that it must be vacated when it is going to be used by a primary user. The reason it is called dynamic spectrum access is because the spectrum used for communications is changing in a dynamic fashion as the wireless environment around the device changes. This concept raised many technical issues and questions.

In order to make DSA work, the devices require another emerging technology called the Cognitive Radio (CR). A standard radio has a limited range of frequencies it can tune to, which is not well suited for a DSA environment. A radio for communications using DSA needs to be flexible and to allow tuning to many different bands. A cognitive radio is a highly flexible radio that can tune to a very wide range of frequencies to allow transmitting and receiving on a great variety of channels.

It is important to have access to a wide range of frequencies in a DSA network. If the device must change channels when primary (licensed) users arrive, then the radio must be able to jump to a variety of channels to avoid interference with primary users while still being able to find open channels to allow it to do its own communicating.
Another technical challenge for the CR is that it needs to be able to change frequencies quickly. One of the big concerns in writing networking protocols for DSA environments has been trying to utilize a large amount of the available spectrum while also not forcing the radio interface to have to change channels too frequently. These radio interfaces require a non-negligible amount of time to change channels. One of the factors for how much time it takes is how big the frequency gap is between the channels. For example, shifting the frequency band by 2 GHz takes more time than shifting it 2 MHz.

### 1.2 Interference Temperature

An important aspect of the lower networking levels in a DSA network is a new concept called Interference Temperature (IT). While nobody has set in stone how to measure interference temperature as of yet, the general ideas behind it are fairly established. IT should measure the amount of interference a device is creating for other devices in its range. Once IT can be measured, limits on IT can then be set so that DSA networks can use these measurements to decide how it can utilize the spectrum without interfering with primary users [1].

Kolodzy [5] defines interference temperature as “a measure of the RF power available at a receiving antenna to be delivered to a receiver-power generated by other emitters and noise sources. More specifically, it is the temperature equivalent of the RF power available at a receiving antenna per unit bandwidth, measured in Kelvin.” Thus the IT is a measure of how much signal power exists surrounding the receiver. If this level gets too high the receiver will not be able to distinguish the signal it intends to receive from the other signals in the air at its location. This is why there must be limits to IT.

Figure 1.2 illustrates how IT can work. The bold curve shows the received power at a primary user as its range increases from the source. The bottom bar shows the noise floor that exists without any DSA usage of the spectrum. The horizontal line above the jagged
section is the limit to IT; any unlicensed signal that is transmitting at a power above that line is violating FCC regulations. The light areas between the noise floor and IT limit are spots where there are opportunities to access spectrum since the power can be kept below IT requirements, but the signal can still be strong enough to overcome the noise floor. This is where DSA lives. The goal of DSA is to find these holes and exploit them to increase overall spectrum utilization.

1.3 Cross-layer Design

In the design of networking protocols there is a concept called cross-layer design. The concept is that information about what is occurring at lower layers is passed up and is available to the higher layers. The higher layers can then use this information to make better decisions in accomplishing the task at that layer as well as being able to send lower layers commands to make the lower layers run better as well.

The reason this becomes so essential is because there are more choices to make in a DSA environment. Most of the time, the best way to select the best option is to know what is going on in a different layer, and to have multiple layers working together to find the best
comprehensive solution. This cooperation is what makes cross-layer design so powerful.

The way cross-layer design is used in this thesis is a merging of the Medium Access Control (MAC) layer and the Routing layer. By selecting channels and routes in the same algorithm, the two layers end up working together to accomplish better overall Quality of Service (QoS). In addition to the simultaneous channel and route selection, the proposed algorithm also uses information provided by the physical layer to know which channels are available and some statistics about those channels.

1.4 Channel Interference

Channel interference is something that is impossible to avoid in a typical wireless network where all of the nodes are competing for the same frequency band. One of the benefits of DSA is that it allows for the network to exist on a variety of channels such that it is in theory possible to never need to enter a “back-off” state. A common part of the MAC layer for traditional wireless networks is the idea of either avoiding or detecting collisions and then dealing with them by backing off. A collision occurs in a wireless network when more than one device transmits on the same channel within a proximity that makes it such that receivers cannot distinguish between the signals. As an analogy, if a person were talking to someone and a third person starts talking at the same time; the first person now cannot tell what the second person just said because the third person’s message interfered. Now the second person must retransmit his message. If person two and person three both retransmit their messages at the same time again then both have still failed to have a successful communication. Therefore one of them must back off and let the other speak first.

What if, instead, the scenario is such that person one and person two are on the other side of a wall from person three, but they are all within the same proximity? Now when both number two and three speak at the same time, number one can hear just fine because the wall provides separation. The wall separation is like being on a different channel. If
person one and person two are tuned to a different channel than person three, then the third person can speak all he wants without interfering with the conversation between one and two. When the wall is not there and all three are on the same channel, there was channel interference, but by diversifying the used channels that channel interference fades. This allows both two and three to communicate without having to wait for each other, and therefore the QoS increases. The proposed solution, as well as the comparative solutions, uses a decrease in channel interference to help increase the QoS.

1.5 Routing Styles

1.5.1 Proactive vs. Reactive

This section is about how routing protocols are categorized. Every routing protocol is either proactive, reactive, or a hybrid of the two extremes. A reactive routing protocol is one that figures out the route to take when a route is needed. The route is discovered when needed and then discarded. A proactive routing protocol figures out the routes in advance and keeps a routing table to store information on which route to take for each destination. The proactive protocol will update this table as needed.

It is clear that these two take different approaches. The proactive style plans ahead so that as soon as something needs to be sent, it can go right away. This can lower the delay between the time the message is ready to be sent and when it is received since the node does not have to wait to figure out where to send the packet. The reactive style puts off doing the routing work until it is needed, which can introduce delay, but since the path is thrown away after it is been used there is no storage overhead. The routing table in a proactive protocol could potentially get massive in a large wireless network. There is no obvious choice for all wireless routing protocols; it all depends on the network in which the protocol is going to be operating. In a Wireless Sensor Network (WSN), there might be too many nodes and not enough resources in the node to be able to make a proactive protocol.
feasible, but in a Mobile Ad Hoc Network (MANET) of laptops the resources might be available and that decreased delay could be more important.

1.5.2 Centralized vs. Decentralized

The other common way to categorize wireless network routing protocols is by whether they are centralized or decentralized. A centralized routing protocol lets a central server or designated node figure out all of the routing for all of the nodes in a network and then pass the routing information that each node will need to the nodes. A decentralized routing protocol has each node calculating its own routing information and figuring out the paths itself.

When deciding whether to make a protocol centralized or decentralized, knowing the application and the situation in which it will be used is critical. If there is a situation where a central base of operation exists, and it has more computational power than the nodes, then a centralized protocol would make sense. An advantage of the centralized protocol is that when making routing decisions all nodes can be considered. This can allow the protocol to make a decision that may hinder one node a little bit, but enhance several others, which improves the overall topology. In a decentralized protocol, the node only knows information about its own situation and cannot take into account what else is going on in the network. Each node is thinking only of itself, which means there can be situations where this inherently greedy approach could create a choke point where every node wants to travel through one convenient area, but all of that traffic congests the network. A centralized protocol has the ability to alleviate that congestion by load balancing the network. What a decentralized protocol can do is distribute the work that needs to be done to establish the routing topology which can potentially get this task accomplished faster. In a centralized protocol, the entity doing the work must wait to receive all of the relevant information, process the data, and then send out the routing information. In a decentralized scheme each node is only solving its own routing problem, which in general should require
less work than finding the entire topology.

1.6 Motivation

This section describes the motivation behind this research. At the moment the intended use of DSA is to enhance MANETs. By using the extra spectrum that the primary users are not using, a MANET can have more bandwidth and network choices. The other advantage of applying the DSA environment to a MANET is that it makes the MANET able to be established anywhere. If using DSA, a MANET can be established in a territory where the operators do not know anything about how the spectrum is allocated or controlled. With DSA they do not need to know because it enables the network to use spectrum without interfering with pre-established wireless networks of any sort. One example of this is if the military wants to setup a MANET in a foreign country where they have no intelligence on the radio environment. They can now establish their network without wondering if the channels their equipment is set to use are already in use. With DSA the network finds the free channels and uses those.

This raises the question of why new protocols at so many levels are needed. Why not use an existing routing or MAC protocol that is designed to work with MANETs do the job, and let the CR figure out on what channel to broadcast? The answer is that it is quite possible to use some of the existing protocols with perhaps changes only in a few spots and leaving most of the work to the hardware. The issue with doing this is that it does not maximize the potential of the network. Why settle for mediocre performance when one of the uses of this technology is to increase the amount of spectrum the network can utilize? By making new protocols that are designed to be aware of the DSA environment and take advantage of that fact, better performance should be achieved. Once research started in this area, it became rapidly apparent that the way to squeeze out this extra performance was to use cross-layer design to allow communication and cooperation between layers since by definition the DSA environment is more complex than a standard wireless environment.
1.7 Document Structure

The rest of the document is organized as follows. The next chapter covers work related to this one. The first piece of related work describes the examination of the use of genetic algorithms (GA) for this work. In addition, two specific papers, and their ideas, will be discussed. The first is about a layered graph model for routing in a DSA environment. The second paper describes a spectrum aware on-demand routing protocol which also is tailored to a DSA environment.

The third chapter describes the proposed solution of this document, which is a Link-quality-aware Graph Model for Routing Topology Management. First the algorithm is discussed, followed by a description of the simulation tool that was used to test the proposed solution as well as the compare the proposed solution to the layered graph model.

The fourth chapter analyses the results obtained from the simulations run for this research, and compares those results to the results of the layered graph model and spectrum aware on-demand routing protocol papers. In addition, there is some analysis of the results from the aforementioned works. The fifth chapter draws conclusions from this research and offers ideas for how this work could be extended in the future.
Chapter 2

Related Work

This chapter describes the work related to this research. The first section describes the exploration of genetic algorithms as a potential solution to the routing problem in DSA networks. The two sections after that give an overview of the two works most inspirational to this one. The first work was a layered graph model for routing, and the second is a spectrum aware on-demand routing protocol.

2.1 Genetic Algorithms

Genetic Algorithms (GA) were initially considered in this research, but the approach did not prove to offer a good solution; however, the concept did inspire part of the proposed solution, as explained later in this work. The concept of GA is to use a biological model for solving problems. It uses the ideas of reproduction and evolution to find an answer to a problem without searching through every possible answer [11]. This can be advantageous in situations where the solution set is infinite or very large. Solutions are represented as genes in GA, where each gene has several chromosomes that represent the different pieces of the possible solution. If the physical layer were being analyzed, factors such as frequency, modulation scheme, and power would be some of the possible chromosomes.

There is a cycle of phases that take place in GA and they are as follows. Before the cycle starts there is an initialization phase where the initial gene pool is created. The
selection stage chooses the strongest genes, and those will survive to see another cycle. Next, a reproductive phase is encountered which typically has two sub-phases as follows. In crossover, the strongest genes are crossed together to make new unique genes in the hopes that the crossover will randomly take the strong parts of one gene and mix it with the strong parts of another gene to make an even stronger new gene. The mutation phase takes the gene pool and randomly alters some of the chromosomes to simulate mutations that occur in the biological world. The idea here is that this mutation could randomly introduce a very strong chromosome into the mix to help create a stronger gene. At this point, the algorithm cycles back around to the selection phase to select the strongest genes to continue, and the cycle repeats. Eventually there will come a point where there will be a gene determined to be strong enough to be considered the answer.

The key in genetic algorithms is the fitness function. This function is used to evaluate how good an answer is. The function can be a simple equation where the gene is a set of coefficients, and the evaluation of the function gives an answer. The closer that answer is to the one being sought out then the better the gene. It could also be a complex function that does a lot of work behind the scenes to evaluate the gene and give back a rating based on certain parameters. Once the genetic algorithm exists, only a new fitness function is needed to make it solve a new problem as long as the problem is represented numerically. The process of homing in on an answer can be fast or slow depending on how complex the fitness function is and how accurate the answer must be. Having a good initial set of genes can help too.

In exploring the possibility of using GA as the basis for the proposed solution, a really rough physical layer simulation was created using Matlab (which was used for all coding done in this research). The simulation was driven by two timers. One timer was to trigger a change in the wireless environment. This was to simulate the evolving nature of the DSA environment. The other timer was to trigger the start of the search for the best physical layer parameters to use in communication. The fitness function evaluated the proposed
solutions by examining how well they fit the environment. The environment parameters were the largest piece of spectrum (defined by a lower and upper frequency bound), the Bit Error Rate (BER) of recent transmissions, the Signal to Noise Ratio (SNR) of recent transmissions, and Interference Temperature (IT). In addition to having these rough parameters, there was also a goal variable. The goal was an array that represented how important three different factors were in the selection of the parameters. The values represented the importance of minimizing power, maximizing data rate, and minimizing BER. The relative values of these numbers to each other created a goal. If the minimal power goal had a high value and the other two goals parameters were low, then the fitness function is going to favor solutions that propose solutions that suggest transmitting on a low power level. The factors are not strictly environmental; they represent statistics of concern in the physical layer.

The genes consisted of the following four chromosomes: power, carrier frequency, bandwidth, and modulation scheme. The genetic algorithm passes in suggestions for each of these, and based on the suggestions’ compatibility with the current environment (including the current goal) a fitness value is assigned. The GA tool that exists in Matlab allows the user to specify ranges that are suitable for each parameter, which is useful because writing a fitness function is easier when there are known constraints on the input. The process stops when the GA tool has converged on a best fitness rating or when the cycle has been completed a maximum number of times specified by the user. At this point the GA returns the best gene, and this outcome is shown to the user.

The simulation created is simplistic and was made to get a feel for how GA could work with a problem involving DSA. The approach was decidedly unsuitable. The following example will illustrate why it was not feasible. There were five possible modulation schemes so the input was from 0 to 4, and, since a non-integer value was given, the input was rounded to the nearest integer to interpret which modulation was suggested. This means that the fitness function is not directly using the value given by the GA tool, which means
the GA tool is not getting ideal feedback. A small tweak to the modulation chromosome provides no new information because it can be interpreted as the same gene. It leads to the conclusion that when a finite number of possibilities exist, GA may be slower than just searching the finite number of possibilities.

The goal for the proposed solution is to combine channel selection and route selection into a single algorithm. There are a finite number of possible solutions. While a large network with a large amount of spectrum available may make that set larger, it cannot justify the process of creating this abstraction necessary to fit it into a GA problem. For each selection there will be a different channel set available. The GA tool would always have the possibility of making suggestions that are not possible, because it may suggest a channel that is being used. The other problem is that the gene would only be a single chromosome which would represent the channel. The key here is that this problem does not naturally fit into the GA paradigm; it needs to be forced into it, which makes no sense from the perspective of doing research to solve a new problem. There is no reason to transform this problem into one that can be solved with GA when it can be solved without GA.

There is something valuable to take away from the GA concept. While fitting this problem into a biological model does not work well, the key to GA, the fitness function, is a quite useful tool. At some point the available channels and routes need to be evaluated to decide which is best, so a fitness function can be useful. Instead of evaluating random inputs given by the GA tool, instead it is better to evaluate each possibility and have the best one win. The finite nature of this problem makes GA a bad choice, but from the experience gained in applying it to the physical layer, it could be useful for MAC and routing layer choices, since the genes would have more chromosomes. This also adds flexibility to the proposed solution where evaluating the channels based on more information just requires a simple update to the fitness function. Of course there would also need to be changes to make the additional information available to the fitness function. When the proposed
solution is presented, the fitness function used will be explained.

2.2 Layered Graph Model

Some of the core ideas of the proposed solution came from Xin[14]. In this paper the authors present a Layered Graph Model (LGM) for combining channel selection with route selection. Essence the goal was to create an elaborate graph that could represent all of the possible connections between all of the nodes, and assign each connection a weight, and then simply use a shortest path algorithm to find the shortest paths. While this is a great idea, it has its issues, which will be explained later.

The first part is to build the graph. As with all graphs there are edges and vertices. In the LGM there are several types of vertices and several types of edges which represent different connections. The vertex types will be explained first. There are main vertices, sub-vertices, and auxiliary vertices. The main vertices represent the node, and for each node there is one main vertex. The sub-vertices and auxiliary vertices come in pairs. For every channel to which a node has access, there is a sub-vertex and an auxiliary vertex. There are four kinds of edges. All edges are unidirectional. The first kind is the horizontal edge which represents a connection over a certain channel from one wireless node to another. This edge will exist if both nodes are free to use that channel. The different layers of the graph represent these different channels that exist in the DSA environment. Vertical edges represent a change in channel at a node. If a message comes in on channel A, but then is sent out to the next node via channel B it would have taken a vertical edge in the graph from the channel A layer to the channel B layer. The last two types are the access edge, and the internal edge. Access edges connect the main node to the sub-vertices and auxiliary vertices, and internal edges connect the sub-vertices to the auxiliary vertices. Figure 2.1 is from the LGM paper and can help with visualizing the graph. The reason for sub-vertices and auxiliary vertices is so that they can penalize receiving and sending on the same channel. Ideally they want to encourage the algorithm to use a diverse set of channels in the path because that will
reduce interference. The unidirectional edges allow for common shortest path algorithms to be used, since many require unidirectional edges.

Horizontal edges originate at auxiliary vertices, and terminate at sub-vertices. Vertical edges originate at sub-vertices and terminate at auxiliary vertices. Access edges either originate at a main vertex, in which case it terminates at an auxiliary vertex, or terminate at the main node, in which case it originated at a sub-vertex. Internal edges originate at sub-vertices and terminate at auxiliary vertices. Each edge type has a different weight, where access edges cost 1000, vertical edges cost -10, internal edges cost 40, and horizontal edges
have an evolving weight that starts at 10. The horizontal edge cost will be explained more in depth later. LGM considers the number of radio interfaces available to it. It assumes there is at least one flexible radio interface, such as a CR. When an interface is assigned to a certain channel, the sub and auxiliary vertices, corresponding to the channel being tuned on the node containing the interface, are now considered active. To be clear, an active vertex has a radio interface (in this case we assume this radio interface is a CR) tuned to that channel in that node, while an inactive vertex represents a channel that is available to a node, but at this time there is no CR tuned into this channel at that node. In constructing the graph, there are rules relating to the status of vertices as active or inactive.

The following is a modified excerpt from the LGM paper explaining how to populate the edges in the graph. With the exception of internal edges, all connections described in the following steps require two edges (one in each direction). The internal edges exception is that they are one way edges from a sub-vertex to its corresponding auxiliary vertex.

- Step 1: On each layer \( i \), if there is a channel available between two potential neighboring nodes, A and B, then let a horizontal edge from \( A_i \) to \( B_i \) exist.

- Step 2: If the number of free interfaces at node A is larger than 1, then connect all layers at that node with vertical edges.

- Step 3: If the number of free interfaces at node A equals 1, then for any active sub or auxiliary vertex and any inactive sub or auxiliary vertex, let vertical edges exist (this step is adding vertical edges between those sub or auxiliary vertices that are active and those that are inactive).

- Step 4: For all active sub and auxiliary vertices, let vertical edges exist.

Now that the graph is built, the algorithm for routing and interface assignment can start. For each node pair, a path is chosen, then, using the interface assignment algorithm described later, the graph is updated, and then the path for the next pair is computed and so
forth. The LGM paper notes that the coder should prioritize the paths as he/she sees fit. In
other words, if there is a particular pair that has high importance then that pair should be
handled first. The other thing to note is that selecting a path, means to use a shortest path
algorithm to find the path with the lowest weight.

To assign the interfaces and update the graph, the LGM authors provide the following
algorithm.

- Step 1: For each inactive sub or auxiliary vertex along the path, assign and tune a
  free interface of that node to the channel associated with said sub or auxiliary vertex;
  record these vertices as active.

- Step 2: If the number of remaining free interfaces at a node is equal to 1, delete all
  vertical edges between any two inactive sub or auxiliary vertices of that node (but
  keep the vertical edges between an inactive and all active sub or auxiliary vertices of
  the node).

- Step 3: If there are no more free interfaces at a node, disable all inactive sub or aux-
  illary vertices of that node by removing all horizontal and vertical edges connected
to these inactive vertices, and record the removed horizontal edges to be added back
to the graph in the future when a free interface is available.

- Step 4: Whenever a path is computed, the cost of all horizontal edges connected
to a sub or auxiliary vertex in this path is increased by 1 (Note that some of these
  horizontal edges are not traversed by the path; they are only incident to a vertex in
  the path).

Since it is a very complex algorithm with a lot of steps, the following should summa-
irize the key operations. The negative weights for the vertical edges encourage the path to
change channels as it travels from node to node. The heavy weight of the access edges
represents an unassigned interface having to change channels in order to receive a packet
on one channel and transmit on another. The internal edge weight represents the increased channel interference caused by overusing a channel. The increasing horizontal edge weight represents the growing amount of traffic going through any given spot in the graph. It does a great job reducing channel interference and, as will be seen in the results section, has good throughput.

There are some issues with the LGM however. The graph makes a very thorough representation of the possible paths, and channel assignments; however it might be too thorough. The results section will show that the run times for this algorithm were prohibitively slow, even on a powerful desktop computer. As one can imagine, as the number of nodes increases, the number of vertices and edges in the graph rapidly increases in number. This would not be such a problem if the end result did not require a shortest path algorithm for the graph.

Most of the shortest path algorithms find the shortest path from a specified source to a specified destination and then as a side effect end up resolving the shortest path to any destination. That would be a great side benefit, but the LGM then modifies the graph after every pair is computed, which means none of the work done in finding the last shortest path is actually valid anymore. You end up computing the shortest path more times than ideally necessary. This number adds up quickly for larger graphs, on top of the fact that the LGM has a graph that grows exponentially with the size of the network.

The LGM did provide a lot of inspiration in creating the proposed solution. The proposed solution also uses a graph model, but the graph is a much simpler one than the LGM, which allows it to be more manageable. It also only requires the shortest path algorithm to be run \( N \) times for a network with \( N \) nodes, which also helps it scale better.
2.3 Spectrum Aware On-demand Routing Protocol

The other protocol examined was the Spectrum Aware On-demand Routing Protocol (SORP), which was presented in Cheng [2]. The basic idea behind this protocol is that it is a modified version of the Ad hoc On-demand Distance Vector (AODV) routing protocol [9]. SORP modifies AODV to make it spectrum aware as its name implies. By making it spectrum aware the benefits of AODV are now brought to the DSA environment.

SORP assumes that the nodes have available at least one flexible radio interface, such as a CR, and also an 802.11 interface for use as a common control channel (CCC). The CCC is a widespread concept in DSA. In typical wireless networks the nodes are preset to know how they are going to communicate with other nodes. In a DSA environment the method of communication must be established. If there is a CCC then there is a guaranteed way to communicate with others and establishes the communication parameters. There is research underway to find a good way to remove the need for a CCC since that would make DSA even more powerful.

SORP uses the CCC to send Route Request (RREQ) packets out to the network. The RREQ is modified from AODV to include SOP information in the packet. A key feature is that the RREQ is not forwarded by a node if it does not share any common SOP with the node that sent the packet. This ensures that when the RREQ arrives at the destination node the packet was forwarded along a path that is feasible given the current SOP throughout the network, as well as that the nodes are within the radio range needed to communicate. Now that the destination has received the RREQ, it chooses a SOP that it has in common with the previous node, and assigns that band to its transceiver. To establish the path the destination node initiates a Route Reply (RREP). This message is sent back to the source and contains the AODV RREP with an addition of the assigned band for the previous hop. The previous node now knows what band to send to the destination on and can use the SOP information from the RREQ to assign a band for it to communicate with the hop before it.
This chain of events establishes and prepares the path.

Cheng [2] presents many equations that SORP uses to calculate delay. The concept behind SORP is to reduce the delay in transmission in a DSA environment with a focus on multi-flow multi-frequency scheduling. The hope is that by scheduling properly and using multiple bands, this protocol can reduce overall communication delay. The communication delay is a combination of switching delay, route delay, and back-off delay to name some.
Chapter 3

A Link-quality-aware Graph Model for Routing Topology Management

This chapter describes the Link-quality-aware Graph Model for Routing Topology Management that resulted from this research. The first section describes the algorithm via several subsections. The remaining sections explain the simulation tool used to test the proposed solution (as well as the layered graph model).

3.1 Algorithm

This section will describe the Link-quality-aware Graph Model for Routing Topology Management (LGMRTM), which is the proposed solution to cross-layer design of the MAC and routing protocols in a DSA environment. As stated earlier it is inspired by the LGM, and similar to the LGM it is a centralized proactive routing protocol. Recall that this means the processing is done on a centralized sever, and the routes and channel selections are computed before they are needed.

3.1.1 Prioritization

The algorithm is a very simple one, yet also quite effective. The first part is a preparation step to allow the algorithm to work in the intended way. The ultimate goal of the first step
is to create a prioritization of the order in which links between nodes should be chosen and established. The algorithm prioritizes by examining how many channels are shared between a pair of nodes. If a pair of nodes has very few choices of channels to connect on then that pair gets handled before a pair that has more channels to choose from. Accomplishing this prioritization required using a structure to hold an array of the common channels for each pair, and then a separate variable to hold the numbers of channels in those arrays. The algorithm sorts the array containing the numbers of channels from least to greatest, and then uses that order to loop through the array that holds the sets of channels for each pair. Since it is simplest to have one-dimensional arrays for sorting, those two data structures are indexed in an unconventional way. Each pair only needs to exist once in the array, meaning the pair of 3 to 1 does not exist because the pair of 1 to 3 would have been created already. To track which node pair was represented by which index into these arrays a couple of other data structures were used so that a pair of nodes could be used to lookup an index and an index could be used to retrieve the pair of nodes.

3.1.2 Channel Selection

Once the prioritization has been established, the algorithm can start to perform the channel selection. Note that the prioritization accomplished some necessary work in discovering all of the channels available to each pair. The idea is to loop through each pair starting with the pairs with the fewest choices and ending with the pairs with the largest number of choices. For any given pair the procedure is to retrieve the set of channels that can be used, then evaluate each channel, and select the best choice. To evaluate the channels, a fitness function is used, which as stated in the background section was inspired by the investigation of using genetic algorithms to solve the problem. The details of the fitness function will be described after the explanation of the rest of the algorithm. Once the evaluations are finished, each channel has a fitness value, and the channel with the best fitness value is chosen to be the channel that this pair communicates on. To represent this selection, in the graph two edges must be created; one traveling in each direction on the chosen channel.
The weight of the edges is set as the fitness value. The fitness function is designed such
that the lower the number, the more fit a channel is. This was done so that the fitness values
could be used as the edge weights, since a lower edge weight is better in the context of
shortest path algorithms in graph theory. The last step is to update the “chanUsage” data
structure. This is an array with as many elements as there are channels existing in the sys-
tem. The entry for each channel is a number representing how many times that channel has
been selected as the channel to use in previous iterations of the loop. This information is
used in the fitness function.

3.1.3 Routing

Once all of the selections have been made, all of the edges now exist in the graph. The
next step is to create all of the vertices for the graph, which are initialized as follows. The
“distance” field is set to infinity, the “predecessor” field is set to blank, and the “predChan”
field is set to blank. This initialization was purposefully designed to be the same as in
the shortest path algorithm. So now the algorithm loops for each node and executes the
shortest path algorithm using the current node as the source node to find the shortest paths
from the current node to each other node. Inside the loop there is another loop that iterates
through all destinations. For each destination, the path is found by tracing through the
predecessors starting at the destination. The predecessor chain will end at the source. The
information about the path taken and channels used are recorded during this trace. When
this set of loops finishes, the routing paths, as well as the channels used for each hop, have
been established, and the algorithm ends.

3.1.4 Fitness Function

One of the keys to the algorithm is the fitness function. The fitness function is simple
and flexible. It is a weighted summation that can range from 0 to 1. Twenty percent of
the fitness value is based on the BER, 20% on the SNR, 25% on IT, and 35% based on the
channel rating. This break down is shown in Equation 3.1. The channel rating is derived from the “chanUsage” variable mentioned above. The channel rating is the number of times the channel in question has been used divided by the number of times the most used channel has been assigned. If this is the first channel assignment and no channels have been selected then the channel rating is 0. Recall that low numbers are better. One great thing about the fitness function is that due to the simplicity the weights of each variable can be tweaked and tested. The initial values chosen were somewhat arbitrary, but based on which factors should hold the most weight. Through trial and error the current values have been decided upon. Obviously the channel rating needs to hold a lot of weight because it is the channel rating that helps minimize channel interference by forcing the algorithm to use a variety of channels. The other factors evaluate the quality of the channel. If the quality of the channel is poor, it lowers the rating which may mean that a channel that has been used more, but is of significantly higher quality can be chosen as the channel to use. The equation is written in such a way that each factor is normalized to a 0 to 1 scale, and then multiplied by its weight. Since the weights add to 100%, the equations yield fitness values from 0 to 1, where 0 is the best and 1 is the worst. The SNR variable needs to be manipulated in the normalization to make it such that high SNR creates a low value for the SNR rating so that the meaning of the SNR rating is not inverted from the rest.

\[
fitVal = \frac{ber}{0.2} * 0.2 + \frac{30 - snr}{30} * 0.2 + \frac{intTemp}{10} * 0.25 + chanRating * 0.35 \quad (3.1)
\]

### 3.1.5 Shortest Path Algorithm

To find the shortest paths in the graph created, the Bellman-Ford shortest path algorithm was used. While Dijkstra’s algorithm is certainly the most famous algorithm, warnings were found that it should not be used on graphs with negative edge weights because it has no way of detecting a negative edge weight cycle. Since the LGM has negative edge weights and it was implemented first, a different algorithm had to be chosen for finding the shortest path. The Bellman-Ford algorithm is another classic algorithm for doing this,
and it has negative edge cycle detection. The Bellman-Ford algorithm works as follows. The first step is initialization, where the distance to the source is set as zero, while all other distances are set to infinity. Each node’s predecessor is set to none. Then a nested for loop is started in which the outer loop goes for as many iterations as there are vertices, and the inner loop iterates a number of times equal to the number of edges in the graph. The inner loop is looping through each edge, so the two vertices connected to the current edge are examined. If the distance to the vertex where the edge terminates is greater than the distance to the vertex where the edge originates plus the weight of the edge, then using that edge is part of the shortest path at this moment. The distance of the termination vertex is updated to reflect the lower value and the predecessor of that node is set to the origination vertex. As the algorithm loops through the set of edges multiple times the distances and paths will converge. After examining each edge a number of times equal to the number of vertices, the graph will have converged unless there is a negative edge cycle. Once the looping is done there is a separate loop that checks the edges one last time, and if there is any point where there is not convergence then there must be a negative edge weight cycle. That concludes the Bellman-Ford algorithm.

In the interest of making the algorithm go faster for larger graphs (mostly due to the massive size of some of the LGM graphs), the algorithm was modified such that if the algorithm goes ten rounds in a row of examining the edges without a change then the paths are considered to have converged and the algorithm stops prematurely. The graph is still checked for the negative edge weight cycle. As will be seen the LGM still takes a long time even with this optimization.

### 3.2 Simulation Tool

The protocol has been implemented in Matlab. When the main program is run it generates a figure window with two buttons in the upper left and a plot in the rest of the window. The simulation can be run with any specified number of nodes. The area represented by
the plot is a square of 1000 meters on each side. The plot shows the positions of the primary and secondary users, and when routing paths are computed it also shows the routing topology including channel selection as can be seen in Figure 3.1.

Figure 3.1 shows an example with 7 primary users, 15 secondary users, and 10 total channels. The primary users are shown as circles while the secondary users are shown as squares. The lines connecting secondary users indicate they are connected. The legend decodes how the different line styles correspond to the channel being used. As can be seen only nodes 5 and 6 are susceptible to channel interference since they each used channel 5 twice. Looking at the specific channel statistics behind the scene at the time of routing,
channel 5 happened to have very good IT and SNR values, which made the fitness function deem it suitable for reuse.

### 3.3 Updating the Node Positions

Each node, whether it is a PU or an SU, has an X-Y pair of coordinates that keep track of where it is on the plot. To add some realism to the scenario being created, the nodes move since the assumption is that the simulation is of a MANET. There is a function for updating the position of all of the nodes, which works as follows. In any call to the function that updates the positions, each node has a 1 in 3 chance to enter the movement code. Once in the movement code, there is separate movement for the X and Y directions. There are seven possible moves for each coordinate, ranging from -3 to +3, with 0 in the middle. The value chosen is multiplied by 10 so that the node can move up to 30 meters in the positive or negative direction along the axis. The X and Y movement are independent of each other. If the new position were to be less than 0 or greater than 1000 for either coordinate, then the new position is simply set as the boundary value.

At the end of the position function the simulation updates which nodes are neighbors based on distance and radio range. Each radio is set to have a range of 300 meters, and if two nodes are within 300 meters, as determined by the distance formula as applied to a two dimensional plane, then they will be considered to be able to reach each other. This is useful for finding potential connections when the algorithm is running.

### 3.4 Primary Users

Primary users are simulated as an array of structures. Each entry represents a primary user, and the fields of the structure are as follows. There is an “id” field to give each primary user an identification number, this was not utilized in the simulations done for this thesis, but if the primary users were controlled in a more sophisticated manner where they could
enter and leave the area, then identification numbers would be useful. The next two fields are the X and Y coordinates of the primary user on the plot. There is a field containing the channel that the PU is occupying. If the PU is not using a channel, which represents a cell phone user (primary user) who is in the area, but not making a call, then the occupied channel is set to 0. The radio range field mentioned earlier is 300 meters for all nodes. The interference range field never got used in the algorithms simulated for this research. There is a similar field for secondary users that did get used. There will be more about that when the secondary users’ structure is discussed in detail. The next two fields have to do with simulating the primary users using spectrum. The “usingComms” field is a flag to indicate that the PU is using spectrum, and the “commsLength” field indicates how long the PU has been using the channel. The channel is specified by the “chan” field as mentioned above.

This paragraph explains how the PU structure is used to simulate a PU. The simulation simply consists of updating the PU structure. All varying aspects of the PU are updated in a function whose sole purpose is to update the primary users. The one exception is that the position updates are handled in a separate function that does position updates for both the primary and secondary users as explained above. The PU update function updates the communication status of the PU. If the PU is using a channel and has been doing so for more than a minute, then the PU releases the channel. This is making a simple assumption that the average phone call or spectrum usage will last a minute. That could be adjusted with real statistics, but it serves no purpose other than to make it such that when spectrum is taken by a PU they take it for a finite period of time and then release it. If the PU is using spectrum and it is been less than a minute, then the clock is incremented and no other action is taken. If the user is not using a channel, then the user has a 10% chance to decide to use a channel. If the user is going to use a channel then the PU selects a random channel from the remaining available channels (it cannot choose a channel in use by another PU). These updates occur every 5 seconds.
3.5 Environment

Since the proposed solution considers environmental information about each channel in making a selection these pieces of information need to be updated frequently to reflect the dynamic environment of DSA. There are three values tracked for each channel, which represent Bit Error Rate (BER), Signal to Noise Ratio (SNR), and Interference Temperature (IT). For the purpose of the simulation the BER has a range of 0 to .2, but obviously a full hardware simulation could allow it to go higher. The simulated SNR has a range of 1 to 30, and the simulated IT ranges from 0 to 10. Each value is generated randomly, and the SNR and IT are assigned as integers through the use of rounding. These values are updated every 5 seconds.

3.6 Graph Representation

To represent the graphs needed for both simulating the LGM and the proposed solution, two data structures were created; one to represent vertices and one to represent edges. The LGM structures were a little different from the structures used for the proposed solution since the LGM was more complicated and required more information to be stored. The edge structure for the proposed solution is as follows. There is a “from” field representing the node (vertex) at which this edge originates. There is a “to” field representing the node (vertex) at which this edge terminates. The “chan” field states which channel the two nodes are connecting on, and the “weight” field holds the weight of this edge for use in finding the routing paths.

The vertex structure for the proposed solution contains three fields, which are all used for finding the routing paths. The “distance” field tracks how far from the declared source this node (vertex) is. The “predecessor” field keeps track of the previous hop in arriving at this node from the source node. The “predChan” field stores the channel that was used in hopping from the predecessor to this node. These fields were dictated by needs based on
the shortest path algorithm used to compute the routing paths.
Chapter 4

Simulation Results and Comparisons

This chapter starts by presenting and analyzing the results obtained in the LGM and SORP works. The next piece of the chapter presents and examines the results obtained from this research, and the chapter ends with a comparison and cross-examination of the results.

4.1 Layered Graph Model Results

Not all of the results covered here will be from the simulations done during this research. It is important to take note of the results obtained in some of the background work that was studied. In this section, the results in the LGM paper will be discussed and examined. The LGM paper had two sets of simulations, where one was a network of 15 nodes and the other was a network of 30 nodes. There are 6 channels that exist in their simulations where each has a bandwidth of 10Mbps. The number of radio interfaces is randomly generated between 1 and 3 for each node. The nodes generate Poisson packet traffic. The throughput was the measured quantity, and the results are considered to have a 95% confidence interval after doing 10 runs of the experiment with different random seeds. The LGM throughputs were compared to a simple interface assignment heuristic called Sequential Interface Assignment (SeqAssign). SeqAssign assigns channels to interfaces in a descending order of the number of neighbors the channel can reach. So the channels that reach the most neighbors of a given node are assigned to the available interfaces. There will be more discussion of SeqAssign,
and how its strategy compares to that of LGM and the proposed solution later.

Figure 4.1: Throughput versus Load with a 15 Node Network [14]

Figure 4.1 shows how LGM performed against SeqAssign in a 15 node network as the load increases. LGM degrades gracefully in a linear fashion as the load increases from 10 to 60 packets per second. SeqAssign has overall lower performance and has a bit of a curve to it. However its slope seems to decrease some as the load continues to increase. The error bars show the 95% confidence interval. LGM interestingly has a larger confidence interval. Ten experiments were done to generate these statistics. The purpose of looking at what percentage of the load could be handled is to see how well the LGM mitigates the issue of channel interference. The plot shows that the LGM can achieve higher throughput by acknowledging and addressing channel interference.

Figure 4.2 shows the results of a similar test using 30 nodes. The confidence interval on the LGM is still very large, perhaps a little larger, while the confidence interval for SeqAssign are quite small. This seems to indicate that the LGM can be inconsistent in its results, which may be a result of randomizing the number of interfaces available at each node. LGM could be quite affected depending on how many total interfaces exist in the network. The meaning of this confidence interval, which would be related to the standard
deviation, based on 10 data points is usually considered unreliable since there are so few points with which to derive the confidence interval.

4.2 Spectrum Aware On-demand Routing Protocol Results

This section will discuss the results from the SORP paper. The SORP simulations placed up to 100 simulated wireless nodes in an 1800m by 1800m square. As assumed in the algorithm, each node has one 802.11 interface as well as one flexible interface. Each node had anywhere from 2 to 8 available frequency bands at any point in time. The simulations done for the paper compared SORP with two other styles of routing; one that is aware of the switching delays introduced by changing channels and another that tries to be K-hop distinct (changes channels between hops often) to reduce channel interference. The authors introduce a metric called Sparsity of Spectrum Distribution which describes the average difference between two consecutive frequency bands in the pool of spectrum opportunities. The authors simplify the meaning by saying that the greater the SSD the higher the average switching cost. This makes sense given the definition, because the larger the gap between two frequencies, the longer it takes a CR to tune to the new frequency. Their general idea in testing with increasing SSD is to find out how well their algorithm can minimize delay

Figure 4.2: Throughput versus Load with a 30 Node Network [14]
as the average time to change channels increases.

Figure 4.3 shows the performance of the 3 routing styles studied in the SORP paper as they pertain to delay in an environment with increasing SSD. As explained above, the rising SSD represents a longer average switching delay between any two channels, since the SOP are spread further apart on average. SORP performs well as it has a flat curve in the end which shows that higher SSD really does not make SORP degrade. The switch-aware protocol they compared against also did well because its main focus is analyzing the delay introduced by changing channels. The K-hop distinct performs poorly under these conditions because it tries to change its channels often, which creates larger and larger delays as the distance between channels increases.

In addition to analyzing how well SORP can handle increased switching cost, the authors also wanted to see how well their algorithm could handle a rising number of intersecting flows. A network has intersecting flows when a node is handling more than one stream of data at a time. For instance, if a node is sending a series of packets to a destination, and
at the same time is forwarding a different series of packets that are using that node as an intermediary in the path, then that node has two intersecting flows. There are two distinct flows of traffic at the node. The goal of SORP was to minimize both node delay, which was evaluated with how well SORP dealt with the rise in SSD, as well as maintaining low path delay, which is tested by increasing the number of flows. If a node must deal with several flows at once, they will have a channel interference problem if there is not enough channel switching. At the same time the rising SSD would be a problem if the algorithm is switching channels too often.

Figure 4.4 shows the SORP paper results for delay as the number of intersecting flows increased. All three algorithms being compared struggled with this task as would be expected. Unlike the rising SSD all three have consistently rising curves. As might be expected SORP has the best results. The switch-aware and K-hop distinct perform similarly, while SORP has a similar curve, it is noticeably lower on the graph, representing reduced delay. What happens is as the number of intersecting flows increases, the K-hop distinct
keeps changing channels often, because that is its goal, which increases delay. The switch-aware does its best to not switch channels, which creates channel interference, causing delay. SORP, on the other hand, is balancing those two priorities to get the best of both worlds. Hence SORP does the best since it balances the two.

4.3 Link-quality-aware Graph Model Simulations and Results

This section will be split into two parts. The first will cover the results obtained through the simulations done for this research. The second part will be an analysis of the results obtained in this research in relation to those found in the work done by the LGM authors as well as the SORP authors.

The simulations done for this research were done to take metrics on a different set of parameters than the simulations in the LGM and SORP research. Since the proposed algorithm has a goal of reducing channel interference as the LGM does, a way of measuring the diversity in channel assignment was created. It is a ratio of the number of unique channels used to connect divided by the number of unique neighbors with which connections were made. The following is an explanation of how to compute this measurement. Once the routing paths are established, each node is looped through. For each node, each destination is analyzed. For each destination take note of both the neighbor and the channel used in the first hop. Compile a list of first hop neighbors and first hop channels such that each channel and neighbor used appears once in the list. The length of the channel list divided by the length of the neighbor list is the channel diversity. By summing the lengths of the channel list for each node first and then dividing by the sum of the lengths of the neighbor list for each node the channel diversity of the whole routing topology is measured. The only rule is that the channel diversity cannot be greater than one for any individual node (and therefore the overall value cannot be greater than one either). This issue arose because
the LGM can have instances where depending on the destination the source could send to
the same neighbor on two different channels for two different destinations. To keep from
having one node be very diverse and cover up other less diverse nodes, for any given node
the length of the channel list will not be recorded as greater than the length of the neighbor
list. So any node that is using multiple channels to connect to the same neighbor is simply
much more likely to contribute a ratio of 1 to the average.

The other measurements taken were the average number of hops in the routing paths and
the length of time it took the algorithm to compute. The time factor was introduced when it
became apparent that the LGM did not have fast run times. At that point, one of the goals
for the proposed algorithm was to have a significantly smaller run time. The goal in general
for the proposed solution is to provide similar results as the LGM, because the ideas behind
it are good, while doing so in a manner that is far less computationally expensive.

The simulations were done in such a way that both the LGM and the proposed solution
worked on the same scenarios. The positions of the primary users and secondary users, as
well as the available channels were established and sent to the algorithms as a representa-
tion of the network at an instant in time. The number of channels existing in the simulation
was kept at a constant 10 channels, however from run to run the number of channels being
used by primary users was subject to change since it was random whether or not a pri-
mary user would be using a channel. Simulations were conducted for 10, 15, 20, 30, and
50 nodes. For each network size, 100 scenarios were created, and the LGM and LGM-
RTM were each run on all 100 to formulate the statistics. For the 50-node simulations
the LGM was taking so long to run that there was only time to run 25 scenarios with 50
nodes. To keep the playing field completely even, the LGMRTM was not given any extra
50-node scenarios since that could have skewed the statistics for the LGMRTM for 50 node
networks.
Figure 4.5 shows just how large the run time is for the LGM. In comparison to the LGM the LGMRTM looks as though it has barely any run time and that it is linear. The scale required to show the LGM run times is deceiving. The run time for the LGMRTM is also exponentially rising as can be seen in Table 4.1, but it simply starts at a much lower time and has a less steep curve than the LGM. When first simulating the LGM it was apparent right away that the run time was surprisingly long. The ideas behind the LGM are great, but the huge times made it quickly apparent that the goal of this research should be to find a way to achieve similar channel interference benefits with an algorithm that was less complicated. That was the inspiration for the LGMRTM.

Figure 4.6 is a plot of the channel diversity metric of the LGM and LGMRTM versus network size. The plot shows that while the LGM does have overall better “diversity” in channel assignment, it is not overwhelmingly better. The purpose of measuring this metric was to make sure that the LGMRTM does not have significantly lower channel interference avoidance. Interestingly enough, they start at roughly the same value, then the LGMRTM
Figure 4.6: Comparison of Avoidance of Channel Interference

Figure 4.7: Comparison of Average Number of Hops
degrades a little faster, but at 50 nodes, they have roughly the same value again. This may mean that once the network gets large enough the diversity levels off. Overall this is considered successful because the performance lost in the diversity metric is more than compensated for in the reduced complexity of the LGMRTM.

Figure 4.7 shows how the average number of hops in each path changes as the number of nodes increases for both the LGM and the LGMRTM. This measurement was taken to verify that the LGMRTM was not creating paths that were significantly longer than the LGM. If it was, that could have impacted its throughput. They have nearly identical path lengths, which demonstrates further that the LGMRTM should be able to produce similar throughput results with less complexity than the LGM.

Tables 4.1, 4.2, and 4.3 show all of the statistics from the simulations performed. The mean, minimum, maximum, and median values for each network size and both algorithms are shown.

The results (both from this research and other research) have been presented, and in the rest of this section some analysis will be presented. Examining the SORP results will be the starting point. SORP was compared to a “K-hop distinct” method (changes channels between hops often) and a “switch-aware” method (aware of the delay in changing channels). The SORP results showed a smaller delay than both of the alternatives, but there are some things to consider. It is not clear whether the SORP delay times are including the time it takes to send out the RREQ and receive the RREP. It is quite possible that the delay is roughly three times that if the RREQ and RREP travel times were ignored. SORP was trying to improve upon some of the concepts from the LGM, but it is unclear whether they did. LGM is essentially hybrid of a K-hop distinct algorithm and a switch-aware due to the fact that it rewards changing channels from hop to hop; however it also attempts to assign interfaces such that these channel changes only require using a different interface and not actually tuning an interface. The LGM hybrid is closer to a K-hop distinct style than a
Table 4.1: Run Time Results

<table>
<thead>
<tr>
<th># of Nodes</th>
<th>LGM</th>
<th>LGMRTM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Min</td>
</tr>
<tr>
<td>10</td>
<td>7.6417</td>
<td>3.5245</td>
</tr>
<tr>
<td>15</td>
<td>36.777</td>
<td>16.494</td>
</tr>
<tr>
<td>20</td>
<td>134.18</td>
<td>58.14</td>
</tr>
<tr>
<td>30</td>
<td>676.87</td>
<td>298.43</td>
</tr>
<tr>
<td>50</td>
<td>6887.2</td>
<td>3502.1</td>
</tr>
</tbody>
</table>

Table 4.2: Diversity Results

<table>
<thead>
<tr>
<th># of Nodes</th>
<th>LGM</th>
<th>LGMRTM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Min</td>
</tr>
<tr>
<td>10</td>
<td>0.9552</td>
<td>0.8125</td>
</tr>
<tr>
<td>15</td>
<td>0.9507</td>
<td>0.8155</td>
</tr>
<tr>
<td>20</td>
<td>0.9063</td>
<td>0.7386</td>
</tr>
<tr>
<td>30</td>
<td>0.7326</td>
<td>0.5440</td>
</tr>
<tr>
<td>50</td>
<td>0.4917</td>
<td>0.4191</td>
</tr>
</tbody>
</table>

Table 4.3: Path Length Results

<table>
<thead>
<tr>
<th># of Nodes</th>
<th>LGM</th>
<th>LGMRTM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Min</td>
</tr>
<tr>
<td>10</td>
<td>1.5755</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>2.2285</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2.6779</td>
<td>1.5472</td>
</tr>
<tr>
<td>30</td>
<td>2.8198</td>
<td>1.8631</td>
</tr>
<tr>
<td>50</td>
<td>2.6909</td>
<td>2.38969</td>
</tr>
</tbody>
</table>
switch-aware style, but it is still a hybrid because it does not blindly change channels at every opportunity.

A big issue with the SORP simulations is that they provided only a single flexible radio. Clearly the alternates would benefit from having more radio interfaces, unless the alternates were chosen because they also were designed with a single flexible radio in mind. Even SORP should have been able to benefit from more radios. With more interfaces, it could do better with multiple flows, since it could service more than one flow at a time on different channels simultaneously. By the time the SORP research was underway it had been established that it was reasonable to assume that the devices that will support DSA are not necessarily going to be limited to a single CR or other flexible radio. One would expect great performance increases from adding a second CR to DSA devices since it would allow the device to operate on more than one channel simultaneously which means either using two SOPs to talk to the same node (increasing the total bandwidth of the connection), or handling multiple flows at the same time.

The LGM needs to be assessed as well. Clearly the biggest issue with the LGM is the run time. Perhaps when the authors implemented it they used a faster environment than Matlab, or maybe the authors really did not care about run time. As it is explained in the paper, it is a centralized proactive routing protocol. It is not done on demand, nor is it executed by the nodes themselves, which means that a super powerful central server could potentially have the power to deal with larger networks. It does have great results as far as reducing channel interference as can be seen in the results from [14]. Something bothersome about the LGM simulations was the use of the SeqAssign algorithm. Just from the description it becomes clear that it is not a good method of channel assignment in a DSA network. When there are so many choices, assigning the channels without examining the routing layer has little chance in performing well compared to a protocol that utilizes cross-layer design. SeqAssign selects the channels with no regard for the fact that routing will need to take place based on this assignment. This could end up leaving the network with a suboptimal
Figure 4.8 demonstrates the perils of SeqAssign. Assume that each node has two interfaces; when using SeqAssign, those interfaces are tuned to the channels that connect the nodes to the most other nodes. The solid lines indicate the connections that were made. The dashed line between nodes A and F show the potential connection those nodes could have made on channel 1, but did not. Therefore the network is not fully connected, which is a bad thing. In the same situation the LGMRTM would have assessed that nodes A and F should have a channel assigned to connect them as a priority to maximize network-wide connectivity. For example node A could have chosen to tune channels 1 and 2, allowing a connection to the other side of the network, meanwhile A can still reach nodes D and E via node C. It may increase the average number of hops per route, but at least the two halves of the network would not be isolated.

The LGM integrates the channel assignment into the route discovery. It shifts back and forth between routing and channel assignment until all choices have been made. In comparison, the proposed solution on the surface may seem to take a SeqAssign approach
and just assign interfaces and route later, but it really does not. It is not so much about assigning interfaces, in reality that is the last step it takes. It actually is choosing how to connect the different nodes. It does not pick a channel based on an individual node; the algorithm considers the connections between all pairs of nodes. This allows the algorithm to ensure that connections are made where they need to be. The prioritization of which connections are made first ensures that no potential connection gets cut off. It also appears as though SeqAssign will not allow any interface to change channels to potentially allow a node to communicate on more channels than it has interfaces. While this channel change may take time, it could enable a connection that connects two pieces of a network as in the example above. The proposed algorithm separates channel selection from interface assignment, where SeqAssign combines the two, and LGM ties channel selection in with route selection.
Chapter 5

Conclusion and Future Work

This chapter draws conclusions from the information presented throughout this document as well as gives guidance and ideas as to what further research could be done to extend the ideas proposed in this paper.

5.1 Conclusion

The first conclusion that should be drawn from this research is about the use of graphs to model networks. Clearly this has to be done with caution. The run times for the LGM show that while using a graph to model the network connectivity can seem brilliant and simple, it can create a very complex graph. Of course as the graph gets more complex, the run time to find the shortest path goes up. As the size of a graph grows, the complexity rises by default. Even if a graph with 500 vertices is barely connected, it still has 500 vertices and maybe 1000 edges if each node only connects to two others. Too many vertices and too many edges take an excessive amount of time to compute.

Logic and inspection should be enough to conclude that a centralized cross-layer protocol can deliver the good overall results in terms of routing and channel selection/assignment. By having one entity process for all nodes, the algorithm can see the big picture and create a topology that balances channel interference with switching delay. The LGM demonstrated
this concept, but it was too computationally expensive to be sensible. The proposed solution balances use of a graph to model the network while minimizing the size of the graph to make it more manageable when it comes to finding the routing paths via a shortest path algorithm.

5.2 Future Work

This research leaves off at a point where it can be extended, and hopefully can inspire others to create other ideas that, do not necessarily build off of the proposed solution, but rather borrow some concepts from it. There are two main concepts that should be discussed. The first is an idea for improving the current algorithm. The proposed solution leaves interface assignment to the end. Once channels have been chosen and paths have been computed, the interfaces are assigned based on the choices. Suggesting that interfaces be assigned as the channels are selected might suffer from the issues that SeqAssign has, but what might be a good idea is to add extra steps to the algorithm. Analyze the routes computed, and order the list of links based on how many paths use each link. From most used to least used, go through and assign an interface on each end of the links. Allow the interfaces to run out, in other words do not save any to be able to juggle multiple channels on a single interface, pick a channel. Any edge that no longer exists because either one or both ends of the connection did not have a free interface is now removed. At this point, take this smaller graph, and re-run the shortest path algorithm to find a new set of routes. It would be interesting to compare the two topologies to see how much the connectivity is affected. This could break connections, but if they were not highly used it may not matter unless it strands a node or set of nodes, but connections that would separate one or more nodes from the rest of the network would probably be one that was used often enough in the routing to be assigned an interface on the necessary nodes in the first place. The possible downfall of this idea is that it may introduce too much extra processing, unless there is a way to approximate the usage of the different connections without actually
running the shortest path algorithm the first time. This could certainly improve delay in
the network over the proposed solution depending on how often the proposed solution is
actually juggling more channels at a node than it has interfaces.

The other thing to consider in future work would be to somehow use a graph to model
the network, but not need a shortest path algorithm to find the paths. If the graph is used
to accomplish a task other than actually finding the paths, and then the paths are found
using a method other than shortest path, then the graph is still useful, but will not become
a bottleneck by taking a long time to resolve paths.
Bibliography


