ABSTRACT

BISWAS, TRISHA. Redundancy-based Approaches in Wireless Multihop Network Design. (Under the direction of Rudra Dutta.)

Wireless multihop networks such as ad hoc and mesh networks consist of wireless nodes without any fixed base stations or a backbone infrastructure. Such networks are characterized by frequent host mobility, limited power reserve and interfering transmissions. The wireless medium is susceptible to random uncertainty of radio links as well as malicious jamming. While the former is observed in the form of intermittent link failures, the latter is observed as geographically correlated failures in the network. Routing in such networks usually needs to be robust, so as to adapt itself to the dynamics of the network.

Traditional unipath routing techniques reduce the overall packet delivery ratio in presence of network failures. Multipath routing is a common approach to increase the reliability and end-to-end throughput. In this research, we first explore some redundancy based multipath routing techniques in wireless mesh networks. We consider a network of nodes addressed by their locations, and propose a diffuse pathset routing technique. The goal of this approach is to increase reliability of transmissions in failure prone networks. The geo-diffuse multipath routing technique, called Petal Routing, takes advantage of the broadcast nature of wireless networks to reduce the number of transmissions over multiple paths. We compare this approach to a geo-diverse routing technique, which carries out power control to protect against jammers in the network. We highlight scenarios under which one performs better than the other, and also compare them to existing multipath routing techniques.

Transmissions in wireless networks may require performance guarantees, which can be achieved by using advanced routing strategies. Such guarantees may not be delivered due to vulnerabilities such as incorrect neighborhood information, high probability of random link failures and jammers. We examine one such performance metric, namely reliability, or packet delivery ratio. Being able to predict the reliability of a transmission a priori can be useful in
designing robust networks. To this end, we present an analytical model that can predict the reliability of a transmission being carried out using Petal Routing, given certain parameters such as network node density and region of network involved in the transmission.

We also present a more general predictive model for wireless networks inspired by hybrid control theory. We model the network as a stochastic dynamical system where the wireless links are either stationary availability probabilities or Markov transition probabilities. The system we present has a centralized controller that pre-computes optimal paths to the destination by carrying out reachability analysis in order to obtain predictive results. We investigate the accuracy of the theoretical model by comparing reliability predicted by the model and that achieved by OPNET simulations of standard routing protocols such as OLSR and AODV. This technique can provide the highest achievable reliability for a given transmission, and thus be used for benchmarking and improving existing routing protocols to make them more robust to wireless failures.

Lastly, we explore the timescale of wireless link failures in an effort to make our predictive model more realistic. While the two-state up/down Markov model can be used to represent wireless links, the actual transition probabilities from these states can vary widely based on what the observation timescale is. To make our model emulate real wireless links, we carry out some experiments on 802.11 wireless links and measure the timescale of link level fluctuations. The results of these measurements can be used both to design realistic predictive models, as well as routing protocols that consider the link level fluctuations while deciding on the best next hop to deliver a packet to.
Redundancy-based Approaches in Wireless Multihop Network Design

by
Trisha Biswas

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2014

APPROVED BY:

__________________________  _______________________
Mihail Sichitiu            Xuxian Jiang

__________________________  _______________________
Huaiyu Dai                Rudra Dutta
Chair of Advisory Committee
DEDICATION

To my parents, Sagarika Biswas and Rana Biswas,
my sister and brother-in-law, Brinda Biswas and Melvin Devasia,
and my best friend Arpan Chakraborty,
for their endless love and support.
BIOGRAPHY

Trisha Biswas is a Ph.D. candidate at North Carolina State University, Raleigh, NC. She was born and raised in Kolkata, India. She received her B.Tech degree in Computer Science and Engineering from West Bengal University of Technology. She joined the Ph.D. program at NCSU in 2008 under the supervision of Dr. Rudra Dutta and received an en route M.S. degree in Computer Science in May 2011.

Over the course of graduate school, she has been actively involved in various student organizations like STARS, WiCS and CSC-GSA. Through STARS and summer camp instruction, she has helped with outreach activities aimed at broadening participation in computing. She is also a student member of the Institute of Electrical and Electronics Engineers (IEEE) and the Association for Computing Machinery (ACM). As a hobby, she pursues Indian classical vocal music.

After graduation, she will join Aruba Networks as the next step in her career, working on the emerging paradigm of Software Defined Networks.
ACKNOWLEDGEMENTS

Many people have encouraged and supported me, without whom this journey would not have been possible.

I would like to thank my advisor, Dr. Rudra Dutta, for his invaluable guidance and mentorship. You continue to be a source of inspiration for all that is possible in an academic career and life. I would like to offer my special thanks to Dr. Mihail Sichitiu, Dr. Huaiyu Dai and Dr. Xuxian Jiang for serving on my advisory committee and for providing valuable suggestions.

I am deeply grateful to my parents for always believing in me and encouraging me towards the best. Thanks to my mom for being involved in every part of my life from thousands of miles away. Thanks to my dad for getting me interested in computer science. Thanks to my sister for understanding me better than anybody else. Thanks to my brother-in-law for always being there for me.

To my friends from the Networking Lab (present and past): Can Babaoglu, Magreth Mushi, Rob Udechukwu, Amlakawit Medhin, Harsh Joshi, Anjing Wang and Mohan Iyer - I could not have asked for a better group of people to work with during my graduate career. Special thanks to Arpan Chakraborty for the countless discussions and for helping me conduct outdoor experiments.

I would like to express my appreciation to Dr. Meeko Oishi and Kendra Lesser from the University of New Mexico, with whom I was fortunate enough to collaborate with. It has been an invigorating experience from which I have learned a lot. I am grateful to the Science of Security Lablet and Secure Open Systems Initiative at NCSU for supporting my research.
# TABLE OF CONTENTS

| LIST OF TABLES | vii |
| LIST OF FIGURES | viii |

**Chapter 1 Introduction** 1

**Chapter 2 Context** 5

- 2.1 Problem Definition 5
- 2.1.1 Redundancy Based Routing Approaches 7
- 2.1.2 Modeling Wireless Routing 8
- 2.2 Related Work 9
- 2.2.1 Redundant Routing in Wireless Networks 9
- 2.2.2 Predicting Routing Performance 11

**Chapter 3 Geo-Diffuse Pathsets** 13

- 3.1 Problem Description 13
- 3.2 Our Approach: Petal Routing 13
  - 3.2.1 Basic Petal Routing 14
  - 3.2.2 Petal Routing with Redundancy Tuning 20
- 3.3 Analytical Model for Basic Petal Routing 31
  - 3.3.1 Algorithm for Predicting Reliability 32
  - 3.3.2 Generating Topologies for Comparison 37
  - 3.3.3 Results 39
- 3.4 Analytical Model for Petal Routing with Redundancy Tuning 50
  - 3.4.1 Algorithm for Predicting Reliability 54
  - 3.4.2 Results 57

**Chapter 4 Geo-Diverse Paths for Jamming Mitigation** 60

- 4.1 Problem Description 60
- 4.2 Routing Methodology 61
  - 4.2.1 Jamming Effects Modeling 61
  - 4.2.2 Proactive Protection against Jamming 63
  - 4.2.3 Reactive Protection against Jamming 67
  - 4.2.4 Evaluation of Geo-diverse Routing 71
- 4.3 Comparing Geo-diffuse and Geo-Diverse Techniques 73

**Chapter 5 Empirical Measurements of Wireless Link Fluctuation Timescales** 77

- 5.1 Problem Description 77
- 5.2 Experiment Methodology 78
  - 5.2.1 Outdoor wireless measurements 78
  - 5.2.2 OPNET simulation measurements 80
  - 5.2.3 Visualizing Transition Probability Matrices 83
Chapter 6 Modeling Networks with Linear Systems ........................................ 95
  6.1 Problem Description ................................................................. 95
  6.2 The Model: Multihop Network as a Dynamical System ..................... 96
      6.2.1 Modeling Wireless Link States ........................................ 98
  6.3 The System: Reachability Analysis for Network Performance ............... 99
      6.3.1 Reachability Analysis for Stochastic Dynamical Systems .......... 99
      6.3.2 Reachability Analysis for Wireless Networks ...................... 101
  6.4 Numerical Experiments ............................................................ 103
      6.4.1 Experiment Set-up ...................................................... 103
      6.4.2 Varying Ambient Noise in OPNET .................................. 105
      6.4.3 Using Stationary Link Probabilities ................................ 107
      6.4.4 Using Markov Link Probabilities ................................... 108
  6.5 Evaluations and Results .......................................................... 109
      6.5.1 Stationary Link Probabilities ...................................... 109
      6.5.2 Markov Link Probabilities ........................................... 112

Chapter 7 Conclusions and Future Work ................................................. 119
  7.1 Conclusions .............................................................................. 119
  7.2 Future Work ............................................................................ 120
      7.2.1 Limited Node Information .............................................. 121
      7.2.2 Delayed Node Information ............................................. 121
      7.2.3 Correlated Link States .................................................. 122

References .......................................................................................... 123

Appendix ............................................................................................. 133
  Appendix A Enumeration of Cutsets .................................................. 134
      A.1 Computation of Minimal Node Cutsets ................................ 134
      A.2 Converting Node Cutsets to Edge Cutsets ............................ 136
      A.3 Computation of Minimal Edge Cutsets ................................. 139
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Algorithm 6 Terminology</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>Algorithm 7 Terminology</td>
<td>69</td>
</tr>
<tr>
<td>5.1</td>
<td>State transitions for Example 1</td>
<td>83</td>
</tr>
<tr>
<td>5.2</td>
<td>Transition probability matrices for varying packet rates and receiver signal strengths, for wireless environment with interference and no obstacles</td>
<td>89</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Diffused overlapping pathsets</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Structure of a Petal packet</td>
<td>15</td>
</tr>
<tr>
<td>3.3</td>
<td>Calculating the different petal parameters</td>
<td>18</td>
</tr>
<tr>
<td>3.4</td>
<td>Back-off time to reduce transmissions</td>
<td>20</td>
</tr>
<tr>
<td>3.5</td>
<td>Choosing back-off time based on node location</td>
<td>23</td>
</tr>
<tr>
<td>3.6</td>
<td>(a) Curve representing back-off time varying from center of petal to edges (b) Flattening of the curve from source to destination</td>
<td>24</td>
</tr>
<tr>
<td>3.7</td>
<td>Convex Hull Case 1</td>
<td>24</td>
</tr>
<tr>
<td>3.8</td>
<td>Convex Hull Case 2</td>
<td>25</td>
</tr>
<tr>
<td>3.9</td>
<td>Convex Hull Case 3</td>
<td>25</td>
</tr>
<tr>
<td>3.10</td>
<td>(a) Calculation of cancellation score $S_c$; (b) Inverse sigmoid curve</td>
<td>27</td>
</tr>
<tr>
<td>3.11</td>
<td>Probability of transmission for different node locations</td>
<td>29</td>
</tr>
<tr>
<td>3.12</td>
<td>Visualizing transmission cancellations in small, medium, large networks</td>
<td>30</td>
</tr>
<tr>
<td>3.13</td>
<td>Delay vs. Back-off time for Petal Routing, 100 node network with no Failures, Petal Width = 0.01, Aggressiveness = 3</td>
<td>31</td>
</tr>
<tr>
<td>3.14</td>
<td>Number of Transmissions vs. back-off for Petal Routing, 100 node network with no Failures, Petal Width = 0.01, Aggressiveness = 3</td>
<td>32</td>
</tr>
<tr>
<td>3.15</td>
<td>Reliability vs. Failure probability vs. Petal width for Petal Routing, 150 node network, Back-off technique = Coordinated, $t_{lb} = 0$ ms, $t_{ub} = 5$ ms, Aggressiveness = 3</td>
<td>33</td>
</tr>
<tr>
<td>3.16</td>
<td>Reliability vs. Jamming Power (Petal Routing and Network Coding), 100 node network, Petal Width = 0.01, Aggressiveness = 3, Back-off technique = Coordinated, $t_{lb} = 0$ ms, $t_{ub} = 10$ ms, Network Coding k = 4, $E_k = 2$</td>
<td>34</td>
</tr>
<tr>
<td>3.17</td>
<td>Network topology generated with uniform random distribution</td>
<td>40</td>
</tr>
<tr>
<td>3.18</td>
<td>Sample network topology from OPNET simulation with points plotted in 2D space on the right</td>
<td>41</td>
</tr>
<tr>
<td>3.19</td>
<td>Node model for Petal Routing</td>
<td>41</td>
</tr>
<tr>
<td>3.20</td>
<td>Reliability vs. Petal Width for Simulations and Analytical Model. Topology provided to analytical model</td>
<td>42</td>
</tr>
<tr>
<td>3.21</td>
<td>Reliability vs. Petal Width for Simulations and Analytical Model. Topology provided to analytical model, using Algo: 5</td>
<td>43</td>
</tr>
<tr>
<td>3.22</td>
<td>Reliability vs. Petal Width for Simulations and Analytical Model. Generated topologies used in analytical model</td>
<td>44</td>
</tr>
<tr>
<td>3.23</td>
<td>Reliability vs. Petal Width for Simulations and Analytical Model. Different topologies with Algo: 5</td>
<td>45</td>
</tr>
<tr>
<td>3.24</td>
<td>Reliability vs. Petal Width for Simulations and Analytical Model, topology provided to simulation, with binomials</td>
<td>47</td>
</tr>
<tr>
<td>3.25</td>
<td>Reliability vs. Petal Width for Simulations and Analytical Model, topology provided to simulation, with all binomials</td>
<td>48</td>
</tr>
<tr>
<td>3.26</td>
<td>Reliability vs. Petal Width for Simulations and Analytical Model, clustered topology not provided to simulation, with binomials</td>
<td>49</td>
</tr>
</tbody>
</table>
Figure 5.12 (Left) Transition Probability Matrix Visualization for wireless environment with interference and no obstacles, with a gradual increase in packet rate: 20, 40, 60, ..., 500 packets per second for a -60 dBm link; (Right) Plotting the absolute values, i.e., count of the number of each transition.

Figure 5.13 Transition probability matrices for varying packet rates in OPNET. Packet rates were varied from 20 to 800 packets per second, in increments of 20.

Figure 5.14 Percentage of packets lost for varying packet rates in OPNET.

Figure 6.1 Sample 38 node network generated by OPNET, using random node placement.

Figure 6.2 Varying deadline and ambient noise for a 20 node network.

Figure 6.3 Comparison between pre-computed paths and simulations of the network for stationary probabilities.

Figure 6.4 Comparison between pre-computed paths from the model and OPNET simulations; Deadline = 6.

Figure 6.5 Comparison between pre-computed paths from the model and OPNET simulations; Deadline = 7.

Figure 6.6 Comparison between pre-computed paths from the model and OPNET simulations; Deadline = 8.

Figure 6.7 Comparison between pre-computed paths from the model and OPNET simulations; Deadline = 4.

Figure 6.8 Comparison between pre-computed paths and simulations of the network for Markov probabilities.

Figure 6.9 Comparison between pre-computed paths from the model with Markov links and OPNET simulations; Deadline = 4.

Figure A.1 A sample network and its minimal path tree.

Figure A.2 Converting node cutsets to edge cutsets.

Figure A.3 Converting node cutsets to edge cutsets (2).

Figure A.4 Sample network and the computed minimal edge cuts.
Chapter 1

Introduction

Multihop wireless networks such as ad hoc and mesh networks use two or more wireless hops to convey information from a source to a destination. Such networks can be deployed and made operational nearly instantly, making them suitable for applications such as tactical networks. Challenges of multihop networks include the inherent uncertainty and vulnerability of the wireless medium that gives rise to difficulty in guarding against failures of path transmissions. Failures due to the medium are usually observed as intermittent or random link failures. Geographically correlated link failures could be actively caused by malicious network elements such as jammers. A jammer is a node that transmits continuous signals so that nodes within its range only receive garbled information.

Reliability of transmissions in wireless networks heavily depend on the design of routing protocols. Redundancy is a common technique used for achieving reliability in networks prone to failures. Redundancy is often realized by sending the data over multiple paths from the source to the destination, if the network is well connected. Many redundant or multipath routing techniques exist in the literature [24, 71]. Since redundancy increases the number of intermediate transmissions, there exists a tradeoff between reliability and traffic overhead. We explore redundant routing approaches and present models that can be used to predict the reliability of an end-to-end transmission. Reliability is a mission-specific metric that evaluates the probability
that a packet will be delivered to the destination.

In Chapter 3, we present a routing approach, namely, diffuse pathset routing, aimed at increasing reliability of transmissions in failure prone networks. Geo-diffuse pathset routing or Petal Routing [11] uses the broadcast nature of wireless nodes to reduce the transmission overhead of redundant routing. The number of diffuse pathsets can be varied based on reliability requirements of the network. The approach can be considered a form of restricted flooding, which has the desirable characteristic that nodes do not have to maintain neighborhood information or end-to-end paths. Thereby, the control overhead is kept at a minimum since nodes do not have to exchange periodic beacon messages. We present an extended form of Petal routing with redundancy tuning, which can be used to reduce the number of intermediate transmissions by the use of a back-off timer. We believe that such a beacon-less routing approach can also be applied to some of the future Internet architectures, such as named data networks [9,31,60].

We compare Petal routing to a geo-diverse routing [50] approach, which can be used to guard against jammers in the network, by carrying out power control on geographically diverse paths from the source to the destination. We compare these approaches in Chapter 4 with existing multipath routing techniques and against one another to identify scenarios where one works better than the other.

Being able to predict the reliability of a given transmission can be used as feedback to the routing algorithm to tune its parameters as required. This would allow a sender to specify a relative degree of desired reliability a priori, and the routing algorithm can tune its parameters to achieve the target reliability. Based on geo-diffuse pathset routing, we present an analytical model that can predict the reliability of a given transmission. The analytical model takes input parameters such as network node density and the region of the network taking part in the transmission. To analyze all possible ways a transmission can fail, the approach uses minimal cutsets. Such a set provides the smallest number of ways in which a transmission can fail. We extend the set of minimal cutsets to account for all possible failure scenarios using other network parameters such as the overall link failure probability to model Petal routing closely. In this
study, we focus on establishing an analytical relationship between the routing parameters and the level of reliability achievable.

Quantifying reliability can in turn help design a model to analyze more general feedback-based wireless systems. To this end, we have worked jointly with control theory researchers. We have investigated the characteristics of ad hoc wireless network routing, to identify reachability-based predictors of bounds on network performance. The approach we use is to model networks using hybrid control theory to characterize them in terms of fundamental theory. We present a network model to represent a wireless multihop network as a stochastic dynamical system in Chapter 6. We use reachability analysis to compute bounds on the probability of a data packet reaching the destination within a given time. The system consists of a centralized controller, which we call an “oracle,” that computes optimal paths to the destination. As an idealized base case of the system, the oracle knows steady state availability of all links in the network as well as the up/down state of the links at the start of packet transmission. The links themselves are independent of each other, and their states vary either independently or follow a two-state Markov process. As more advanced and practical scenarios, we consider cases where the oracle may not have updated link states of the entire network. We compare the predictive results of our system with simulations of existing routing protocols such as AODV [51] and OLSR [17].

To represent wireless links in the stochastic dynamical system, we use two approaches, namely stationary link probabilities and two-state Markov link transition probabilities. The steady state or stationary probability of a link can be obtained from a network by observing the link over a long period of time and computing its packet delivery ratio. To obtain the Markov transition probability of a wireless link, one would have to observe the link at regular time intervals to note the state of the link relative to its previous state. The value of the transition probabilities in this case, can vary widely based on the time interval. Given the steady-state availability of a wireless link, it is not possible to predict the rate of fluctuations. A link with an overall high availability value, may be up for long periods of time followed by being down for long periods of time. On the other hand, it could also vary between up and down states
frequently, keeping the average availability a constant. In Chapter 5, we present our findings while measuring link states to calculate the timescale of failures in wireless links. This can help in designing more robust routing protocols that take account of the current state of a link and its expected states in future. Further, knowing the timescale of link failures can help distinguish between malicious nodes and failures due to the medium.

Our contributions in this research include: exploring some redundant routing approaches that can be used to achieve a certain level of reliability in failure prone networks. The goal is to use only as many resources as needed to deliver the required reliability; the “resources” being packet transmissions in this context. We also present some modeling techniques to predict the reliability of a given transmission. This can help achieve the following: (1) routing protocols can tune their parameters to ensure that the desired reliability is achieved, (2) existing routing approaches can be evaluated for reliability requirements and, (3) new routing approaches can be designed in order to overcome the limitations of existing approaches identified using the model. In an effort to design models that closely resemble practical wireless networks, we also carry out experiments on 802.11 links to gain an insight on link level fluctuations.
Chapter 2

Context

2.1 Problem Definition

We address the problem of routing packets reliably in wireless networks prone to failures and present modeling techniques to predict reliability of transmissions in this research. We first present a model that is applicable to networks using a specific routing methodology, namely Petal Routing. We then present a more general model that can be used to predict the highest achievable reliability for a given transmission. For both our redundant routing approaches and the models, we consider a network of wireless nodes located over a spatial region addressed by their geographic location. The nodes are capable of transmitting signals with a certain power, which defines the range of each node. Nodes that are within one another’s range form a wireless link and are said to be adjacent to one another. The “link” itself is over the wireless medium which is prone to failures.

Wireless multihop networks are susceptible to failures due to the medium [67]. Such failures are observed as intermittent link failures and may cause a disruption in the network. Malicious nodes, such as jammers, cause geographically correlated failures. Link failures, both random and correlated, affect packet transmissions and can reduce the reliability or packet delivery ratio between a given source and a destination in the network. The reliability of network transmissions
can be increased by sending information over multiple paths instead of one \cite{82}, and this is a common technique used in wired networks. In this research, we propose to use multipath routing in wireless ad hoc networks and present two approaches that can increase the reliability of transmissions in failure prone networks.

While designing redundancy based approaches, we utilize certain characteristic features of wireless networks. One such feature is the broadcast nature of wireless nodes. In wired networks, nodes are connected to one another by physical point-to-point links. Due to the absence of physical links in wireless networks, nodes simply transmit over the wireless medium when they have data packets to send. Consequently, all other nodes that are in the range can hear this packet. This characteristic of wireless networks has both pros and cons. The advantage is that if a node has to send a packet to several of its neighbors, it does not have to send the packet multiple times. Rather it can simply transmit the packet once and all its neighbors will be able to hear the packet. Our approach, presented in Chapter 3, makes use of the broadcast nature of wireless nodes. The disadvantage of this feature is that due to interference packet transmissions may get corrupted. Medium contention issues are handled in the MAC layer of wireless networks. The multipath approaches presented in this research are algorithms that can be implemented in the network layer.

One of the primary limiting factors of ad hoc networks is the battery life. Since power is limited, most ad hoc routing methods focus on saving power. Consumption of power can be reduced at different layers of the network. For example, the MAC layer can be designed to be power efficient, so that it *sleeps* when there is no activity in the network, and wakes up in time to transmit. Power efficient MAC techniques have been studied extensively in wireless sensor networks \cite{53,74,79}. In the network layer, the number of transmissions can be reduced by designing an efficient routing protocol. Routing schemes can make heuristic improvements to reduce the number of transmissions and thus increase the life of the network. Making use of the broadcast nature of wireless networks can also reduce the number of transmissions made by a wireless node, and thus save its battery life.
2.1.1 Redundancy Based Routing Approaches

Redundant routing can be used for a variety of reasons, such as to meet QoS requirements [30,47] or to improve reliability [35,40,80]. While sending information over multiple paths increases the reliability of the network, it also increases the number of transmissions, which in turn reduces operating time for battery-powered nodes [48]. Thus, there is a tradeoff between reliability and the number of transmissions. If a message is not required to be highly reliable, it does not need to be flooded over the entire network. A balance between reliability and the number of transmissions can be realized if these metrics are defined for each transmission and the network carries out only what is required [58].

We address the problem of delivering packets reliably in presence of link failures in a wireless multihop network. Since ad hoc and mesh networks can be densely populated, exchanging periodic updates can result in high control traffic overhead. Thus we design a beacon-less approach, where nodes do not have to exchange neighborhood information, while utilizing the broadcast nature of wireless nodes. The subsequent section describes the failure models that we consider.

Failure Model

Among other causes, failures in wireless networks could be due to the medium or due to malicious nodes [28,50].

- Intermittent Link Failure: This model captures independent link failures. Thus some links in the network fail, but there is no pattern by which the links fail. Isolated link failures are common in wireless networks due to the uncertainty of the medium. Such failures can also take place due to energy dissipation of nodes or localized environment effects [28].

- Jamming Model: The jamming model captures geographically correlated failures, or patterned failures. Such a failure, affects all wireless links in a circle of radius $R_p$. The choice of the circle is somewhat arbitrary, but it attempts to model radio wave propagation. In
reality, this model can be justified with the fact that environmental effects within a geographic region can cause correlated failures. We use the jamming model in [50]. We assume that a jammer can operate at any location in the network at a given time. The jamming power to signal ratio (JSR) at the receiver determines the degree to which jamming is successful. In general the JSR can be expressed as follows.

\[
JSR = \frac{\text{JammingPowerReceived}}{\text{SignalPowerReceived}} \tag{2.1}
\]

\[
JSR = \frac{P_J}{P_T} \left( \frac{D_{TR}}{D_{JR}} \right)^n \tag{2.2}
\]

where \( P_J \) is the jamming signal’s transmit power, \( P_T \) is signal’s transmit power, \( D_{TR} \) is the distance between the transmitter and receiver, \( D_{JR} \) is the distance between jammer and receiver and \( n \) is the path loss exponent. The above Equation 2.2 is based on the \( R^n \) propagation model [50].

### 2.1.2 Modeling Wireless Routing

In this section we discuss various modeling techniques presented in this research. The goal of all these techniques is to predict the reliability of packet transmissions in wireless networks. Predicted results from the models can be used to tune input parameters in routing protocols and to design routing approaches that are robust to failures in the network.

**Modeling redundant routing techniques**

We present an analytical model to predict the reliability of transmissions using redundant routing techniques presented in [11]. If a transmission needs to have a certain level of reliability, then given network parameters such as node density and the link failure probability, we design a technique to estimate the reliability of the transmission. If the estimated reliability is lower/higher than the required reliability, then the input parameters for the routing algorithm...
could be tuned accordingly in order to achieve the desired reliability. Further, we believe that analyzing such approaches to quantify reliability can in turn help design a model to analyze more general feedback-based wireless systems, which we address in the following section.

**Towards a general model for wireless routing**

Routing in wireless networks has a general feedback quality due to dynamics of the network. Host mobility and link fluctuations add to the dynamics. Routing parameters are often tuned based on periodic sensing and traffic estimates. We design a theoretical model for wireless networks as a stochastic dynamical system. We discuss several ways to model a wireless link using stationary availabilities and Markov transition probabilities.

In this study, we address the problem of how to model wireless routing as a controller, so as to obtain the highest reliability of packet transmissions achievable by any routing strategy. We present a probabilistic technique to maximize the network’s ability to deliver packets within a given deadline. We use reachability analysis as a tool to compute the probability of a packet reaching the destination node. Our goal in designing this approach is to predict the highest achievable reliability of a packet transmission. Although the “oracle” in our system computes optimal paths in the network to reliably deliver packets to a given destination node, we do not guarantee shortest paths using this approach. We propose to use this modeling technique to benchmark existing routing algorithms and possibly identify their limitations.

**2.2 Related Work**

We present existing work in the literature in two parts, namely, in the area of redundant routing and analytical modeling to predict routing performance.

**2.2.1 Redundant Routing in Wireless Networks**

Redundant routing in wireless networks has been addressed in the literature [16], in two broad categories: (a) sending redundant information using multiple paths [24,71], and (b) maintaining
one or more alternate paths for back-up [28, 39]. [28] was one of the first papers to propose multipath routing. In this method, multiple source-destination paths are constructed using two approaches: node disjoint paths and braided paths. With braided paths there are no completely disjoint paths, but many partially disjoint paths. Recent results show that braided paths are more efficient than node disjoint paths [63].

In [24], Dulman et al. present a network coding approach for multipath routing, where, even if transmissions along some of the disjoint paths fail, the destination is able to recover the packet using network coding. Some multipath techniques are extensions of single path routing schemes [43] and use the latter to find a set of end-to-end paths. Reliability of transmissions can be increased [80] with multipath routing in presence of network failures. However, the discovery and maintenance of multiple paths increase control overhead [6].

Location aided routing (LAR) [36] involves using geographic location to carry out routing. LAR or geographic routing techniques have been used to reduce message flooding in single path routing schemes [62] (e.g. beacon messages). Other approaches include SDAR [14] where the goal is to guarantee security and anonymity of transmissions. ExOR [8] uses node broadcasts where intermediate nodes communicate with one another to decide on the node that would carry the packet forward. Recent works on geographic routing include [37, 81].

A jammer’s presence cannot be concluded definitively by using simple statistics (e.g. energy on channel, carrier sensing time, packet delivery ratio) [77]. MAC parameters (e.g. contention window size) can be tweaked within the standard imposed limit [70] such that a compromised network node can consume much of the available bandwidth and starve other nodes without getting detected. Thus it is beneficial to study routing techniques such as [50], which provide protection specifically against jammers. Previously, power allocation techniques have been used to increase reliability [45], and multipath routing has been used to protect against jammers using a path’s availability history [44]. The approach presented in Chapter 4 combines the concept of multipath routing with power control.
2.2.2 Predicting Routing Performance

Reliability of network transmissions have been studied previously [5, 33, 46, 57] for wired networks. Many of these techniques use enumeration of cutsets to analyze failures in the network [5]. More recently, reliability of wireless networks has been studied in [2,19,29,34,66]. In graph theory, a cut is a partition of the vertices of a graph into two disjoint subsets. The cutset of the cut is the set of edges whose end points are in different subsets of the partition. Cutsets are commonly used in reliability analysis because any cut that partitions the source and the destination into two subsets, would cause a transmission failure. Thus enumerating all such cutsets can help enumerate all potential failure scenarios, thereby allowing one to calculate the reliability of a transmission. Our method of predicting reliability is also based on computation of minimal cutsets of the network. However, we tailor the approach to address specific characteristics of the diffused pathset algorithm proposed in [11].

Routing protocols in wireless multihop networks have been studied extensively [15,61]. We study the feasibility of using a control theoretic approach to model wireless multihop networks. To the best of our knowledge, this technique has not been used to model wireless networks before. We also provide a means of applying the concept of reachability analysis [1] to wireless multihop networks, in order to provide reliability guarantees for end-to-end transmissions. In the past, TCP congestion control and traffic flows have been modeled using hybrid control theory [38]. The dynamic behavior of routing algorithms has been studied in [41]. Linear programming formulations [42] have been used to compute optimal routing strategies to maximize the lifetime of wireless sensor networks. Our approach provides a way to obtain the maximum achievable reliability for a given transmission using a dynamic programming formulation. In addition, it provides routing information, so that this can be applied to network scenarios.

In our modeling approach presented in Chapter 6, we model wireless link availabilities using stationary probabilities as well as two-state Markov processes. We carry out link fluctuation measurements to validate the Markov modeling technique in Chapter 5. Link level measurements for 802.11 networks have been studied in [3] to compute loss rates in the second timescale.
Some popular link metrics have been shown in [21] and dynamics of these metrics have been presented in [20]. Some of the findings include the fact that broadcast based metrics are highly sensitive to background traffic and such metrics need to correct themselves based on background traffic. Wireless link burstiness for 802.15.4 networks has been studied in [68]. Our goal in this study is to vary timescales and observe fluctuations at different timescales in different wireless environments for 802.11 networks.
Chapter 3

Geo-Diffuse Pathsets

3.1 Problem Description

In this chapter, we address the problem of routing in failure prone wireless multihop networks. We use redundant routing as a tool to increase the reliability of end-to-end transmissions. Our approach, called Petal routing [11], is suitable for a network of nodes addressed by their geographic locations. Such an addressing scheme can be designed as an overlay on top of other addressing schemes such as IP addresses. We design this approach with several characteristic features of wireless networks in mind, namely, (1) nodes are broadcast based, (2) it costs more power to transmit than to sense the medium and thus transmissions should be only as many as required, (3) a beacon-less approach overcomes the overhead of exchanging periodic control packets. Our goal is to be able to tune the protocol parameters so as to achieve the desired level of reliability with the right amount of resources, which in this case is wireless transmissions.

3.2 Our Approach: Petal Routing

In this section we describe a routing technique that we call Petal Routing, which uses geographically diffuse pathsets. Petal Routing utilizes the broadcast nature of wireless networks to combine multiple transmissions from one node to a single transmission [16].
3.2.1 Basic Petal Routing

We first describe a basic protocol for Petal Routing, and then discuss an extension of it in Section 3.2.2.

Methodology

Given a source and destination, the network carries out constrained flooding to send the packet to the receiver. The flooding is constrained to transmissions within an area that we call a spatially diffuse pathset, or more intuitively as a “petal” (Fig. 3.1), because of the shape, the two ends of which converge at the source and destination. This is the general description of a petal; to apply this concept, some specific shape schema must be used. All nodes within the region defined as the petal aid in the transmission by broadcasting the packet. Intermediate nodes can compute whether they are located inside the petal, using their node location, and other information embedded in the packet header, namely, the source location, destination location and a petal parameter. This parameter quantifies the region of constrained flooding and depends on the specific shape schema being used. Intuitively, we call this parameter to be the ‘width’ of the petal. An example of a schema could be to use an ellipse to represent the petal, and the parameter in this case could be the minor axis of the ellipse, while the major axis would connect the source and destination. Further details on how an ellipse can be used to represent the petal can be found in [11].

When the source transmits a packet, it encapsulates the payload with petal headers. Figure 3.2 shows the structure of a petal packet. Point to point communication on a per hop basis cannot be realized in wireless networks. So, all nodes broadcast any packet that they are sending. When a node receives a packet, it needs to determine whether or not the packet was intended for it. If not, then it is an intermediate node in the transmission, and it now needs to determine whether it is inside the petal or not. If it is inside the petal, it transmits the packet and otherwise drops it. The basic algorithm followed by a node upon receiving a packet is given in Algorithm 1. To avoid flooding loops, all nodes store the IDs of recently broadcasted packets in an array.
that we call \textit{idList}. When a node receives a packet, it checks if it already sent out a packet with the same identifier. If it did, then the node drops the current packet. In our methodology, no individual node retransmits a packet.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure3.1.png}
\caption{Diffused overlapping pathsets}
\end{figure}

\begin{figure}
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
ID & \textit{S}_{loc} & \textit{D}_{loc} & \textit{T}_{loc} & \textit{W} & Payload \\
\hline
\end{tabular}
\caption{Structure of a Petal packet}
\end{figure}

The headers in a petalgram are as follows.

- Packet ID (\textit{ID}): a number that uniquely identifies a packet
- Source Location (\textit{S}_{loc}): co-ordinates of the source
- Destination Location (\textit{D}_{loc}): co-ordinates of the destination
- Transmitter Location (\textit{T}_{loc}): location co-ordinates of the previous node
- Petal Parameter (\textit{W}): this parameter specifies the \textit{width} of the petal

Algorithm 1 outlines the steps involved in Petal routing.
Algorithm 1 Algorithm for Basic Petal Routing

Obtain current node location co-ordinates: $P_{loc}$

Get petal headers: $ID, S_{loc}, D_{loc}, T_{loc}, W$

if $P_{loc} = D_{loc}$ then
    Destination has received packet
    Exit
else
    Use Formula (3.8) to determine if node $P$ is inside the petal
    if $P_{loc}$ is inside petal then
        if $idList[]$ contains ID then
            This packet was already transmitted by this node so drop packet
            Exit
        else
            Transmit the packet
            Add ID to $idList[]$
        end if
    else
        Drop packet
    end if
end if

This technique also allows for further enhancements to reduce the number of transmissions within the diffuse pathsets. Since the underlying protocol is flooding, the individual nodes do not need any prior information about their neighbors or maintain any end-to-end paths.
Assumptions

As in other geographic routing techniques [62], Petal Routing uses node locations to uniquely identify nodes. Messages are routed to the location of the destination rather than a network address. It is assumed that a node always knows its current location. Additionally, the source knows the location of the destination. Nodes do not need to maintain any routes to other nodes or know who their neighbors are. The location of a node determines whether or not it would take part in a certain transmission. The \((x, y)\) coordinate of a node is in the form: \((\text{longitude}, \text{latitude})\).

With regard to nodal density in the network, we assume that the network is reasonably dense. If no geographic source-destination path exists within the area of the petal, then it is up to the source to detect failure of transmission using end-to-end acknowledgement. However, this is a simplifying assumption, and in case of a sparsely populated network, our approach could be used in combination with table-driven approaches to store neighbor information, which can assist in calculating the optimal width of the petal.

In a densely populated wireless network, collisions are an important concern that can lead to poor performance. MAC layer algorithms for contention resolution in wireless networks have been studied extensively [32, 75, 78]. Moreover, the multihop path occurs through a time scale that is more coarse grained, and it does not affect the overall end-to-end performance. We assume that the contention resolution and scheduling issues at the hop time scale are handled by the MAC layer. In the network layer, we provide some heuristic enhancements using back-off time, to reduce the probability of collisions.

We assume low mobility in the network. Thus, the source is always assumed to have an updated location of the destination. To allow for minor movement of nodes, we consider the destination to be within a small radius of its most updated location that the source has. The mobility of intermediate nodes is not of particular concern. This is because the location of a node at a certain instant of time determines whether or not it is in the petal. No pre-determined assumptions are made on the locations of the source itself or the intermediate nodes.
We also assume that the nodes in the network are capable of omnidirectional transmission. There are no special requirements on directional transmission for Petal Routing. In addition to that, we assume that all nodes in the network are roughly identical and have similar transmission range. Thus, if the wireless network were to be converted into a directed graph, all links would be unidirectional. This does not affect the correctness of our protocol, but it improves the efficiency of the back-off mechanism described in Section 3.2.2. With these assumptions in consideration, we present the failure model that is used for testing the Petal Routing protocol.

**Implementation**

When a node receives a packet, it needs to determine whether or not it is inside the petal defined by the packet headers. Consider the source destination pair in Figure 3.3. The source is denoted by the co-ordinates \((x_s, y_s)\) and the destination by \((x_d, y_d)\). Let \(P(x_p, y_p)\) be a point. Given the co-ordinates of the source and the destination, as well as the width \(w\) of the petal, our goal is to determine whether the point \(P\) lies inside the petal or not. This calculation is done as follows.

![Figure 3.3: Calculating the different petal parameters](image)
Let,

\[ a = x_d - x_s \]  \hspace{1cm} (3.1) \\
\[ b = y_d - y_s \]  \hspace{1cm} (3.2) \\
\[ c = \sqrt{a^2 + b^2} \]  \hspace{1cm} (3.3) \\
\[ m = b/a \]  \hspace{1cm} (3.4)

Then, the equation of the line though \( S(x_s, y_s) \) and \( D(x_d, y_d) \) is:

\[ (y - y_s) = m \times (x - x_s) \]  \hspace{1cm} (3.5)

Distance of point \( P(x_p, y_p) \) from the line in equation (3.5) is:

\[ d = \frac{|a(y_s - y_p) - b(x_s - x_p)|}{c} \]  \hspace{1cm} (3.6)

Position along the line in equation (3.5) where the perpendicular from point \( P(x_p, y_p) \) intersects:

\[ e = \frac{a(x_p - x_s) + b(y_p - y_s)}{c} \]  \hspace{1cm} (3.7)

Having obtained the point \( e \) along the \( S - D \) line, we design a threshold function that calculates \( t(e) \), the maximum distance from the \( S - D \) line to the perimeter of the petal from point \( e \). The formula used to calculate \( t(e) \) could depend on the shape of the petal used. For instance, instead of a petal if we were to use a rectangle shape, then the value of \( t(e) \) would be a constant. In our implementation, we use the shape of an ellipse as a close approximation of a petal. Using an ellipse, the value of \( t(e) \) at point \( e \) can be calculated using Formula (3.8).

\[ t(e) = \begin{cases} 
0 & \text{if } \delta = 0 \\
\sqrt{\delta + \beta^2} & \text{otherwise}
\end{cases} \]  \hspace{1cm} (3.8)
where $\delta = 1 - \left( (e - \frac{c}{2})^2 / \alpha^2 \right)$; $\beta = w/2$; $\alpha = c/2$.

### 3.2.2 Petal Routing with Redundancy Tuning

The basic technique for Petal routing was presented in the previous section. However, it was observed that not all nodes inside the petal need to transmit the packet. Some nodes inside the petal can cancel transmission without affecting the overall reliability. Consider the set of nodes in Figure 3.4 where $t_{\text{backoff}}$ of node B is higher than that of A and C. Nodes D, E, F and G receive the packet during B’s back-off time. Supposing that B hears transmissions from D, E, and F, if the value of $k \geq 3$ then node B would drop the packet. In this case, clearly, node B would not be able to reach any new node even if it transmitted, so the back-off time is successfully able to reduce the number of transmissions. To realize this concept, we present the idea of a backoff and a technique to tune the redundancy of intermediate transmissions.

![Figure 3.4: Back-off time to reduce transmissions](image)

Figure 3.4: Back-off time to reduce transmissions
Backoff

Petal Routing provides a back-off mechanism to reduce the number of transmissions within the petal. The number of transmissions is directly related to the probability of collisions, so this technique also reduces medium collisions. Note that the backoff mechanism is not designed to eliminate medium collisions, and MAC layer scheduling algorithms are necessary to avoid medium collisions. The main purpose of this technique is to reduce the number of intermediate transmissions. In general, we assume that the time frame for a petal transmission is greater than the MAC layer time frame.

Within a petal, all nodes do not need to transmit for the packet to reach the destination. This is more pertinent, if all successors of an intermediate node have already received the packet. When a node receives a packet and finds itself to be inside the petal, a back-off time $t_b$ can be introduced. The node can be made to back-off $t_b$ milliseconds, before it deciding whether to transmit or not. If, within this time, it can hear transmissions from $k$ nodes, then it decides to drop the packet, otherwise it transmits. Note that the value of $t_b$ can vary for individual nodes based on their location with respect to the petal. Knowledge of the node density allows a node to compute the expected value of $k$, by considering the area represented by the nodes from which transmissions were heard: this has been described in Section 3.2.2.

The value of $t_b$ can be selected based on the delay requirements. While the back-off time may reduce the number of transmissions in the network, it may lead to high delays if its value is very large. Thus, there is a trade-off between the delay and the number of transmissions. However, the reliability of transmission is not affected by introducing back-off. This is because of the fact that even if the node backs off for a longer period of time, it would eventually transmit the packet if its neighbors did not receive it. This would cause a greater delay, but would not affect the reliability of transmission. Three methods for selecting $t_b$ are as follows.

- **Random Backoff**: A simple technique to select $t_b$ could be choose a random number (with some pre-defined upper bound). This would ensure that different nodes back-off
for different periods of time. Although there is no pattern in the back-off values of the different nodes, it would still alleviate some medium contention issues.

- **Coordinated Back-off**: This method allows selection of $t_b$ based on a node’s location with respect to the petal. Given a petal of width $w$, we would like to choose $t_{backoff}$ in such a way, that the nodes located towards center of the petal have a higher probability of transmitting the packet first. Thus, these nodes would have a smaller $t_{backoff}$, while the nodes located near the sides of the petal (or the perimeter of the petal) have a higher value of $t_{backoff}$. In Figure 3.5, the green nodes are closer to the line joining the source (S) and destination (D), and have lower values of $t_{backoff}$ compared to the orange nodes.

Varying the back-off time based on node locations, we consider that $t_{backoff}$ is least along the S-D line ($t_{lb}$), and it uniformly increases to reach an upper bound ($t_{ub}$) at the edge of the petal. Thus, when the nodes along the central line transmit first, some of the nodes towards the edges may not have to transmit if they are able to hear transmissions from all the downstream neighbors. This phenomenon is illustrated in Figure 3.6(a). It should be noted that the curve in Figure 3.6(a) does not represent actual back-off values. Instead, if the back-off value of the curve at a certain location is $t_{loc}$, we choose $t_{backoff}$ such that, $0 \leq t_{backoff} \leq t_{loc}$. In our current implementation $t_{lb}$ is not varied along the backbone of the petal. However, varying the lower bound from source to destination could also be a possible method to choose backoff time.

- **Randomized Coordinated Back-off**: In coordinated backoff, if two nodes are located exactly the same distance away from the backbone of the petal (perpendicular to the backbone), both nodes would choose the same backoff time using coordinated backoff. This would reduce the effect of backoff, because both these nodes would wait the same period of time to heard for downstream transmissions. To alleviate this problem, we use a third method to choose backoff time, namely, randomized coordinated backoff. This method involves the same steps as in Coordinated Backoff. However, instead of selecting the value on the
curve (shown in Figure 3.6) at a certain location, a random number between $t_{lb}$ and the value on the curve is selected. This allows for more randomness, rather than choosing a specific value based on the curve and solves the problem of nodes choosing symmetric backoff time.

**Algorithm for Petal Routing with Backoff**

With the introduction of backoff time, the steps for Petal Routing is presented in Algorithm 2.

**Redundancy Tuning with Convex Hull**

After the backoff timer expires, a node has to decide whether it would cancel the transmission or not. There can be many strategies for this decision algorithm, one of which could be using a convex hull. While a node is backing off, it keeps track of node locations that transmit the same packet. In this decision algorithm, the current node creates a convex hull of all other nodes it heard the packet from. A convex hull of a set $X$ of points is the smallest convex set that contains $X$. Several techniques exist for generating a convex hull of given points in $O(n \log n)$ time.

There are three cases that may occur with respect to the convex hull and the current node A. If A is inside the hull as shown in Figure 3.7, it transmits the packet. If A is outside the
hull and the hull is towards the source (Fig. 3.9) it transmits the packet. If A is outside the hull and the hull is towards the destination (Fig. 3.8), it cancels the transmission. The only case in which A cancels the transmission is when sufficient downstream nodes have heard the transmission. This is a heuristic technique, and does not yield optimal results.

Figure 3.7: Convex Hull Case 1
Algorithm 2 Algorithm for Petal Routing with Backoff

Obtain current node location co-ordinates: $P_{loc}$
Get petal headers: $ID, S_{loc}, D_{loc}, W, S_t, L_{hop}$

if $P_{loc} = D_{loc}$ then
    Destination has received packet
    Exit
else
    Use Formula (3.8) to determine if node P is inside the petal
    if $P_{loc}$ is inside petal then
        if $idList[]$ contains ID then
            This packet was already transmitted by this node so drop packet
            Exit
        else
            Add ID to $idList[]$
            Choose back-off time $t_b$
            Add packet to waiting buffer $B_{wait}$
        end if
    else
        Drop packet
    end if
end if
Redundancy Tuning with Cancellation Score

When the back-off timer expires, the node knows the number of neighboring nodes from which it heard the packet as well as the location coordinates of the neighbors. Using this information, the node computes a value, that we call the 'cancellation score' or \( S_c \). Intuitively, cancellation score is a number that indicates how aggressively a node should cancel its transmission and vice versa. A high cancellation score indicates that the probability of transmission should be very low. This section provides methods to compute the cancellation score and map these values to transmission probability ranging from 0 to 1.

Figure 3.10a illustrates a sample scenario after a backoff timer expires. Node \( P \) is the current node, while \( C \) is the location of the weighted centroid of all the neighboring nodes \( P \) heard from while it was backing off. \(|PP'|\) is the perpendicular distance of \( P \) from the \( SD \) line, while \(|CC'|\) is the perpendicular distance of \( C \) from the \( SD \) line. Based on our understanding of the parameters, we compute the cancellation score using Formula (3.9).

\[
S_c = k \ast \left( \frac{|PD|}{|SD|} \right) \ast |PP'| \ast |CC'| \ast |PC| \ast \text{neighborCount} \tag{3.9}
\]

where \( k \) is a constant, \( \text{neighborCount} \) is the number of nodes \( P \) heard from and the remaining parameters are consistent with Figure 3.10a.

The justification for using Formula (3.9) is as follows. The geographic source destination line is closest to the shortest path between \( S \) and \( D \). Transmissions close to the source destination line should therefore be canceled rarely. Transmissions closer the perimeter of the petal, should however be avoided because the perimeter represents a much longer path to the destination. These notions are captured in the parameters \(|PP'|\) and \(|CC'|\), which indicate the deviation of points \( P \) and \( C \) from the \( SD \) line. Note that this preference is preceded by the number of nodes the current node heard from, i.e. \( \text{neighborCount} \). If \( \text{neighborCount} \) is very low, the cancellation score would also be low, and the node would be more likely to transmit regardless.
of its deviation from the \(SD\) line.

Transmissions closer to the destination should be canceled less aggressively so that the overall transmission has a higher likelihood of succeeding. This is represented using the parameter \(\frac{|PD|}{|SD|}\). The closer a node is from the destination, the lower the cancellation score would be. Lastly, the term \(|PC|\) indicates how far the centroid is from node \(P\). It also captures whether \(C\) is closer to the destination than \(P\). If \(C\) is further away from the destination than \(P\) we assign this value to be negative. Note that the further away from and behind \(P\) the centroid \(C\) is, the lower the cancellation score is.

![Graph showing calculation of cancellation score and sigmoid curve](image)

Figure 3.10: (a) Calculation of cancellation score \(S_c\); (b) Inverse sigmoid curve

In order to map the cancellation scores to probability values, we use an inverse sigmoid curve shown in Figure 3.10b with the pivot at \(x = 0\). According to the inverse sigmoid curve, the higher the cancellation score is, the lower the probability of transmission is and vice versa. Note that for negative values, \(P_{\text{trans}}\) ranges from 0.5 to 1, i.e., if the centroid is behind \(P\) the node is highly likely to transmit the packet because it can reach more nodes in front of it. The formula for the sigmoid curve is as follows:

\[
P_{\text{trans}} = \frac{1}{1 + e^x}
\]  

(3.10)
where \( x \) is the cancellation score, or the x-axis of the curve from Figure 3.10b.

To validate the formula for calculating the cancellation score, we generated a 3-dimensional plot of the node locations with the height at each point as the probability of transmission. For the purpose of this study, we used a mean value of \( \text{neighborCount} \) obtained from simulation for all the nodes. The centroid of the neighbors, \( C \), was always placed a mean distance in front of node \( P \) parallel to the SD line. From Figure 3.11, it can be seen that probability of transmission is higher near the destination (green node), and is the highest along the SD line. Both of these, were notions which we tried to achieve using the cancellation score.

Note that the probability values obtained in Figure 3.11 range from 0 to 0.5. This is because the centroid \( C \) was always placed towards the destination for this study. If \( C \) is behind the destination or closer to the source, then the term \(|PC|\) in formula (3.9) becomes negative, which in turn gets mapped onto the negative X-axis of the sigmoid curve from Figure 3.10(b). As can be seen from the sigmoid curve the negative X-axis has values between 0.5 to 1.0. The logical significance of choosing such values is that if the centroid \( C \) is behind the current node, then it should transmit with a higher probability than if \( C \) is in front of the current node \( P \).
Figure 3.11: Probability of transmission for different node locations

Figure 3.12 shows visualizations of simulations where nodes canceled transmissions. For this figure, we tuned up the cancellation score by squaring it, so as to observe which nodes are most likely to transmit. As expected, the nodes along the SD line and closer to the destination transmit the most.
Results

We evaluated petal routing using extensive OPNET simulations. The nodes were generated randomly using a 2-dimensional Poisson distribution. We compare the three back-off techniques in Fig. 3.13. The upper bound for back-off is increased from 2 milliseconds to 10 milliseconds in steps of 2. It can be seen that the delay is consistently least for randomized coordinated back-off, as expected.

The objective of back-off is to reduce collisions and therefore repeated transmissions, wasteful of time and energy. In Fig. 3.14, we see that for all three techniques the total number of transmissions reduces as the back-off time is increased, thereby proving the usefulness of back-off. The three dimensional plot in Fig. 3.15 shows the interplay between expected reliability, petal width and node failure probability; as the failure probability increases the reliability goes down, but can be improved by widening the petal.

We also compared the results of Petal Routing with an existing network coding scheme [24]. The basic approach of the network coding technique is to find $k$ node disjoint paths from source to destination and then send partially overlapping information along the $k$ paths. If at least $E_k$ packets are received by the destination (where $E_k \leq k$), they can be used to reconstruct the original packet. In our implementation of the network coding approach, disjoint paths are computed externally instead of during simulation. This favors the network coding technique,
3.3 Analytical Model for Basic Petal Routing

We present a procedure to predict the reliability of an end-to-end transmission [10]. In this context, we define reliability to be the packet delivery ratio. This predictive procedure is based
3.3.1 Algorithm for Predicting Reliability

The analytical model for Petal Routing takes a set of input parameters and calculates the expected reliability of the transmission. Input parameters include network node density, range of each node, overall link failure probability, width of the petal and distance between the source and destination. The basic idea behind the analytical model is as follows. An end-to-end transmission can fail only if a combination of paths fails such that it converts the petal into a disconnected graph. Note that any cutset of the petal would cause a transmission failure. In graph theory, a cut is a partition of the vertices of a graph into disjoint subsets. A cutset is
Figure 3.15: Reliability vs. Failure probability vs. Petal width for Petal Routing, 150 node network, Back-off technique = Coordinated, $t_{lb} = 0 \text{ ms}$, $t_{ub} = 5 \text{ ms}$, Aggressiveness = 3
Figure 3.16: Reliability vs. Jamming Power (Petal Routing and Network Coding), 100 node network, Petal Width = 0.01, Aggressiveness = 3, Back-off technique = Coordinated, $t_{lb} = 0$ ms, $t_{ub} = 10$ ms, Network Coding $k = 4$, $E_k = 2$

defined as the set of edges whose end points are in different subsets of the partition. To calculate all possible ways in which a transmission would fail, one would have to compute all cutsets of a given network. We present a technique that starts with all possible minimal cutsets of the petal. A cutset is said to be a minimal cutset if, when any edge is removed from the set, the remaining edges are collectively no longer a cutset. Computation of minimal cutsets is discussed in Appendix A.

The use of cutsets in estimating reliability of a network is fairly common in the literature [46, 57]. We specifically use minimal cutsets, because such a set provides the smallest number of ways in which a transmission can fail. The approach provided in this study however, uses the computed minimal cutsets along with other network parameters such as the overall link failure probability to model Petal routing closely. We consider each minimal cutset to be a
family of failures and then try to enumerate the number of actual failures scenarios each such set represents. A minimal cutset can represent multiple failure scenarios provided the size of the cutset is smaller than the expected number of failed links in the network. We first compute all minimal cutsets for a given petal. Let this be denoted by \( C = \{c_1, c_2, ..., c_k\} \), where there are \( k \) minimal cutsets.

The expected number of failed links, \( E(e_{\text{fail}}) \), can be computed from link failure probability of the network, node density, distance between source and destination and width of the petal as shown in Algorithm 3. If the size of the minimal cutset is greater than the expected failed links then the cutset is discarded. Otherwise, we calculate the remaining links that would fail in the given scenario, \( E(e'_{\text{fail}}) \). It is to be noted that all other link failures would have to be before the cut, i.e., in the partitioned set that contains the source node. Using combination, we can calculate the number of different ways in which the remaining nodes can fail, out of the nodes located before the cut. This number represents the expected number of failure scenarios for the cutset under consideration. For each minimal cutset, the probability of this combination of links failing can be calculated using the formula,

\[
p_{\text{fail}} = (p_l)^f \ast (1 - p_l)^s
\]  

(3.11)

where \( p_l \) is the link failure probability, \( f \) is the number of failed links and \( s \) is the number of successful links.

Formula (3.11) can be extended for all possible failure scenarios represented by this cutset as using the formula,

\[
p_{c_i} = E(c_{\text{fail}}) \ast (p_l)^f \ast (1 - p_l)^s
\]  

(3.12)

where \( E(c_{\text{fail}}) \) represents the number of ways in which this minimal cutset could cause a failure scenario.

The value of \( p_{c_i} \) for each minimal cutset is summed up, and finally, the reliability of the
transmission can be calculated as,

\[ \text{Reliability} = 1 - \sum_{i=1}^{k} p_{ci} \quad (3.13) \]

The complete set of steps to calculate the reliability of a given transmission can be found in Algorithm 3.

**Algorithm 3** Calculate reliability for basic Petal routing

Inputs: Graph \( G(v, l) \) (consisting of all nodes inside the petal), link failure probability \( (p_l) \), source location \( (S) \), destination location \( (D) \)

\[ \text{max}(c) = \text{floor}(p_l \cdot l) \]
Compute all minimal cutsets, \( C = \{c_1, c_2, ..., c_k\} \), for \( k \) cutsets

\[ \begin{align*}
\text{for } i = 1 \rightarrow k \text{ do} & \\
\text{if } |c_i| \leq \text{max}(c) \text{ then} & \\
& e = \text{number of edges before the cut} \\
& E(e_{\text{fail}}) = \text{floor}(p_l \cdot (e + |c_i|)) \text{ //Expected failed links} \\
& \text{if } |c_i| \leq E(e_{\text{fail}}) \text{ then} \\
& E(e'_{\text{fail}}) = E(e_{\text{fail}}) - |c_i| \text{ // Remaining links to fail} \\
& p_{ci} = \frac{C_{E(e'_{\text{fail}})} \cdot (p_l)^{E(e_{\text{fail}})} \cdot (1 - p_l)^{(e + |c_i| - E(e_{\text{fail}}))}}{} \\
& \text{end if} \\
& \text{end if} \\
\text{end for} & \\
\text{end if} \\
\text{end for} & \\
p_c = \sum_{i=1}^{k} p_{ci} \\
\text{Reliability} = 1 - p_c
\]

In the above algorithm, we assume that the number of failures in the network is large enough to have a significant and representative effect on the links inside the petal. In other words, we assume that the overall link failure probability of the network \( (p_l) \) is consistent for a localized region in the network, i.e., inside the petal. Note that the expected number of failed links \( (E(e_{\text{fail}})) \) is an approximation of the actual number of links that can fail in a given scenario.

We first carry out experiments with the assumption that exactly \( E(e_{\text{fail}}) \) links fail for a given
cut. Later, we relax this approximation and carry out further tests. Lastly, while calculating $\text{max}(c)$ and $\text{E}(e_{\text{fail}})$, we compute the floor of the fractional value. This introduces some error in our reliability predictions, but is necessary in order to compute the combinatorial value ($\binom{e_{\text{fail}}}{\text{C}}$) further down in the algorithm. By computing the floor of these values, we are not biasing the reliability predictions. On the contrary if the number of failures for a given scenario is less due to the floor the final reliability prediction will be higher, thereby increasing the difference between the simulation results and that of the analytical model.

### 3.3.2 Generating Topologies for Comparison

Since the exact topology of the network is not known to the analytical model, there are several ways to characterize the topology. For example, given the node density, range of each node, locations of the source and destination, and width of the petal, we can generate a sample topology that satisfies these parameters. In addition, we use clustering coefficient to characterize topologies. Even with networks having similar characteristics such as node density, area and clustering coefficient, the actual topology may vary greatly from the generated topology. Therefore, we generate multiple such topologies and obtain the mean of the reliability values calculated for each topology as the overall reliability of the transmission.

We first carry out tests by providing actual topologies to the analytical. As a more practical approach, we design an algorithm that can generate a sample topology given the node density and area of a network using a uniform random number generator. Since all our comparisons are with OPNET simulation, we try to make the generated topologies as close to the ones generated by OPNET. This is done by studying the clusteredness of the OPNET topologies and applying the same to our topology generation algorithm. As an alternative to using an algorithm to generate topologies, we use sample topologies from OPNET (but not the actual topology), to compare results of simulation with the analytical model. Note that this is different from providing the actual topology - in this case we provide many OPNET generated topologies with the same node density to our model, but not the actual one on which the simulation runs.
This ensures that we have a fair comparison between the simulation results and the ones from the model, without providing specifics to the latter. The results for these different input topology methods are shown in Section 3.3.3. In this section we describe the method for obtaining the clustering coefficient of a network.

There are practical implications of abstracting the actual topology from the analytical model. For example, consider a tactical scenario where the number of soldiers in a certain area is known, but their exact location is not known. Note that in this case, it is impractical to also abstract the location of the source and destination since reliability is a mission oriented metric, and depends strongly on the source destination pair involved in a given transmission.

Clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together [64]. According to Watts and Strogatz [76] the clustering coefficient of a network is the average of the local clustering coefficients of all nodes, as shown in Formula (3.14).

\[
\bar{C} = \frac{1}{n} \sum_{i=1}^{n} C_{i} \tag{3.14}
\]

The local clustering coefficient of a node in a graph quantifies how close its neighbors are to being a clique (complete graph). A graph \( G = (V, E) \) formally consists of a set of vertices \( V \) and a set of edges \( E \) between them. An edge \( e_{ij} \) connects vertex \( v_{i} \) with vertex \( v_{j} \).

The neighborhood \( N_{i} \) for a vertex \( v_{i} \) is defined as its immediately connected neighbors as follows:

\[
N_{i} = \{ v_{j} : e_{ij} \in E, e_{ji} \in E \} \tag{3.15}
\]

Let \( k_{i} \) be the number of vertices (\(|N_{i}|\)) in the neighborhood, \( N_{i} \), of a vertex.

The local clustering coefficient \( C_{i} \) for a vertex \( v_{i} \) is then given by the proportion of links between the vertices within its neighborhood divided by the maximum number of links that could exist between them. For a directed graph, \( e_{ij} \) is distinct from \( e_{ji} \), and therefore for each neighborhood \( N_{i} \) there are \( \binom{k_{i}}{2} \) or \( k_{i}(k_{i} - 1) \) links that could exist among the vertices within

38
the neighborhood \(k_i\) is the number of neighbors of a vertex).

The local clustering coefficient for directed graphs is given as [76]:

\[
C_i = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)}
\]  

An undirected graph has the property that \(e_{ij}\) and \(e_{ji}\) are considered identical. Therefore, if a vertex \(v_i\) has \(k_i\) neighbors, \((k_i(k_i - 1))/2\) edges could exist among the vertices within the neighborhood. Thus, the local clustering coefficient for undirected graphs is defined as:

\[
C_i = \frac{2|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)}
\]  

Using a uniform random number generator, we obtained network topologies such as the one in Figure 3.17. In OPNET, we used the Wireless Network Deployment tool with a random topology, to obtain networks such as the one shown in Figure 3.18. The clustering coefficients of the generated topologies and the ones from OPNET were both 0.67. So it was verified that by using a uniform random number generator, we were able to generate networks with characteristics similar to the ones in OPNET. Note that the Wireless Network Deployment tool in OPNET has various random number generators. Using a different random number generator in OPNET results in that are less clustered. However, since we were not able to verify the function used to generate these topologies, our comparative plots with OPNET are with uniform random generators. In some cases, where the exact topology was provided to the analytical model we used other random number generators in OPNET to verify the behavior of the analytical model with different topologies. In the following section we present comparative results obtained from our analytical model and from OPNET simulations.

3.3.3 Results

In this section, we present comparative results between the analytical model and OPNET simulations. We implemented Petal Routing over the IP layer using the wlan_station_adv model
Figure 3.17: Network topology generated with uniform random distribution in OPNET. Figure 3.19 shows the node model for our implementation of Petal routing. As mentioned before, node locations are used to uniquely identify nodes as opposed to network addresses. However, this technique can coexist with other addressing schemes such as IP addresses.
Figure 3.18: Sample network topology from OPNET simulation with points plotted in 2D space on the right

Figure 3.19: Node model for Petal Routing
Figure 3.20 shows a plot of reliability versus increasing petal width for OPNET simulations and the analytical model for a 200 node network. It can be seen that the analytical model provides an upper bound on the reliability value. Note that in this case the analytical model was provided with the exact topology from simulations and we used a perturbed grid topology in OPNET.

Figure 3.20: Reliability vs. Petal Width for Simulations and Analytical Model. Topology provided to analytical model

Figure 3.21 shows the same plot using the minimal cut algorithm from Appendix A.3. We
were able to obtain many more minimal cutsets using Algorithm 5, and the reliability values in this case were much closer to those obtained from simulations as a result. In this figure it also can be observed that the reliability values are close to the ones from simulation for lower widths of the petal, where the number of minimal cuts was less. We from Figures 3.20 and 3.21 it can be concluded that the algorithm being used for enumerating failure modes or minimal cuts is important in estimating how any diverse path mechanism will perform.

![Reliability vs. Petal Width (meters)](image)

Figure 3.21: Reliability vs. Petal Width for Simulations and Analytical Model. Topology provided to analytical model, using Algo: 5
Figure 3.22 show similar comparisons using 100 node clustered topologies. In Figure 3.22, no topologies were provided to the analytical model, and we used the mincut enumeration method from Appendix A and A.2. The model calculated reliability values for generated topologies and computed the mean value for a given width of the petal.

Figure 3.22: Reliability vs. Petal Width for Simulations and Analytical Model. Generated topologies used in analytical model

Figure 3.23 shows a comparison of different 150 node clustered topologies generated using a uniform random number generator and the minimal cutset method from A.3. We show three variations of topologies in the analytical model, (1) by providing the exact topology, (2) by using
generated topologies and (3) using sample topologies generated by OPNET. The reliability values obtained by generated topologies were closer to simulations than sample topologies from OPNET. This may be because the sample set from OPNET was not as representative.

Figure 3.23: Reliability vs. Petal Width for Simulations and Analytical Model. Different topologies with Algo: 5
Another technique employed in addition to enumerating minimal cuts, was using binomials. In a practical scenario, suppose that the expected number of failed links is 15. There is no guarantee that exactly 15 links would fail. The actual links that fail could be 13 or 16. There could be higher or lower values of the actual number, each with a diminishing probability. To incorporate this into the analytical model, for each minimal cut, we computed the failure probability considering the expected failed links to be the number obtained from Algorithm 3 $E(e_{fail}) \pm 1$ and $\pm 2$. We call the former to be ‘1 Binomial’ and the latter to be ‘2 Binomials’. The failure probability of the cut $(p_{ci})$ was then considered to be the mean of the values obtained using $E(e_{fail})$, $E(e_{fail}) \pm 1$ and $E(e_{fail}) \pm 2$.

Figure 3.24 shows a plot using binomial values of $E(e_{fail})$ on a 200-node perturbed grid topology that was provided to the analytical model, and Figure 3.25 shows the same plot using all possible values of $E(e_{fail})$ (i.e. from $|c_i|$ to $(e + |c_i|)$).
Figure 3.24: Reliability vs. Petal Width for Simulations and Analytical Model, topology provided to simulation, with binomials
Figure 3.25: Reliability vs. Petal Width for Simulations and Analytical Model, topology provided to simulation, with all binomials

Figure 3.26 shows the plot with binomials for 150 node clustered topologies that were not provided to the analytical model. In each case, it can be observed that using the binomials of $E(e_{fail})$ results in reliability values closer to the ones obtained from simulations.
Figure 3.26: Reliability vs. Petal Width for Simulations and Analytical Model, clustered topology not provided to simulation, with binomials.
3.4 Analytical Model for Petal Routing with Redundancy Tuning

In this section we present an analytical model for the extended version of Petal Routing with redundancy tuning [10]. In Section 3.2.2 we presented a strategy to cancel transmissions constructing a convex hull. This decision algorithm can have different strategies. In the following section, we first present an alternate strategy to cancel transmissions, compare it with the convex hull method, and then present an analytical model for the same.

Comparing Cancellation Strategies

Figure 3.27 shows a comparative plot of reliability versus backoff using convex hull and cancellation score. From the plot, it can be seen that there is no strong correlation between increasing backoff time and the reliability of transmission. The cancellation score method does not reduce the reliability. In addition to that, it is computationally less expensive to use the cancellation score method. Therefore, we use this method in our analytical model.
Figure 3.28 shows a comparison between the number of intermediate transmissions using the convex hull method and the cancellation score strategy. It should be noted that using the cancellation score method, the number of transmissions can be varied by changing the value of the constant or moving the pivot of the sigmoid curve.
Using Cancellation Score to Visualize the Petal

In Figure 3.11, the transmission probability falls sharply outside the petal, because nodes located outside petal never compute the cancellation score and always drop the packet. An alternate way
to implement constrained flooding could be by varying the curve obtained using cancellation scores. For example, if nodes outside the petal also computed cancellation scores and probability of transmission, we would obtain a curve such as the one in Figure 3.29.

There could be various ways to control the curve obtained using cancellation score. Firstly, the constant $k$ gives a direct handle on the cancellation score. However, this value is mapped to probability values ranging from 0 to 1, therefore any change in the transmission probability values would have to be controlled using the sigmoid curve. With regard to the sigmoid curve, the pivot point could be modified. Other than that, the inverse sigmoid function itself could be changed to get different degradation characteristics of the curve.
3.4.1 Algorithm for Predicting Reliability

For fine grained results from the analytical model, we propose to vary the failure probability of a link based on a node’s location with respect to the petal. The degree of variation can be directly based on the node cancellation behavior observed in Section 3.2.2. The nodes closer to the \( SD \) line and closer to the destination should have a lower link failure probability as compared to nodes closer to the perimeter of the petal and closer to the source. Based on the
maximum backoff value, we can obtain the average neighborCount and average distance $|PC|$ for a network with certain node density, from simulations. These values are provided as input to the analytical model.

We use the cancellation score formula from Section 3.2.2, to directly vary the failure probability of a link. The basic idea behind the analytical model with backoff is as follows. Since the nodes in the petal now backoff and can cancel or transmit based on neighbors heard from, we incorporate this concept into the analytical model. We compute the cancellation score of the starting and ending nodes ($S_{c_1}$ and $S_{c_2}$) of each link using Formulae (3.9) where the notations are same as before. We then compute the mean transmission probability of using the sigmoid function from Formula (3.10). To obtain the cancellation probability for the link $p_{ej}$ we subtract the transmission probability from 1. $p_{ej}$ is the failure probability for link $e_j$ in addition to the overall link failure probability $p_l$. Note that $p_{ej}$ would be minimum along the backbone of the petal and maximum along the perimeter.

We compute $p_{ej}$ for each link of a given minimal cut. The probability of the links of the minimal cut failing $p_e$, is then given by:

$$p_e = \prod_{j=1}^{m} \frac{(p_{ej} + p_l)}{2}$$

(3.18)

In the above formula, the probability of a link in the minimal cut is weighted based on the cancellation score obtained from its location inside the petal. To calculate the failure probability of a cut $p_{ci}$, we use the following formula:

$$p_{ci} = \mathcal{C}_{E(e_{fail})} * (p_l)^{(E(e_{fail}) - |c_i|)} * (1 - p_l)^{(e + |c_i| - E(e_{fail}))} * (p_e)$$

(3.19)

The complete set of steps to calculate the reliability with backoff can be found in Algorithm 4.
Algorithm 4 Calculate reliability for Petal routing with Backoff

Inputs: Graph $G(v, l)$ (consisting of all nodes inside the petal), link failure probability ($p_l$), source location $(S)$, destination location $(D)$, average distance $|PC|$, average $neighborCount$

$max(c) = floor(p_l \times l)$
Compute all minimal cutsets, $C = \{c_1, c_2, ..., c_k\}$, for $k$ cutsets

for $i = 1 \rightarrow k$ do
  if $|c_i| \leq max(c)$ then
    $e =$ number of edges before the cut
    $E(e_{fail}) = floor(p_l \times (e + |c_i|))$
    if $|c_i| \leq E(e_{fail})$ then
      for each edge $j$ in cut $c_i$ do
        $P_1$ is the starting node for the edge $j$
        Compute point $C_1$, $|PC|$ distance away from $P_1$, parallel to $SD$ line
        Compute point $P_1'$, perpendicular projection of $P_1$ on $SD$ line
        Compute point $C_1'$, perpendicular projection of $C_1$ on $SD$ line
        $S_{c_j} = k \times \left(\frac{|P_1D|}{|SD|}\right) \times |P_1P_1'| \times |C_1C_1'| \times |P_1C_1| \times neighborCount$
        $p_{e_j} = 1 - \left(\frac{1}{1 + e^{s_{c_j}}}\right)$
      end for
      $p_e = \prod_{j=1}^{k} (p_{e_j} + p_l) / 2$ // Failure probability of links of cut $c_i$
    end if
  end if
end for
$p_e = \sum_{i=1}^{k} p_{e_i}$

$Reliability = 1 - p_e$
3.4.2 Results

In this section we present comparative results between OPNET simulations and the analytical model for Petal Routing with redundancy tuning. Figure 3.30 shows a plot of reliability versus petal width for a 200 node perturbed grid topology for petal routing with backoff, where the topology was provided to the analytical model. The analytical model in this case is using Algorithm 4. Using the backoff algorithm (2) increases the reliability from simulations and using Algorithm 4 for the analytical model reduces the reliability values. The result is that the difference is these values from simulations and the model is much less than in the case without backoff.
Figure 3.30: Reliability vs. Petal Width, 200 node perturbed grid topology provided to analytical model, using Petal routing with backoff

Figure 3.31 shows a similar plot for a 150 node clustered topology where the topology was not provided to the analytical model. In this case, it can be observed that the reliability values from the analytical model and from simulations are close for larger petal widths. It can be
reasoned that this is because the effect of the clustered topology is reduced for larger networks, i.e., larger widths the petal.

![Reliability vs. Backoff (sec)](image)

Figure 3.31: Reliability vs. Petal Width, 150 node clustered topology, not provided to analytical model, using Petal routing with backoff
Chapter 4

Geo-Diverse Paths for Jamming Mitigation

4.1 Problem Description

In this chapter, we describe a previously presented routing layer approach for jamming mitigation [50]. Similar to Petal routing this technique also uses redundancy to increase reliability in presence of failures. However this approach uses diverse node disjoint paths as opposed to diffused pathsets over a region in Petal routing. A unique feature of this diverse multipath approach is that it carries out power control on the nodes of the paths to further protect against jammer in the network. In the following sections, we first describe a model to study the effects of jamming, followed by two approaches to jamming mitigation, namely proactive and reactive. We compare the performance of this approach to Petal routing.
4.2 Routing Methodology

4.2.1 Jamming Effects Modeling

We adopt a jamming effects model from contiguous research of our group [49]. In a typical jamming scenario, network nodes in the jammed region constantly find the medium busy due to higher energy (noise) on the medium. Due to the lack of an opportunity to transmit, they lose connectivity to the rest of the network. In this section, we present some techniques to model jamming effects for networks that use adaptive energy threshold for CCA. We choose the $R^n$ propagation model for modeling because it is widely applicable to both indoor and outdoor environments by selecting an appropriate path loss exponent.

$R^n$ Propagation Model

Theoretical and empirical results show that the average received signal power decreases exponentially with an increase in the distance between the transmitter and receiver [59]. As per the $R^n$ propagation model, path loss at a receiver with the distance $d$ from a transmitter is given by equation 4.1.

$$PL(dB) = PL(d_0) + 10 \log_{10} \left( \frac{d}{d_0} \right)^n$$  (4.1)

where $d_0$ is a reference point near the transmitter with known path loss $PL(d_0)$, $n$ is the path loss exponent which indicates the rate at which path loss increases with the distance.

Absolute received power based on the $R^n$ propagation model is given by the equation 4.2 [52].

$$P_R = \frac{P_T G_T G_R}{10^{(PL(d_0)+10\log_{10}(d/d_0)^n)/10}}$$  (4.2)

where $P_R$ is received power, $P_T$ is transmit power, $G_T$ is transmit antenna gain, $G_R$ is receive antenna gain and other terms have the same meaning as in the path loss equation 4.1. Using the $R^n$ propagation model the jamming to signal ratio at the receiver is shown in Formula 2.2.
Jamming Vulnerability of a Link

Jamming vulnerability of a link is a region around the receiver in which the jammer’s presence causes link(s) to fail. Jamming vulnerable region is considered circular under simple constant range model and the region is termed as a **jamming circle**. Jamming circle radius formula 4.3 is derived from the JSR formula 2.2 by rearranging the terms.

\[
D_{JR} = D_{TR} \left( \frac{P_J}{P_T \times JSR} \right)^{1/n}
\]

(4.3)

---

**Figure 4.1:** Jamming Vulnerability of a Link

---

Jamming Vulnerability of a Path

If the jammer’s power and minimum JSR to break a link are given, then jamming circle radius for each link on a path can be found using the equation 4.3. Jamming vulnerability of a path is the total region occupied by these jamming circles. Figure 4.2 shows an example of jamming vulnerability of a path.
4.2.2 Proactive Protection against Jamming

The proactive approach to prevent jamming involves routing data redundantly along node disjoint paths. Power control on the links of the node disjoint paths help defend against simultaneous jamming of such paths. We first present a technique to select node disjoint paths and then discuss the steps involved in proactive protection.

Multipath Routing and Power Control Approach

Assume that jamming conditions in the network (number of jammers, jammer’s maximum power and jamming strategy - BBN, PBN etc.) are known to the network nodes. Distributed jamming detection and localization techniques [77] can provide this information to the network nodes. Using this information, the jamming vulnerable region for a traffic flow which uses a single path, can be computed as described in section 4.2.1. We propose a multipath routing and power control approach for reducing jamming vulnerability of the traffic flow. In this approach, a traffic flow is routed redundantly on node disjoint paths and power control is performed on the links of the node disjoint paths. Redundant routing on the node disjoint paths ensures that any one path’s failure due to jamming doesn’t disrupt the traffic flow. Power control on the links of the node disjoint paths is required to defend against simultaneous jamming of the paths.
by a jammer. Power control is performed such that the following two conditions are satisfied:

1. Jamming vulnerable regions around the source and destination of the node disjoint paths are minimum possible

2. Jamming circles around the intermediate nodes on one path do not overlap with any jamming circles on the other paths

Figure 4.3 shows an example of the two node disjoint paths satisfying conditions 1 and 2.
Algorithm for Preplanned Protection

Based on the multipath routing and power control approach, we describe the steps involved in providing proactive protection against a single jammer. For a given source and destination in the network, power is assigned to links of distinct path pairs. Power assignment on the links is carried out such that conditions 1 and 2 from the section 4.2.2 are satisfied. For a given path pair \((A, B)\), the jamming circle radius for intermediate nodes on the paths \(A\) and \(B\) are found, such that condition 2 from the section 4.2.2 is satisfied. If there is no power assignment that satisfies condition 2 from section 4.2.2, then path pair \((A, B)\) is ignored. The end nodes for a path-pair \((A, B)\) are assigned the maximum allowed power, if condition 1 is already satisfied. This ensures that jamming circles around the end nodes are minimum possible and condition 2 is satisfied. These steps are carried out for all possible path pairs and the pair for which the total assigned power is minimum is chosen. The worst case running time for this algorithm is \(O(n^2)\).

Table 4.1: Algorithm 6 Terminology

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(PathSet(u, v))</td>
<td>set of node disjoint paths between nodes (u) and (v)</td>
</tr>
<tr>
<td>(LocSet(u, v))</td>
<td>set of node locations for nodes in (PathSet(u, v))</td>
</tr>
<tr>
<td>(P^d(u, v))</td>
<td>set of default power values for the links in (PathSet(u, v))</td>
</tr>
<tr>
<td>(P^d(X))</td>
<td>set containing default power assigned to the links on a path (X)</td>
</tr>
<tr>
<td>(P^d_h \rightarrow a)</td>
<td>default power for a link (h \rightarrow a)</td>
</tr>
<tr>
<td>(P_X)</td>
<td>set containing power assigned to the links on a path (X) by the algorithm</td>
</tr>
<tr>
<td>(P_h \rightarrow a)</td>
<td>power assigned to a link (h \rightarrow a) by the algorithm</td>
</tr>
<tr>
<td>(P_{max})</td>
<td>maximum transmit power that can be assigned to a link</td>
</tr>
<tr>
<td>(P_{jam})</td>
<td>jammer’s maximum power</td>
</tr>
<tr>
<td>(P_{La})</td>
<td>transmit power assigned to the node (a)’s incoming link of length (L_a)</td>
</tr>
<tr>
<td>(JSR)</td>
<td>minimum jammer-to-signal ratio (JSR) that breaks an incoming link</td>
</tr>
<tr>
<td>(n)</td>
<td>path-loss exponent in the (R^n) propagation model</td>
</tr>
<tr>
<td>(X, Y, A, B)</td>
<td>paths</td>
</tr>
<tr>
<td>(PairSet)</td>
<td>set containing pairs of node disjoint paths</td>
</tr>
<tr>
<td>(R_{La})</td>
<td>jamming circle radius for the node (a)’s incoming link of length (L_a)</td>
</tr>
<tr>
<td>(R_u)</td>
<td>jamming circle radius at node (u)</td>
</tr>
</tbody>
</table>
Algorithm 6 shows the complete set of steps for the proactive power control approach, and Table 4.1 shows the terminology used in Algorithm 6.

**Algorithm 6 Geo-Diverse Proactive Protection against Jamming**

**Input:** \( \text{PathSet}(u,v), \text{LocSet}(u,v), P^d(u,v), P_{max}, P_{jam}, JSR, n \)

**Output:** One of the following outputs: (1) \( X, Y, P_X, P_Y \) (2) NULL

**Initialization:** \( \text{MinPowSum} = \infty, \text{Output} = \text{NULL} \)

for each PathSet \( \{(X,Y) / X,Y \in \text{PathSet}(u,v) \text{ and } X \neq Y\} \) do

for Node \( a \) : Path A do

for Node \( b \) : Path B do

\( L_a = \text{length of biggest incoming link to node } a \)

\( L_b = \text{length of biggest incoming link to node } b \)

\( D = \text{linear distance between nodes } a \text{ and } b \)

\( R_{L_b} = L_b \times ((1/JSR) \times (P_{jam}/P_{max}))^{1/n} \)

if \( R_{L_b} > D \) then

continue to next PathSet as it is impossible to avoid overlap on two paths

end if

\( R_{L_a} = D - R_{L_b} \)

\( P = (P_{jam}/JSR) \times (L_a/R_{L_a})^n \)

if \( P > P_{max} \) then

continue to next PathSet as required power to avoid overlap exceeds \( P_{max} \)

end if

Compute jamming circle radius \( R_{L_a} \) and \( R_{L_b} \)

Compute required transmit power \( P_{L_a} \) and \( P_{L_b} \)

Adjust jamming circle radius if required

Compute power assignment for incoming link at node \( a \) of Path A

end for

end for

for Node \( b \) : Path B do

for Node \( a \) : Path A do

\( L_a = \text{length of biggest incoming link to node } a \)

\( L_b = \text{length of biggest incoming link to node } b \)

\( D = \text{linear distance between nodes } a \text{ and } b \)

\( R_{L_a} = L_a \times ((1/JSR) \times (P_{jam}/P_{max}))^{1/n} \)

if \( R_{L_a} > D \) then

continue to next PathSet as it is impossible to avoid overlap on two paths

end if

\( R_{L_b} = D - R_{L_a} \)

\( P = (P_{jam}/JSR) \times (L_b/R_{L_b})^n \)

if \( P > P_{max} \) then

continue to next PathSet as required power to avoid overlap exceeds \( P_{max} \)

end if

end for

end for
end if
Compute jamming circle radius $R_{La}$ and $R_{Lb}$
Compute required transmit power $P_{La}$ and $P_{Lb}$
Adjust jamming circle radius if required
Compute power assignment for incoming link at node b of Path b
end for
end for

$PowSum(A, B) =$ summation of power assigned to links of paths $A$ and $B$

if $PowSum(A, B) < MinPowSum$ then
  $MinPowSum = PowSum(A, B)$
  $Output = \{A, B, P_A, P_B\}$
end if
end for
Return $Output$

### 4.2.3 Reactive Protection against Jamming

Reactive protection against jamming involves computing alternate routes to destination only when a link on the path to the destination fails. In this section, we present a distributed geographic routing algorithm that finds an alternative route to the destination, starting from the first node with failed link on the original path. Both the proactive and reactive techniques ultimately produce geo-diverse source-destination paths, the difference being that in the latter, alternate path computation is initiated only when link failure is detected on the primary path.

#### Link Failure Detection

We assume that every node in a network can detect link failures by performing periodic link maintenance for all neighboring links. Link maintenance is a mechanism by which a node is able to detect whether a neighboring link is functional. There are various ways to perform link maintenance. For example, a node $A$ periodically sends ‘Hello Request’ packets to a neighbor node $B$. Node $B$ sends ‘Hello Reply’ packets to node $A$ for each ‘Hello Request’ packet received. Node ‘$A$’ can monitor link condition (failed or working) based on the ‘Hello Request’ and ‘Hello Reply’ packets statistics over a period of time. Link is considered failed if the number of ‘Hello Reply’ packets received are much less compared to the number of ‘Hello Request’ packets sent in a given time interval.
Algorithm for finding an Alternate Path

A jammer causes collocated link failures in the network. Collection of collocated failed links forms a failed region, called a **jamming void**. The size of a jamming void depends on the type of jamming and the jammer’s power. A high power jammer causes a bigger jamming void compared to a low power jammer. Traffic on a path which passes through the jamming void gets disrupted. The algorithm presented in this section discovers a new path to route traffic around the jamming void to the destination. Algorithm 7 computes an alternate path between the first node \( n \) with a failed link on the original path, and the destination. A working path from the source to the destination is formed by merging the original path (from the source to the node \( n \)) and newly discovered path (from the node \( n \) to the destination). Traffic from the source to the destination is routed on the working path until the original path starts working. Algorithm 7 uses \textit{ROUTE\_DIS} and \textit{ROUTE\_SUC} packets (Figure 4.4) for route discovery.

<table>
<thead>
<tr>
<th>type</th>
<th>org</th>
<th>org_loc</th>
<th>dest</th>
<th>dest_loc</th>
<th>trav_list</th>
<th>DLM</th>
<th>ignore_list</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ROUTE_DIS Packet</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>org</td>
<td>dest</td>
<td>path</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ROUTE_SUC Packet</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ROUTE\_DIS** packet is created by the node that initiates a route discovery and it is forwarded among network nodes as per the algorithm 7 until a path to the destination is found or until the route discovery fails as the destination is not reachable. \textit{type} identifies \textit{ROUTE\_DIS} packet, \textit{org} contains the node id that initiated the route discovery to the destination, \textit{org\_loc} contains location of the \textit{org} node, \textit{dest} contains destination node id for the route discovery, \textit{dest\_loc} contains location of the \textit{dest} node, \textit{trav\_list} is the list of node ids on the path traversed.
from the \textit{org} node to a current node which has just received the packet. Node ids that cannot lead to the \textit{dest} are added to the \textit{ignore list}. Nodes in the \textit{ignore list} are not traversed again during the route discovery. \textit{trav list} and \textit{ignore list} are variable size lists. DLM is a special delimiter character which separates \textit{trav list} and \textit{ignore list}.

When route discovery succeeds, the destination creates a \textit{ROUTE_SUC} packet and sends it to the node which originated the route discovery. \textit{org} contains node id that created \textit{ROUTE_SUC} packet. \textit{type} identifies \textit{ROUTE_SUC} packet. \textit{dest} contains destination node id for the packet. \textit{path} is a variable sized list containing node ids on the \textit{org} to \textit{dest} path. Table 4.2 describes terminology used by algorithm 7.

Table 4.2: Algorithm 7 Terminology

<table>
<thead>
<tr>
<th>\textit{Cur_node_id}</th>
<th>Self identity of a node</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{Neighbors}</td>
<td>Set containing all active one hop neighbors of a node</td>
</tr>
<tr>
<td>\textit{Neighbors_loc}</td>
<td>Set containing locations of all active one hop neighbors of a node</td>
</tr>
<tr>
<td>\textit{Packet}</td>
<td>Packet(\textit{ROUTE_DIS} or \textit{ROUTE_SUC}) input to the algorithm</td>
</tr>
<tr>
<td>\textit{2 * Path}</td>
<td>If path to the destination is found, algorithm 7 returns \textit{Path}, on the node that initiated route discovery</td>
</tr>
<tr>
<td>\textit{2 * FAIL}</td>
<td>If no path to the destination is found, algorithm 7 returns \textit{FAIL}, on the node that initiated route discovery</td>
</tr>
<tr>
<td>\textit{NONE}</td>
<td>If there is nothing to return, algorithm 7 returns \textit{NONE}</td>
</tr>
</tbody>
</table>

\textbf{Algorithm for Reactive Protection}

Algorithm 7 is a distributed algorithm and it is run on every node in the network. We assume that every node in the network knows its own location as well as the locations of its neighbors. We assume that the node initiating a route discovery knows the location of the destination. Algorithm 7 is invoked when a node initiates route discovery or when a \textit{ROUTE_DIS} or \textit{ROUTE_SUC} packet is received by the node.
**Algorithm 7 Geo-Diverse Reactive Protection against Jamming**

**Input:** Cur.node.id, Neighbors, Neighbors.loc, Packet

**Output:** One of the following outputs: PATH, FAIL

if Packet.type == ROUTE_DISC then
    if Packet.dest == Cur.node.id then
        Create ROUTE_SUC packet
        Search for a neighbor x from NewPacket.path. Remove any nodes between x and Cur.node.id from NewPacket.path. Send NewPacket to node x
    else
        N ← Neighbors not in (Packet.ignore.list OR Packet.trav.list)
        if N == NIL then
            if Packet.org == Cur.node.id then
                Return FAIL
            else
                Remove Cur.node.id from Packet.trav.list, if it exists in the list
                Add Cur.node.id to Packet.ignore.list
                Send Packet to the last node in Packet.trav.list
            end if
        else
            Find a neighbor x from N with highest positive Advance [Advance is the difference of (linear distance between Packet.dest and Cur.node.id) and (linear distance between Packet.dest and neighbor node x)]
            if x == NIL then
                Find neighbor x that makes largest angle between the vector connecting (Packet.dest, Packet.org) and the vector connecting (Packet.dest, neighbor x)
            end if
            Add Cur.node.id to the end of Packet.trav.list, if not in list
            Send Packet to neighbor x
        end if
    end if
else if Packet.type == ROUTE_SUC then
    if Packet.dest == Cur.node.id then
        Return Packet.path
    else
        Search for a neighbor node x from Packet.path
        Remove any nodes between x and Cur.node.id from Packet.path
        Send Packet to node x
    end if
end if

Search performed by the algorithm 7 to reach a destination is essentially like a traversal on a tree which contains all reachable network nodes. Algorithm 7 makes a maximum $2(n-1)$
transmissions in the search process (i.e., traverse at most one path to each node), where \( n \) is all network nodes reachable by the originator of the route discovery. Note that choosing the neighbor with the largest \textit{advance} helps to propagate the packet towards the destination and finding the neighbor with the largest \textit{angle} helps to find an alternate geo-diverse path. If the destination is not reachable, then the last transmission of \textit{ROUTE\_DIS} packet is always for the node which originated the route discovery and \textit{ignore\_list} of the packet contains all the nodes that are reachable by that node. Information about all reachable nodes is very useful, as a new route discovery for any unreachable node can be avoided within a short time after a failed route discovery.

Algorithm 7 can be embedded into on-demand routing schemes to minimize the number of transmissions required in a route discovery. In on-demand routing schemes, route discovery typically floods the entire network with ‘route request’ packets. Therefore, performing a flooding based route discovery to find an alternative path for a failed path is very inefficient. On-demand routing schemes can instead use the algorithm 7 to find the alternative path. In most cases, Algorithm 7 requires very less number of transmissions in a route discovery compared to a flooding based route discovery mechanism. Evaluation results show that algorithm 7 finds optimal (shortest) paths in most cases.

4.2.4 Evaluation of Geo-diverse Routing

A grid topology of 81 nodes was used for the OPNET simulations, out of which 10 (source, destination) node pairs were selected for creating uni-directional traffic flows. We compare jamming mitigation effectiveness of three routing approaches under different jamming scenarios. These are: shortest path routing, redundant routing on node disjoint paths and redundant routing on node disjoint paths along with power control on the links of the paths (as described in Section 4.2.2). Figure 4.5 shows results from a network with a randomly moving jammer. It can be observed that the proactive approach for power assignment is able to receive all of the traffic sent by the source which is not the case for the other two routing techniques.
Algorithm 7 was implemented as a program that takes network topology, origin node and destination node as inputs, and then runs the algorithm 7 to discover a path from the origin node to the destination node. If a path is found, the program outputs the path from the origin node to the destination node and the number of transmissions required to reach the destination node. Fig: 4.6 shows the comparative results from a network having a uniform random topology of 400 nodes with three non-overlapping jamming voids. The results show that paths found from algorithm 7 are nearly optimal (shortest). Although algorithm 7 does a lot of of backtracking for some path pairs, the total number of transmissions in each case is less than the number of transmissions required for a network wide flooding for route discovery.
4.3 Comparing Geo-diffuse and Geo-Diverse Techniques

In section we provide initial results of comparison between Petal routing and the Geo-diverse routing technique. We interchangeably call the latter to be the Jamming Mitigation method. Both techniques use redundancy to protect against network failures. In Petal routing, a constrained flood is carried out over a limited region of the network, that we call the ‘petal’. In Jamming mitigation, we find node disjoint path and carry out power control on the paths to protect against jammers. In the former case, a lot more transmissions are encountered, because all nodes in the region participate in the transmission. In the latter case, the number of transmissions is less, however if the jammer can disrupt both paths, then the transmission will fail. In Petal routing, this is equivalent to creating a jamming void to disconnect the petal.

We implemented the Proactive Jamming Mitigation technique on OPNET. The disjoint path and power assignment computations were implemented outside OPNET and the values were provided to OPNET. We carried out tests using 100 node perturbed grid topologies with the source and destination fixed, where a stationary jammer was placed within the region defined
as the petal (Figure 4.7). The paths used for Geo-diverse routing corresponded somewhat to the outline of the petal. Packet delivery ratio for each technique was measured for increasing power of the jammer.

Figure 4.7: Sample topology showing the Petal, paths used for Geo-diverse routing, and the jammer

Figure 4.8 shows a plot of reliability versus increasing jamming power. It can be observed that the reliability of Petal Routing starts to reduce before that of Geo-diverse routing. The power control mechanism in Geo-diverse routing responds well in presence of a jammer. The fall in reliability values is much steeper in case of Geo-diverse routing. This is presumably because the approach uses disjoint paths, and if both paths fail then the transmission fails. In case of
Petal routing however, if the jammer disrupts a region of the petal, then the transmission may still succeed since the entire region is flooded. Thus even though the reliability gets reduced sooner, they fall to zero later than in case of Geo-diverse routing.

Figure 4.8: Comparison of Reliability between Petal Routing and Jamming Mitigation for increasing transmission power of the jammer

Figure 4.9 shows a comparative plot of the number of intermediate transmissions in Petal Routing and Geo-diverse routing for different topologies with a fixed node density. It can be seen that for these class of topologies Petal routing always incurs higher transmissions than Geo-
diverse routing. For other network characteristics (such as node density, clustering coefficient etc.) Petal routing could require fewer transmissions in specific jammer scenarios with multiple jammers. Analyzing such performance metrics is part of our ongoing work. It should be noted that Petal routing is suited for both random link failures and jammers in the network, whereas Geo-diverse routing is more suited for protection against jammers. So it would be beneficial to compare the techniques in random link failure scenarios as well.

![Number of transmissions vs. Jamming Power (watts)](image)

Figure 4.9: Comparison of the number of transmissions in Petal Routing and Jamming Mitigation for increasing transmission power of the jammer
Chapter 5

Empirical Measurements of Wireless Link Fluctuation Timescales

5.1 Problem Description

Wireless links are inherently prone to failures due to the medium. These can be observed as intermittent link failures, and can reduce the end-to-end packet delivery ratio of a network. Such random link failures usually do not follow any pre-defined pattern and are hard to model. In this study, we carry out experiments to measure link level fluctuations in wireless networks. We vary the length of the link in order to increase or reduce the signal strength at the receiver.

In addition to signal strength, we also vary the timescale at which packets are sent. Our hypothesis is that wireless links can be modeled using a two-state Markov process. Consider a system that is always in one of two states, $S = \{0, 1\}$. Every time a clock ticks, the system updates itself according to a $2 \times 2$ matrix of transition probabilities

$$
\mathbb{P} = \begin{bmatrix}
\alpha & 1 - \alpha \\
1 - \beta & \beta
\end{bmatrix}
$$

the $(i,j)$ entry of which gives the probability that the system moves from state $i$ to state $j$ at
any clock tick. The process can remain in the state it is in, and this occurs with probability $p_{ii}$. An initial probability distribution, defined on $S$, specifies the starting state. A two-state Markov chain is such a system, where the next state depends only on the current state and not on any previous states.

A discrete time Markov chain is a stochastic process, where transitions occur at regular intervals of time, whereas transitions can occur at any interval for a continuous time Markov chain. A wireless link can be thought of as a continuous time Markov process, which goes up and down due to the uncertainty of the medium. Our goal in this study is to carry out experiments to study wireless link behavior, and to identify if such links transition between states at discrete times. Further, we would like to identify the timescale or the time interval at which state transitions occur. In subsequent sections, we describe the experiment methodology and results from outdoor and OPNET simulator measurements.

5.2 Experiment Methodology

5.2.1 Outdoor wireless measurements

For outdoor experiments, we set up an access point and a UDP server on one computer. A second computer acts as a UDP client, sending packets to the server. While sending UDP packets from the client, we observe the wireless environment using a handheld network analyzer, which
is able to measure the signal strength from the AP. The experiment set up is shown in Fig. 5.2.

![Outdoor experiment setup](image)

Figure 5.2: Outdoor experiment setup

We vary the distance between the sender and receiver, in order to gradually decrease the signal strength. The actual value of the signal strength is observed using the network analyzer, which is kept close to the sender. For each value of signal strength, we send packet streams consisting of 1000 packets at four different rates: 1, 10, 25 packets per second, as well as back to back. Each packet has a stream ID and a sequence number, using which, the receiver can identify lost packets.

We carried out the experiments in three distinct wireless environments: (1) without radio interference or obstacles, (2) without radio interference but with obstacles, (3) with radio interference but without obstacles. The physical site for (1) was at the NCSU Loonie Poole Golf Course, for (2) it was a walking/running trail near Lake Raleigh, and for (3) it was in an open space on NCSU Centennial Campus. Figure 5.3 shows a sample reading from WiSpy DBx network analyzer at location (1), where the signal strength for the hosted AP was -25 dBm and there was no interference from other wireless networks. Figure 5.4 shows a similar reading in location (3), where the signal strength of the AP (wolfnet) was -60 dBm. In Section 5.3.1 we
present our observations from experiments carried out at the three locations.

![WiSpy DBx network analyzer](image)

Figure 5.3: Sample reading from WiSpy DBx network analyzer: signal strength = -25 dBm, no interference from other wireless sources

### 5.2.2 OPNET simulation measurements

We carried out experiments to observe link behavior in the OPNET simulator. The method for the experiments was as follows: generate a random network topology with OPNET’s wireless deployment tool, using the `manet WKSTN ADV` node model. For each node pair, make the sender broadcast a stream of 1000 packets at a certain time interval $i$. The receiver keeps track of the last state of the link, based on whether or not it received a packet in the previous time
step \((t - i)\), where \(t\) is the current time. If it does receive a packet in the current time step, it increases the count for appropriate transition (01 if the last state was 0 and 11 if the last state was 1). After all 1000 packets have been received, values for each row of the transition matrix are normalized such that they add up to one. If the number of observed state transitions from 0 to 0 is denoted as \(a\), from 0 to 1 as \(b\), from 1 to 0 as \(c\) and from 1 to 1 as \(d\), then that transition probability matrix for the link can be written as,

\[
\mathbb{P} = \begin{bmatrix}
\frac{a}{a+b} & \frac{b}{a+b} \\
\frac{c}{c+d} & \frac{d}{c+d}
\end{bmatrix}
\]  

(5.1)

Figure 5.4: Sample reading from WiSpy DBx network analyzer: signal strength of AP “wolfnet” -60 dBm, interference from other wireless sources observed
Example 1

Consider a wireless network where 10 packets are sent between a pair of nodes $x$ and $y$. Suppose that $y$ receives packets at time-steps 1, 3, 4, 5, 8, 9, 10 (as shown in Fig. 5.5).

Based on the previous state, the count of state transitions is updated at each time-step (the first time-step is discarded, since the previous state is unknown at that time-step). Table 5.1 shows all the link states and corresponding transition count update for each time-step. Note that $-1$ for “previous” of time-step 1 indicates that it is undefined.

After 10 time-steps, the sum of the state transitions will thus be, $a = 1$, $b = 2$, $c = 2$, $d = 4$. From (5.1), the transition probability matrix for the link would be,

$$P = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}$$
### Table 5.1: State transitions for Example 1

<table>
<thead>
<tr>
<th>Time-step</th>
<th>Last State</th>
<th>This State</th>
<th>State Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>n/a</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>c</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>a</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>b</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>d</td>
</tr>
</tbody>
</table>

#### 5.2.3 Visualizing Transition Probability Matrices

Using the approach described in 5.2.2, we compute the $2 \times 2$ transition probability matrix of a link for a given signal strength and packet rate at the sender. It is worth noting that while the packet rate can be fixed programmatically, the signal strength is not a fixed value, it can vary for 2 fixed nodes. The signal strength values shown in all the experiments in this chapter, correspond to average value over the period of measurement.

To visualize transition probability matrices, we use a polar plot where each value is plotted along the 45° line in each quadrant. The quadrants themselves correspond to the transition probability matrix (i.e. quadrant I = $p_{01}$, II = $p_{00}$, III = $p_{10}$, IV = $p_{11}$). For example, given the transition probability matrix:

\[
P = \begin{bmatrix}
0.23 & 0.77 \\
0.45 & 0.55
\end{bmatrix}
\]

the polar plot would be as shown in Fig. 5.6.

Visualizations for a strong link and a weak link are shown in Fig. 5.7.
5.2.4 Expected Observations

For both the outdoor and OPNET experiments, we vary the time interval $t_s$ at which the sender sends packets. We expect one of the three following cases to occur, based on the value of $t_s$ in relation to the timescale of link fluctuations $t_l$.

- **Case 1**: If the time interval matches the timescale at which wireless link fluctuations occur, or $t_s = t_l$, we would expect to see the actual link transition probabilities at the end of the experiment.

- **Case 2**: If the time interval $t_s > t_l$, then several fluctuations could occur in between each sent packet. The resulting transition probability matrix in this case would vary widely.
Figure 5.7: Transition Probability Matrix Visualization for a ‘strong link’ (left) and a ‘weak link’ (right)

between different runs of the experiment.

• Case 3: If $t_s < t_l$, then for each time step $t_l$ our sender would have sent multiple packets to the receiver. This would be identified, if several runs of the experiment result in high values of $\alpha$ and $\beta$, i.e., high probability of link staying up when it is up and down when it is down. Visualization for this case is shown in Fig. 5.8.

5.3 Results

5.3.1 Outdoor experiments

We conducted link level measurements in three distinct wireless environments.

• No interference, no obstacles: These experiments were carried out in an environment where there was no wireless interference or physical obstacles. We found that for all the different time intervals the experiment was carried out, there were negligible packet losses.
For rates such as 1 packet per second, there was no loss even for a weak signal of $-87$ dBm. For packets sent back to back, there was only about 0.02% packet loss. If the signal strength is further reduced to $<-90$ dBm, the socket connection between the sender and receiver was broken. The transition probability matrix visualization for this case was similar to Fig. 5.7a for a strong link.

- **No interference, obstacles present**: For the experiments with physical obstacles (mostly trees and other foliage), we found that packet losses were very low for high signal strengths. However, for low signal strengths, almost all packets were lost. For the packets that reached the destination, the usual pattern for all packet rates was consecutive losses,
followed by consecutive successes, for all timescales (Fig. 5.9). This leads us to believe that the wireless link fluctuations in this environment were at a timescale higher than the fastest packet rate (i.e., back to back packets).

Figure 5.9: Transition Probability Matrix Visualization for wireless environment with no interference, but physical obstacles present

- **Wireless interference, no obstacles:** These experiments were carried out in presence of wireless interference but no obstacles. By varying the signal strength and packet rate, we observed varying levels of packet losses, as shown in Fig. 5.10. In this case, we found that back to back packets always incurred high losses, while a rate of 1 packet per second, incurred very low losses.
By computing the transition probability matrices, we found that for low packet rates and high signal strength (−19 dBm to −56 dBm), the values of $\alpha$ and $\beta$ were very high as shown in Table 5.2. We believe that in this case fluctuations occur at a timescale less than the second timescale. For higher packet rates, this phenomenon was not observed. The lowest packet rate at which high $\alpha$ and $\beta$ values are not observed, could be the true transition probability for the link. Based on this approach, it can be seen from Table 5.2 that the optimal packet rate to observe link fluctuations varies for each signal strength value.

For a signal strength of −19 dBm, all packet rates up to 25 packets per second, show low fluctuations at the corresponding timescale. For back to back packets however, we observe a change in transition probabilities. Similarly for a signal strength of −56 dBm, this change is occurred at the same timescale. For a signal strength of −68 dBm, a change is occurred at the 10 packets per second timescale. For a signal strength of −82 dBm, no specific change is noted for the timescales tested.

Visualizations of the transition probability matrices from Table 5.2 are shown in Fig. 5.11.
Each plot represents all matrices for a given signal strength value, i.e., each column of Table 5.2 corresponding to different signal strengths.

Table 5.2: Transition probability matrices for varying packet rates and receiver signal strengths, for wireless environment with interference and no obstacles

<table>
<thead>
<tr>
<th></th>
<th>-19dBm</th>
<th>-56dBm</th>
<th>-68dBm</th>
<th>-82dBm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 packet per second</td>
<td>[1.0 0.0]</td>
<td>[1.0 0.0]</td>
<td>[0.998 0.002]</td>
<td>[0.167 0.833]</td>
</tr>
<tr>
<td></td>
<td>[0.0 1.0]</td>
<td>[0.0 0.1]</td>
<td>[0.14 0.86]</td>
<td>[0.178 0.822]</td>
</tr>
<tr>
<td>10 packets per second</td>
<td>[1.0 0.0]</td>
<td>[1.0 0.0]</td>
<td>[0.189 0.811]</td>
<td>[0.703 0.297]</td>
</tr>
<tr>
<td></td>
<td>[0.001 0.999]</td>
<td>[0.001 0.999]</td>
<td>[0.031 0.969]</td>
<td>[0.123 0.877]</td>
</tr>
<tr>
<td>25 packets per second</td>
<td>[1.0 0.0]</td>
<td>[1.0 0.0]</td>
<td>[0.096 0.904]</td>
<td>[0.383 0.617]</td>
</tr>
<tr>
<td></td>
<td>[0.001 0.999]</td>
<td>[0.001 0.999]</td>
<td>[0.029 0.971]</td>
<td>[0.084 0.916]</td>
</tr>
<tr>
<td>Back to back packets</td>
<td>[0.002 0.998]</td>
<td>[0.006 0.994]</td>
<td>[0.59 0.41]</td>
<td>[0.801 0.199]</td>
</tr>
<tr>
<td></td>
<td>[0.003 0.997]</td>
<td>[0.007 0.993]</td>
<td>[0.529 0.471]</td>
<td>[0.178 0.822]</td>
</tr>
</tbody>
</table>

Fig. 5.12(left) shows a visualization of transition probability matrices obtained from runs where the packet rate was increased more gradually: 20, 40, 60, . . . , 500 packets per second for a -60 dBm link. In this case, it was interesting to note that the link went from a higher to lower value of $p_{00}$ as the packet rate was increased. We believe that the link fluctuations are higher than the initial packet rates, and thus the transition probability matrices look somewhat like Case 3 in 5.2.4 (Fig. 5.8). As the packet rate is increased, this phenomenon is not observed. If the packet rate is continuously increased, after a certain value, the $p_{00}$ probability will again increase and the general availability of the link would likely decrease. We also plotted the count of the number of transitions for each case in this run 5.12(right), to validate the normalized values on the left.
Outdoor Experiment Observations

From the above experiments, we observe that for the same environment, links fluctuate at different timescales for different signal strengths. We do not claim to obtain an accurate value
of the timescale of fluctuations from the above experiments. A more systematic increase of packet rate from the sender with a wider variety of values would be required to observe the timescale at which link fluctuations switch from the pattern of consecutive losses followed by consecutive successes (Fig. 5.8) to a more stable behavior between multiple runs. We attempt this gradual increase in packet rates as shown in Fig. 5.12, however, multiple runs of such experiments would be needed to conclude which timescale yields the true transition probability for the link.

5.3.2 Simulation experiments

We carried out OPNET simulations using the method described in 5.2.2. We used a randomly generated network of 10 nodes, and ran simulations for all node pairs in the network. We present representative results of a randomly picked node pair (about 80 meters apart) in the network. By increasing the sender’s packet rate from 20, 40, ..., 800 packets per second, we observed the
computed transition probability matrices (shown in Fig. 5.13)

The first plot in Fig. 5.13 shows the transition probability matrices for the entire range of packet rates examined (20 - 800 packets per second). From the figure, clearly no conclusion can be made about the transition behavior of the link. However, if we observe packet rates 20 - 220 (second plot in Fig. 5.13), it can be seen that the transition probability matrix exhibits high loss/high success behavior (i.e. Case 3 of 5.2.4). Starting from 240 packets per second up to 460 packets per second, we observe a switch in the link behavior and it is consistent for all the packet rates in this range (Case 1 of 5.2.4). For rates higher than 460 packets per second, we observe that the resultant transition probability matrices varies significantly (Case 2 of 5.2.4).

We also plotted the overall percentage of packets lost for each packet rate (Fig. 5.14). It was interesting to note that the percentage was varying, then it stabilized for certain packet rates, after which it increased monotonically. Further, the packet rates at which the transition probability was consistent from Fig. 5.13 (240 - 460), the loss rate in Fig. 5.14 was also stable. This behavior was consistent for various node-pairs for the 10-node graph. It was observed that the specific packet rates at which the percentage of lost packets stabilized, varied considerably from one node-pair to another.

From the experiments, we can conclude that for the node-pair shown above, the resultant transition probability matrix for packet rates from 220 to 460 packets per second, show the true link fluctuations, while packet rates higher and lower than that range are outliers. Thus using OPNET simulations, we are able to vary the packet rates systematically, so as to identify a range of rates at which wireless link fluctuations occur. We use this technique to obtain transition probability matrices of wireless links in OPNET, in order to compare with a modeling approach presented in Chapter 6.

5.3.3 Observations

Our general observations from the outdoor and OPNET experiments are as follows:

1. Link fluctuation behavior for varying signal strength, depends on wireless interference and
the physical obstacles present in the environment

2. The optimal packet rate to obtain the true transition probability of a link varies from one environment to another

3. The optimal packet rate for a given wireless environment also varies based on the signal
Based on these findings, we cannot conclude on any optimal packet rate to correctly observe wireless link fluctuations, which would be applicable to all wireless environment and link strengths. Rather, the approach should be to carry out tests on wireless links by varying the signal strength and sender’s packet rate, in order to obtain the optimal packet rate for observing link level fluctuations. Such tests would need to be extensive and methodical, in order to gradually reduce the signal strength at the receiver as well as reduce the packet rate in small decrements. Initial findings show that the packet rates at which the availability of the link stabilizes, can help yield the true transition probability of the link. We observe that while a two-state Markov process is a reasonable approach to model wireless links, more specialized equipment would be required to make precise measurements of link fluctuations. In particular, a very high packet sending rate would allow one to observe periods of failures and successes, and thereby study the timescales of these failures.
Chapter 6

Modeling Networks with Linear Systems

6.1 Problem Description

We present a general approach to model routing in wireless networks [12,13]. Our goal is to design a model that can predict the highest achievable reliability for a transmission given the availabilities for individual links in a wireless network. The link availabilities are obtained by observing a link over a prolonged period of time, to get its steady state probability. The reliability prediction solely depends on the link probabilities and the network, and does not consider any specific routing algorithm. Our claim is that this approach gives the highest reliability path from a source to destination, and it can be used to compare and benchmark existing routing approaches.

In the following sections, we first present a modeling technique for wireless multihop networks as a stochastic dynamical system. We then present a system that uses reachability analysis as a tool to compute paths to the destination and their corresponding reliability probabilities.
6.2 The Model: Multihop Network as a Dynamical System

In order to use reachability analysis to gain a theoretical understanding of network performance (namely a network’s ability to deliver packets), the network must be modeled as a dynamical system, consisting of a state, a control input, and a transition model that determines how the state evolves over time. Additionally, in order to capture random link failures, the transition model is defined as stochastic and incorporates randomness into the state transitions. We therefore define the discrete time multihop network model as follows.

**Definition 1. Multihop Network Model**

1. $\mathcal{N} = \{1, 2, \ldots, N\}$ is a finite set of nodes.

2. $\mathcal{M} = 1, \ldots, M$ is a finite set of $M$ links between nodes.

3. $n_t \in \mathcal{N}$ is the node location of a packet in the network at time $t$.

4. $q_t \in \{0, 1\}^M$ is the state of all links in the network at time $t$, where the $m^{th}$ entry of $q$ equals 1 if link $m$ is up, and 0 if it is down.

5. $x_t = (n_t, q_t) \in \mathcal{X}$ is the full state of the system at time $t$, with $\mathcal{X} = \mathcal{N} \times \{0, 1\}^M$.

6. $u_t \in \mathcal{U}(x_t)$ is the routing decision for time $t$, telling the node $n_t$ where to send the packet given current link condition $q_t$.

7. $\tau: \mathcal{X} \times \mathcal{X} \times \mathcal{U} \rightarrow [0, 1]$ is a discrete stochastic transition kernel assigning a probability distribution to $x_{t+1}$ given $x_t$ and $u_t$.

We are given a network with a set of $N$ nodes labeled from 1 to $N$, and a set of links between those nodes labeled from 1 to $M$. By assigning the value 1 to any link that is currently up, and the value 0 to any link that is currently down, the variable $q_t^m \in \{0, 1\}$ can represent the availability of link $m$ at time $t$. The vector $q_t = [q_t^1, \ldots, q_t^M]^T \in \{0, 1\}^M$ then represents the state of every link (assuming $M$ total links) in the network at time $t$. Further, letting
\( \mathcal{N} = \{1, 2, \ldots, N\} \) be the set of all possible nodes, \( n_t \in \mathcal{N} \) can represent the node a packet is being held in at time \( t \).

We identify the state of the system using \( x_t \), where the subscript \( t \) indexes time. The state space of the system, \( \mathcal{X} \), represents all possible values \( x_t \) can take, for any time \( t \). The full state \( x_t \) combines \( n_t \) with \( q_t \) so that \( x_t = (n_t, q_t) \) and \( \mathcal{X} = \mathcal{N} \times \{0, 1\}^M \). The control input \( u_t \) is defined as the node the packet is sent to between times \( t \) and \( t + 1 \) according to some routing strategy, so that \( u_t \in \mathcal{U}(x_t) \subseteq \mathcal{N} \). The set \( \mathcal{U}(x_t) \) contains all potential nodes the packet can be sent to, given it is currently at node \( n_t \) and the links are in state \( q_t \).

In a first-order discrete time system, which is the only one we consider, the transition function is of the form \( x_{t+1} = f(x_t, u_t) \), so that the next state at time \( t + 1 \) is determined completely by the state and control input at time \( t \). In a stochastic dynamical system, the future state is not completely determined by the current state and control input, but rather has a level of uncertainty associated with it. For instance, a random variable \( v_t \) (often referred to as a disturbance or noise) may also affect the transition function \( f \), so that \( x_{t+1} = f(x_t, u_t, v_t) \). In this case, multiple values for \( x_{t+1} \) are possible, each with an associated probability of occurrence. Because \( x_{t+1} \) is a function of a random variable, it itself is a random variable, and can actually be thought of as a controlled Markov process. We can think of its transitions as being governed by a stochastic transition kernel, \( \tau \), rather than the function \( f \), so that we have \( \tau(x_{t+1} \mid x_t, u_t) \) representing the probability distribution of \( x_{t+1} \) conditioned on the known values \( x_t \) and \( u_t \).

We thus define the stochastic transition function \( \tau \). If \( \mathcal{U}(x_t) \) is defined as above, so that \( u_t \) can only be selected from feasible forwarding nodes, and assuming that once a forwarding command is issued it is carried out with probability 1, then \( \mathbb{P}[n_{t+1} = i \mid x_t, u_t = i] = 1 \). If we further assume that the state of the links is governed by a discrete stochastic transition kernel \( T_q(q_{t+1} \mid q_t) \) so that the link state probabilities may be affected by previous link states, but not by the packet location or routing policy, then

\[
\tau(x_{t+1} \mid x_t, u_t) = 1_{u_t}(n_{t+1}) T_q(q_{t+1} \mid q_t) \tag{6.1}
\]
6.2.1 Modeling Wireless Link States

We now describe some variations of modeling wireless link states that we will use in the subsequent analysis, ranging from how $T_q$ is defined to what information is available at each node. By considering these different variations, we will have the tools to develop theoretical network performance results using an array of routing policies, to compare the performance of basic routing policies when different types of information are available, and to produce bounds on packet delivery probabilities. Although we intend to look at several variations of this model to reflect varying levels of stochasticity and information available to the controller when picking forwarding nodes, we now focus on two approaches to model wireless links.

In general, the discrete stochastic transition kernel $T_q(\bar{q}_{t+1} \mid \bar{q}_t)$ gives the probability that the link states at time $t + 1$ are as described in $\bar{q}_{t+1}$. This can be written as:

$$T_q(\bar{q}_{t+1} \mid \bar{q}_t) = \prod_{m=1}^{M} \mathbb{P}[q_{t+1}^m \mid q_t^m]$$  \hspace{1cm} (6.2)

**Independent Links, Stationary Link Probabilities**

Here we assume that the state of each link is independent of the state of all other links, and also independent of its past state, so that (6.2) simplifies to:

$$T_q(\bar{q}_{t+1} \mid \bar{q}_t) = \prod_{m=1}^{M} \mathbb{P}[q_{t+1}^m].$$  \hspace{1cm} (6.3)

**Independent, Markov Link Probabilities**

Here we again assume that the state of each link is independent of all other links, but that its probability follows the Markov property, and is determined by a transition matrix $\mathbb{P}^m$:

$$\mathbb{P}^m = \begin{bmatrix} \alpha & 1 - \alpha \\ 1 - \beta & \beta \end{bmatrix}$$  \hspace{1cm} (6.4)
so that $P_m(0,0) = \alpha$ is the probability of the link $m$ being down, given it was down at the previous time step, $P_m(0,1) = 1 - \alpha$ is the probability of link $m$ going up given it was down at the last time step, etc. Then

$$T_q(q_{t+1} | q_t) = \prod_{m=1}^{M} P_m(q_m^t, q_{m+1}^t)$$

(6.5)

6.3 The System: Reachability Analysis for Network Performance

Our general system consists of a centralized controller, or oracle that generates routing strategies based on knowledge of the current state of all links in the network (up or down), as well as network and link information available from the model in Section 6.2. Each node determines its forwarding strategy when it receives a packet by querying the pre-computed oracle, till the packet reaches the destination. In this section, we describe the technique of reachability analysis used by the oracle to pre-compute optimal paths to the destination.

6.3.1 Reachability Analysis for Stochastic Dynamical Systems

The concept of reachability can be applied to any dynamical system in order to determine whether the state of the system is able to remain within some desired set of states, reach a desired state or set of states, or both, all within a given time horizon. For stochastic dynamical systems with some level of uncertainty, stochastic reachability extends that concept to provide the probability of the state of the system reaching or maintaining a desired set of states [1].

Let $K \subset \mathcal{X}$ be the set of states we would like the system’s state to reach by some time $T$, and $\mathcal{K} \subseteq \mathcal{X}$ a set of states that the state should remain within for all $t = 0, \ldots, T$. If the system is known to start in a specific state $x_0$, and a set of control inputs $\bar{u} = [u_0, u_1, \ldots, u_{T-1}]$ have been predetermined, the reach-avoid probability is defined as

$$RA_{x_0}^{\bar{u},T}(\mathcal{K}, K) = \mathbb{P} \left[ \text{There exists } t \leq T \text{ such that } x_0, \ldots, x_{t-1} \in \mathcal{K}, x_t \in K \right]$$

(6.6)
with \( \mathbb{P} \) denoting probability.

Recalling that \( \mathbb{P}[x \in K] = \mathbb{E}[\mathbf{1}_K(x)] \), where \( \mathbb{E} \) is the expected value, and \( \mathbf{1}_K(x) \) is the indicator function defined as

\[
\mathbf{1}_K(x) = \begin{cases} 
1, & \text{if } x \in K \\
0, & \text{if } x \notin K
\end{cases}
\]

equation (6.6) can be rewritten as [69]

\[
RA_{x_0} T(\mathring{K}, K) = \mathbb{E} \left[ \sum_{t=1}^{T} \left( \prod_{i=0}^{t-1} \mathbf{1}_{\hat{K}}(x_i) \right) \mathbf{1}_K(x_t) \right] 
\]

since

\[
\sum_{j=1}^{T} \left( \prod_{i=0}^{t-1} \mathbf{1}_{\hat{K}}(x_i) \right) \mathbf{1}_K(x_t) = \begin{cases} 
1, & \text{if } \exists t \in [0,T] \text{ s.t. } x_0, \ldots, x_{t-1} \in \hat{K}, x_t \in K \\
0, & \text{else}
\end{cases}
\]

If, instead of assuming a predetermined set of control inputs \( \mathbf{u} \), we want to pick the control inputs \( u_t \) to maximize the reach-avoid probability, (6.7) can be reformulated as a stochastic optimal control problem:

\[
RA_{x_0} T(\mathring{K}, K) = \max_{u_0, \ldots, u_{T-1} \in \mathcal{U}} \mathbb{E} \left[ \sum_{t=1}^{T} \left( \prod_{i=0}^{t-1} \mathbf{1}_{\hat{K}}(x_i) \right) \mathbf{1}_K(x_t) \right] 
\]

Picking the input \( \mathbf{u} \) optimally provides an upper bound on the probability of reaching the set \( K \) by time \( T \), and remaining in set \( \mathring{K} \) until \( K \) is reached. Any other control input will generate a reach-avoid probability less than or equal to the optimal one.

One difficulty in solving the above optimization problem (6.8) is that we want to choose the control input \( u_t \) at time \( t \), based on the observed current state \( x_t \). This gives us what is called a “closed-loop” or feedback-based control input, which produces more optimal results. Needing a closed-loop control input leads to the concept of a policy or forwarding decision. A policy \( \pi \) is a function that maps states to control inputs, \( \pi_t : \mathcal{X} \rightarrow \mathcal{U} \). Given a state \( x_t \), the control input
$u_t$ is then chosen as $\pi_t(x_t)$. Therefore, rather than optimizing over a set of control inputs $\bar{u}$ in (6.8), we optimize over all possible policies. It is then the optimal policy that will give us an upper bound on the reach-avoid probability.

### 6.3.2 Reachability Analysis for Wireless Networks

We reachability techniques applied to Def. 1 to determine with what probability packets will reach some goal node, denoted $\text{end}$. We can also define a set of nodes $\tilde{\mathcal{N}} \subseteq \mathcal{N}$ that packets should be restricted to being sent to if nodes outside of this set are known to be unreliable or compromised by a malicious attacker, for instance. Using a formulation presented in [69], the probability that a packet starting from node $\text{start}$ reaches $\text{end}$ no later than time $T$, while avoiding nodes outside of $\tilde{\mathcal{N}}$ can be represented as follows.

$$RA_{\text{start}}(\tilde{\mathcal{N}}, \text{end}) = \mathbb{E}\left[\sum_{t=1}^{T} \left(\prod_{i=0}^{t-1} 1_{\tilde{\mathcal{N}}}(x_i)\right) 1_{\text{end}}(x_t)\right]$$

(6.9)

The notation $\tilde{\pi}$ denotes a specific routing policy that tells nodes where to send the packet based on the current state of all links. Specifically, $\tilde{\pi}$ is a sequence of functions $(\pi_0, \ldots, \pi_{T-1})$ mapping the current state to a routing decision, $\pi_t(n_t, q_t) = u_t$.

To address the problem defined in Section 2.1, we focus only on reaching an intended goal node, and do not consider a set $\tilde{\mathcal{N}}$ of nodes to avoid (so $\tilde{\mathcal{N}} = \mathcal{N}$). We also want to find an upper bound to the reachability probability (6.9) in order to determine the highest achievable reliability for packet transmission. The problem is twofold. First, the optimal routing policy $\tilde{\pi}^*$ must be calculated that delivers packets to $\text{end}$ with highest probability. Second, (6.9) must be evaluated according to the optimal policy $\tilde{\pi}^*$.

Both problems can be jointly solved using dynamic programming (for an overview of dynamic programming, see [7]). The following recursion, modified from Summers et al. [69], is
iterated backwards in time starting at $T$ and ending at 0.

$$V_T^*(n, \bar{q}) = 1_{\text{end}}(n)$$  \hspace{1cm} (6.10)

$$V_t^*(n, \bar{q}) = \max_{u \in U(n, \bar{q})} \left\{ 1_{\text{end}}(n) + \sum_{\bar{q}_{t+1} \in \{0,1\}^M} V_{t+1}^*(\tilde{\pi}_t^*(n, \bar{q}), \bar{q}_{t+1}) T_q(\bar{q}_{t+1} | \bar{q}) \right\}$$  \hspace{1cm} (6.11)

In fact, (6.11) reduces to

$$V_t(n, \bar{q}) = 1_{\text{end}}(n) + \sum_{\bar{q}_{t+1} \in \{0,1\}^M} V_{t+1}(u_t, \bar{q}_{t+1}) P_{\bar{q}}[\bar{q}_{t+1}]$$  \hspace{1cm} (6.12)

when the link probabilities are assumed completely independent, as in (6.3).

Equation (6.10) is evaluated first for all $(n, \bar{q}) \in \mathcal{N} \times \{0,1\}^M$, to get the so-called “value function” at time $T$, and then for all $t = T - 1, \ldots, 0$, the value function (6.11) is calculated recursively for every $(n, \bar{q})$, finally producing $V_0(n, \bar{q})$, where $V_0(\text{start}, \bar{q}_0) = RA^{\tilde{\pi}^*, T}_{\text{start}}(\mathcal{N}, \text{end})$. The value function $V_t(n_t, \bar{q}_t)$ tells us the following: at time $t$, and given that the packet is currently at node $n_t$ and links are currently up and down according to $\bar{q}_t$, what is the probability that the state reaches the desired node $\text{end}$ before time limit $T$ is reached.

The optimal routing policy is defined at each time step by

$$\tilde{\pi}_t^*(n, \bar{q}) = u_t = \max_{u \in U(n, \bar{q})} V_t^*(n, \bar{q}).$$ \hspace{1cm} (6.13)

When $T$ is finite, the routing policy is non-stationary, meaning the optimal control input $u_t$ may change depending on the time $t$ even though the state remains the same (i.e. $\tilde{\pi}_{t_1}(n, \bar{q})$ may not equal $\tilde{\pi}_{t_2}(n, \bar{q})$ for $t_1 \neq t_2$). To produce a stationary policy, assuming the stochastic kernel $\tau$ is stationary (not a function of $t$, which is true for the case (6.3) ) the equivalent infinite horizon ($T = \infty$) problem may be solved by repeatedly solving (6.11) until $V_t^*(n, \bar{q}) = V_{t+1}^*(n, \bar{q})$ for all $(n, \bar{q})$. In this case, the value function is calculated recursively until it converges, rather than a
fixed number of times, to produce the final value function \( V^*(n, \bar{q}) \) satisfying

\[
V^*(n, \bar{q}) = 1_{\text{end}}(n) + \sum_{\bar{q}_{t+1} \in \{0,1\}^M} V^*_{t+1}(\bar{n}_t, \bar{q}_{t+1}) T_q(q_{t+1} | \bar{q}).
\]

From a networking perspective, \( T \) can be considered the deadline or time to live (TTL) for a given packet transmission. If the TTL value is a finite number, we solve equation (6.11) exactly \( T \) times. If the deadline to reach the destination is unlimited, then we compute the value function recursively until it converges. A potentially infinite deadline can be used to identify reliable transmissions when the packet needs to be delivered regardless of the time taken for delivery, due to failures in the network.

6.4 Numerical Experiments

6.4.1 Experiment Set-up

The system presented in Section 6.3 can predict the reliability of a packet transmission, given the independent availability values of the links in the network. This computation is done by the “oracle” that knows the probabilities of each link being available in the steady state. The “oracle” also stores the computed paths to the destination to achieve the predicted reliability. To verify the paths computed by the “oracle”, we simulate networks with intermittent link failures. We also compare the predicted reliability of different transmissions with OPNET simulations of existing routing protocols.

In order to represent practical wireless networks and failure scenarios, we propose to use two metrics for a given source-destination pair, namely, deadline and ambient noise. Deadline signifies the time limit by which the packet must get delivered or \( T \) from the system in Section 6.3. Ambient noise or background noise is a major cause of packet drops in wireless mesh networks. As the noise in the network increases the quality of the links decrease, till the point when the link fails all or most of the time. While ambient noise is not represented directly in our
model, we consider this to be an inherent property of the input network to our system, and it is represented in the link probability values described in Section 6.2. By varying the deadline and ambient noise in the network, we compare predictions of our model with various simulations.

We carry out our experiments with stationary as well as Markov link probabilities. For both cases we compare the predicted values by the oracle with simulations. Simulations are of two kinds. First, we simulate the application of paths computed by the oracle, to compare reliability predicted by the oracle with achieved results based on applications of it. Simulations of our system are carried out with networks deployed using either a uniform random distribution or a 2D Poisson distribution. Nodes in the network have pre-set range and fading models implemented to account for multipath fading. We apply the oracle’s pre-computed paths to a sequence of likely events of links failing at different time instances. In other words, nodes in these simulations use the oracle’s paths as their routing policy. Based on the sequence of events occurring, nodes pick their best next hop from the pre-computed paths. We simulate multiple such scenarios and present the aggregated results of packet delivery ratios. These values would indicate the reliability achieved by different paths from the source to destination.

Simulations in OPNET are carried out using the wireless network deployment tool with uniform random node placement. We use the *manet wkstn adv* node model for ad hoc nodes, and run simulations with standard implementations of AODV [51] and OLSR [17]. Wireless link failures are simulated using the radio module in OPNET. We set appropriate fading parameters and ambient noise to simulate link failures. Further details on setting parameters in OPNET to vary the ambient noise, can be found in the following section (Sec. 6.4.2). Each set of 1000 simulations with a fixed source-destination pair and a network deployment results in a single data point. We compute the packet delivery ratio for a source-destination pair based on the number of packets that were received by the destination.

The output from our system is in the form of a vector of predicted reliability values, one for each combination of link states for links incident on the source node. We present these values in the form of best, average and worst case reliability. The best case reliability is achieved when all
the incident links of the source are up. For the average and worst case reliabilities, we compute the average and minimum reliability values from the system, excluding the case when all the incident edges of the source have failed. This definition of the best, average and worst case reliability values from the system, is used consistently in all the comparative plots.

6.4.2 Varying Ambient Noise in OPNET

We use the `manet_wkstn_adv` for our OPNET simulations of existing routing protocols such as OLSR and AODV. We vary the ambient noise to get varying levels of link availabilities for our simulations. Since wireless medium is broadcast by nature, a single transmission can affect many receivers simultaneously. Received signal strength at each receiver depends on many factors such as transmitted signal power, distance between transmitter and receiver environment noise etc. So each receiver may receive the transmitted signal differently. The timing of signal reaching the receivers also varies. This concept is implemented in OPNET as pipeline stages. A separate pipeline must be executed for each eligible receiver. The radio transceiver pipeline consist of total fourteen stages most of which must be executed on per receiver basis. All of the pipeline stages can be modified. Below is description of the stages relevant to ambient noise.

- **Interference Noise**: This stage is executed only when: the packet is valid and arrives at the destination while another packet is being received, or the packet is being received while another (valid/noise) arrives. The purpose of this stage is to account for the transmissions that arrive at the receiver concurrently. The value is stored only for the valid packets. The result can be shared by pipeline of two packets if both are valid packets.

- **Background Noise**: The purpose of this stage is to represent the effect of all noise sources except for other concurrently arriving transmissions (because these are accounted for by the interference noise stage). The expected result is the sum of the power (in watts) of other noise sources, measured at the receivers location and in the receiver channels band. Typical background noise sources include thermal or galactic noise, emissions from neighboring electronics, and otherwise un-modeled radio transmissions (such as commercial...
radio, amateur radio, or television, depending on frequency). We vary the ambient noise values in this pipeline stage to get varying link availabilities.

- **Signal-to-Noise Ratio**: This stage executed for a valid packet for following three conditions: (1) the packet arrives at its destination channel, (2) the packet is already being received and another packet (valid or invalid) arrives, and (3) the packet is already being received and another packet (valid or invalid) completes reception. The purpose of SNR stage is to compute the current average power SNR for the arriving packet. This calculation is usually based on values obtained during earlier stages, including received power, background noise, and interference noise. The SNR of the packet is important in determining receivers ability to correctly receive the packets content. The result computed by this stage is used by the simulation kernel to update standard output results of receiver channels and usually also by later stages of the pipeline.

- **Bit Error Rate**: BER stage is executed for all the valid packets for which SNR stage is executed. The purpose of the BER stage is to derive the probability of bit errors during the past interval of constant SNR. This is not the empirical rate of bit errors, but the expected rate, usually based on the SNR. In general, the bit error rate provided by this stage is also a function of the type of modulation used for the transmitted signal.

**Background Noise Pipeline Stage in OPNET: dra_bkgnoise**

As stated above, we vary the value of ambient noise in the Background Noise pipeline stage. The 802.11 standard model uses a default value of 1.0 for the noise figure ($f_n$). The total background noise ($B_N$) that effects the received packet (as calculated by the dra_bkgnoise pipeline stage) is therefore given by the Johnson Nyquist [73] noise formula:

$$B_n = BW(290f_n)k + A_N + I_N$$

(6.14)
where Boltzmann’s Constant $k = 1.379 \times 10^{-23}$, 290 is the receiver background temperature in degrees Kelvin, $BW$ is the transmission bandwidth of $16.56\,MHz$, the OPNET default ambient noise $A_N = 10^{-26}$, and the inter-packet interference $I_N$ is as calculated in the dra\_noise pipeline stage.

We vary the ambient noise $A_N$ from $10^{-26}$ to $10^{-20}$ to get effective noise values ranging from $-156\,dBm$ to $-96\,dBm$. Note that computed values for signal strength in OPNET was observed to be between $-50\,dBm$ to $-80\,dBm$. By varying the ambient noise, we were able to vary the strength of the wireless links, which in turn affected the availability of the link while comparing with our model in Section 6.2.

### 6.4.3 Using Stationary Link Probabilities

We compare the oracle’s predictions with stationary probabilities to OPNET simulations of wireless networks with existing routing protocols. In order to achieve a fair comparison between OPNET simulations and our system, we provide the network generated by OPNET to our system. Stationary link failure probabilities are computed by studying the links on OPNET. Steady state link availability for a certain ambient noise value was obtained from OPNET by making each node broadcast a stream of packets. One-hop neighbors compute the packet delivery ratio based on how many of the sent packets were received, in order to compute the success probability of the link.

Given this network generated by OPNET, and computed link failure probabilities, we use our system to get the oracle’s pre-computed paths. We compare the reliability of the paths computed by the oracle, with OPNET simulations of existing routing protocols. Figure 6.1 shows a sample 38 node network used for OPNET simulations and comparisons with the model. The network has 38 nodes spread over a region of 6 km$^2$ and the nodes transmit with a power of 200 milliwatts.
6.4.4 Using Markov Link Probabilities

To carry out comparisons between OPNET simulations and our model for the Markov case, we use an OPNET generated network of 8 nodes, and provide the topology information to our system. We obtain link transition matrices from OPNET by the following method: generate a stream of packets between each node pair in the network. For each packet received, a node looks up the state of the link in the previous time step, which it maintains. Based on the previous state it increases the appropriate count of received packets (down to up or up to up), and also accounts for all the missing packets in between this time step and last time it received a packet. At the end of the simulation, each link’s transition probability matrix values are obtained by normalizing the values along each of the two rows of the matrix shown in (6.4). The appropriate
value of the time-step is obtained by using the technique outlined in 5.2.2.

Using Markov transition probabilities creates computational challenges, in that we have to evaluate (6.11) for all possible states $x_t$, and for each $x_t$ and $u_t$ we will need to evaluate $\tau(x_{t+1} \mid x_t, u_t)$. To reduce the number of computations, we use stationary probabilities to generate link state probabilities for links more than two hops away from the current node. All comparisons of the oracle’s pre-computed paths include this approximation. In addition, due to the large number of computations, we present initial comparative results for the Markov case, with fairly small networks.

### 6.5 Evaluations and Results

#### 6.5.1 Stationary Link Probabilities

**Visualizing computed paths**

We observe the reliability of paths computed by the oracle using stationary link availabilities (as in 6.2.1). We vary the deadline value as well as the ambient noise (See Appendix 6.4.2), to note the difference in reliability. For this visualization, we used a randomly generated network of 20 nodes. It was observed that there were paths of lengths 5 hops and above between a randomly picked source-destination pair. We varied the deadline and the ambient noise. As expected, we observed that increasing the noise and decreasing the deadline both reduce the probability of successful transmission, until the reliability drops to 0.

**Simulations of variant 6.2.1**

We compare reliability predicted by the oracle, to reliability achieved from applications of the paths in real network scenarios. Since the pre-computed paths give the next hop information from any node, given any combination of incident edge states, we are able to apply these paths when specific links in the network are made to fail. Reliability values from the oracle are based on incident link states from the starting time step. We ran 1000 simulations for each ambient
Figure 6.2: Varying deadline and ambient noise for a 20 node network

noise value, and the result of each was aggregated to get the overall packet delivery ratio. Comparisons show that the reliabilities computed by the oracle are indeed achievable (Fig. 6.3). Note that the reliability for both the oracle and simulations drop to zero as the noise is increased because the network becomes disconnected if the noise becomes too large.
Figure 6.3: Comparison between pre-computed paths and simulations of the network for stationary probabilities

Comparisons to existing routing protocols

For comparisons with existing routing protocols, we ran OPNET simulations with OLSR and AODV routing. We obtained network topologies and stationary link probabilities from OPNET and provided these as inputs to our system. From the OPNET simulation runs, we observed that the length of the paths from a randomly picked source and destination varied between 5 and 8. We thus calculated the reliability values from our system when setting the deadline to 5, 6, 7, and 8. Comparative plots for each deadline value are shown in figures 6.4, 6.5, 6.6, and
6.7. In each of the cases, we observed that the oracle’s highest prediction is better than AODV and OLSR’s achieved performance. Specifically the best case predictions of the oracle always have higher values than what was achieved by either OLSR or AODV. This verifies the claim that our technique of modeling wireless networks yields the maximum achievable reliability for a given transmission.

It is interesting to note that for shorter path lengths (such as 6), there is a significant variation between the best and worst case estimate. In such cases, the performance of the routing algorithm would depend heavily on specific network scenarios, i.e. for an "unlucky" combination of link failures, the reliability could be much lower than the best case, irrespective of the routing approach.

We also observed that for lower deadline values, AODV performed poorly, but performed better than OLSR for higher deadline values. As a general observation, we found that packet delivery ratios in AODV were high, but the paths computed by the algorithm were usually longer than the ones from OLSR. Our technique can be useful to benchmark exiting routing protocols. For the comparison we carried out, it was seen that for shortest path lengths, the variation between best and worst case was large. Therefore, in such cases, routing may or may not be able to deliver high reliability, depending on specific sequences of link failures.

Using this approach, we observe specifics of routing protocols, such as packet delivery ratio under specific deadlines, to infer the performance of a routing protocol under those conditions. The model also provides the best case achievable delivery ratio for each condition. This in turn can be used to design more robust routing protocols.

6.5.2 Markov Link Probabilities

Simulations of variant 6.2.1

In this case, our system pre-computes paths to the destination using Markov probabilities. A link’s state at time $t$, thus depends on its state at the previous time instant. We then simulate networks, where these pre-computed paths are used to obtain packet delivery ratios. Each value
of the simulation run shown in Figure 6.8 was obtained from 1000 runs with the same source-destination pair. Since the Markov variation is more computationally expensive, we were not able to run the pre-computation for very large networks. For the limited sized networks, it can be seen that the computed best reliability values are always higher than the ones achieved from simulations.
Comparisons to existing routing protocols

We compared the predicted reliability from the oracle with OPNET simulations of OLSR and AODV, in a network with Markov links. An 8-node network with a randomly selected source-destination pair was used for this comparison. From OPNET simulations, it was observed that most of the paths were of length 4 for the cases where the packet was delivered to the
destination. We thus set the deadline to be 4 in our model, and discarded all paths from the OPNET simulations that used paths of other lengths. Figure 6.9 shows that both AODV and OLSR perform better than the average predicted value from our model for the small network of eight nodes. In this case, it is again observed that there is a significant difference between the predicted best and worst case reliability values.

Figure 6.6: Comparison between pre-computed paths from the model and OPNET simulations; Deadline = 8
Figure 6.7: Comparison between pre-computed paths from the model and OPNET simulations; Deadline = 4
Figure 6.8: Comparison between pre-computed paths and simulations of the network for Markov probabilities
Figure 6.9: Comparison between pre-computed paths from the model with Markov links and OPNET simulations; Deadline = 4
Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we investigated various techniques to increase reliability of transmissions in failure-prone wireless multihop networks. We reviewed the literature for existing multipath routing techniques, and proposed an approach called Petal routing that takes advantage of key features of ad hoc networks. The approach involves restricted flooding over a region of the network, by utilizing the broadcast nature of wireless nodes. We presented further enhancements on the basic approach, to reduce the number of transmissions without decreasing the reliability. We compared this method with a network coding approach from the literature as well as with a geographically diverse routing technique.

We then presented several ways to model and analyze wireless networks in order to help design robust routing techniques. The first modeling approach considers networks using Petal routing to compute the reliability of transmissions using a minimal cutset method. We tailor the method to model Petal routing closely. Comparisons between OPNET simulations of Petal routing and the analytical model show that the predicted values were consistently an upper bound of achieved reliability from transmissions.

In our effort to model wireless networks, we carried out empirical tests on the quality of
wireless links. Experiments were carried out with a two-node 802.11 network deployment in various outdoor wireless environments with varying levels of medium interference. We also carried out experiments on the OPNET simulator to measure the timescale of wireless link fluctuations. Our results show that no timescale correctly represents link fluctuations for all wireless environments and link strengths. For a specific interference model, we show how one can make empirical measurements to find the timescale of link fluctuations. To obtain fine grained timescale measurements, one would have to use more specialized equipment.

We presented a second method and a more general approach to model wireless networks and measure the performance of routing in such networks. In this approach, we model wireless networks as a stochastic dynamical system. This helps represent the dynamic nature of wireless links. We model links in the network as stationary probabilities and Markov transition probabilities. Our predictive system gives a reliability value for each combination of starting link states. We compare predictions from our model with simulations of existing routing protocols such as OLSR and AODV. The oracle in our system predicts the optimal reliability values. We observed that for shorter path lengths, there was a marked difference in the predicted best and worst case scenarios. Thus, even with a good routing algorithm, the achieved reliability could turn out to be much lower than the average case, for a given sequence of link failures.

7.2 Future Work

The contribution of this thesis is mainly focussed on finding novel techniques to increase reliability of transmissions in wireless networks and model such networks to identify vulnerabilities in existing routing algorithms. A general technique to model wireless link failures is using random graphs [22, 23, 25]. Applying the concept of random graphs to the analytical model for Petal routing, one could make use of strength and connectedness results from random graphs to predict reliability of packet transmissions.

Based on our work on wireless link measurements, we found that at the timescales explored, interference played a greater role in disrupting communication than noise. In the absence of
interference, we were not able to record a significant number of packet losses. Future directions on this research could include: (1) exploring much higher timescales by possibly modifying auto-rate functionalities in the MAC layer, and (2) measuring the achieved bit rate along with packet losses. For noisy communications where packet losses are low, the expected bit rates should be high.

Based on our linear systems model, some extensions can be made to better reflect the information available when routing choices are made, and more accurately represent the stochastic nature of link failures. Our current model considers a network whose links are likely to be available at any given time with a certain probability, and where routing decisions are made based on perfect knowledge of which links are up or down. Variations to this model to better represent practical networks could include:

7.2.1 Limited Node Information

Consider the case where the link transition probabilities are Markov, but each node makes forwarding decisions based on limited knowledge of the network. Each link is assumed to have a stationary distribution $\nu^m$, which will be the case for some simple conditions on $\mathbb{P}^m$, and all nodes are aware of the stationary distribution of each link.

The state can be redefined at time $t$ to be the location of the packet $n_t$, and the current link states that node $n_t$ is aware of, which will be denoted $q_t(n_t)$. For instance, if each node only knows the condition of its direct links connecting it to all 1-hop neighbors, then $q_t(n) = [q^{(n,j^1)}_t, \ldots, q^{(n,j^l)}_t]^T$ where $\{j^1, \ldots, j^l\}$ is the set of 1-hop neighbors of node $n$.

7.2.2 Delayed Node Information

In this case, each node again has limited information about the state of the network, but now in the form of delayed information. We preserve the assumption that the state of direct links (to 1-hop neighboring nodes) is known at the current time, but all other link states are not current, and their delay is correlated to their distance from the current node. The state at time
$t$ is defined as $x_t = (n_t, \tilde{q}_t(n_t))$, where $\tilde{q}_t(n_t)$ includes the most current information available to node $n_t$ on each link state.

For both these variants, calculating the transition function $\tau(x_{t+1} \mid x_t, u_t)$ is computationally expensive, and thus some approximation techniques have to be employed. A possible approximation with more feasible computation would be to combine variation 7.2.1 with the delayed information, so that each node only stores information about nearby links, but the information it stores is subject to delay.

### 7.2.3 Correlated Link States

Link probabilities are always considered independent of one another in the model we present. Future directions could include a more practical form of link model, where nodes located nearby in a spatial region affect one another. Then each link would depend on the state of other links close to it. This variant could be especially useful in modeling jammers that affect geographic regions in the network.
REFERENCES


123


[37] F. Kuhn, R. Wattenhofer, and A. Zollinger. An algorithmic approach to geographic routing


Record (Cat. No.01CH37240)*, pages 3201–3205, 2001. 10

[40] X. Li and L. Cuthbert. A reliable node-disjoint multipath routing with low overhead
in wireless ad hoc networks. *Proceedings of the 7th ACM international symposium on

[41] S. Low and P. Varaiya. Stability of a class of dynamic routing protocols (igrp). In *
FOCOM’93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and
IEEE, 1993. 11

[42] R. Madan and S. Lall. Distributed algorithms for maximum lifetime routing in wireless
11

*Wireless Communications and Mobile Computing, 6*(7):969–988, Nov. 2006. 10

127


129


130


Appendix A

Enumeration of Cutsets

In this section we discuss some existing techniques to compute all minimal cutsets of a given graph.

A.1 Computation of Minimal Node Cutsets

Enumerating all cutsets has been previously used in evaluating the reliability of networks [18, 26, 56, 66]. We use the minimal cut enumeration technique and tailor it to specifically model Petal routing. For networks with a source and sink, \((s, t)\) cuts are of special interest [55]. Our analytical model also uses \((s, t)\) cuts, where the cut disconnects the source and the destination. This is known to be an NP-Hard problem as shown by Provan and Ball [54]. The conventional approach to enumerating minimal cutsets is using set-space enumeration [4]. Using this concept, Tsukiyama et. al. proposed an algorithm to compute all minimal cutsets in linear time per cutset [72]. Note that a complete graph (or clique) has \((2^n - 1)\) cutsets, so this is still a very large number for well connected graphs. We show the use of some of these algorithms such as [33, 65] to enumerate all minimal cuts of a given graph.

Several approaches have been proposed to calculate cutsets of a graph [5, 33, 46, 57] to evaluate reliability of networks. Most of these approaches refer to a cutset as a set of nodes, which when removed, disconnect the graph. We refer to such a set as a node cutset. Such node
cutsets are more useful in modeling wired networks where node failures are more common. In wireless networks, however, link failures are more pertinent due to the uncertainty of the medium. In order to model link failures of a network, we compute cutsets consisting of a set of edges, rather than a set of nodes. We refer to such sets as edge cutsets. Note that it is computationally more expensive to enumerate edge cutsets compared to node cutsets, because a moderately connected graph has many more links than nodes. To compute edge cutsets, we first compute node cutsets using the algorithm from [33]. We then present some heuristic techniques to convert node cutsets to edge cutsets. Finally we discuss a new algorithm that we use to directly enumerate edge cuts [65].

To define the graph, let us consider graph $G = (V, E)$ to consist of a set $V$ of vertices and a set $E$ of edges whose elements are unordered pairs of vertices. The edge $e = (u, v) \in E$ is said to be incident with vertices $u$ and $v$, where $u$ and $v$ are the end points of $e$. These two vertices are called adjacent. The set of vertices adjacent to $v$ is written as $A(v)$, and the degree of $v$ is the number of vertices adjacent to $v$ and is denoted as $|A(v)|$. To enumerate all the minimal node cutsets of a given graph, we use the technique outlined in [33]. Based on this approach, we first represent the set of nodes as a minimal path tree starting at the sink node. For example, Figure A.1 shows a network and its corresponding minimal path tree. From the minimal path tree, we compute the set of minimal paths, i.e. all paths from the root of the tree to the source node. From the set of minimal paths, it is possible to compute the basic minimal paths for the network. According to [33], basic minimal paths are chosen from the set of minimal paths if they satisfy the following two conditions:

1. The minimal path is obtained from the tree without having to loop back to another branch (in example A.1, 1-2-3-4-5 is not a basic minimal path due to looping at node 4)

2. There is no direct link between any elements in a minimal path except with elements immediately preceding or following it

Once the basic minimal paths of a network have been enumerated, the set of all unique nodes
in the basic minimal paths are selected. Note that this set, called the basic minimal nodes, is a subset of all nodes of the network. The authors prove that the set of nodes obtained from the basic minimal pathsets is sufficient to determine all the minimal cutsets of the network. To find the minimal cutsets of the network, all combinations of basic minimal nodes are selectively removed from the network to check if the resultant network is disconnected. The authors provide a heuristic with hashing arrays for further elimination of combinations that do not form a cutset [33].

**A.2 Converting Node Cutsets to Edge Cutsets**

To convert a given node cutset to edge cutset we propose a technique. Consider the network shown in Figure A.2. \( S \) and \( D \) represent source and destination nodes of the network. Nodes 4, 5, 6 form a minimal node cutset, i.e., removing any one of these nodes would not make the set a cutset.

To convert the node cutset to an edge cutset, we carry out the following steps:

1. Remove all outgoing links from cutset nodes. In Figure A.2, this would indicate all the red (solid and dashed) links.
2. Traverse the graph using depth first search (DFS) algorithm, making nodes as *visited* once they have been traversed. Since all outgoing links of the cutset nodes have been removed, the DFS traversal reaches up to the cutset nodes (incoming links are active), and does not go any further.

3. Mark the remaining nodes of the network as *unvisited*.

4. All edges going from the set of *visited* nodes to *unvisited* nodes comprise the edge cut. Edges between cutset nodes are ignored. In example Figure A.2, this is the edge cut is indicated by the dashed red links.

Although the above steps result in a minimal cutset of the graph, it assumes that a given node cutset can result in only one edge cutset. This assumption is incorrect. Consider the network in Figure A.3.

Using the steps outlined above, we obtain the minimal edge cut $\{f, g\}$. However, we do not obtain other valid edge cuts such as $\{a, d, g\}$ and $\{c, e, f\}$. We observed that these *missing* edge cuts are more pronounced in topologies with clusters of nodes, where a local region has high connectivity and there are very few links between multiple such clusters. In order to obtain such cuts, we propose the following heuristics. Once a node cut has been obtained, the cut can be...
used to specify a local region to look for edge cuts. This is process is described by the following steps:

1. Generate combinations of all edges adjacent to cutset nodes

2. For each combination of edges check if it is a cut, by performing DFS traversal of the graph, and check if it is a minimal cut by verifying if a proper subset of this cut has already been identified as a minimal edge cut

We do not guarantee that these heuristics will allow us to obtain all minimal edge cuts. Indeed, we observe that cuts such as \( \{a, b, e, g\} \) in Figure A.3 would still not be enumerated with the above-mentioned heuristics. However, the heuristics help identify more failure scenarios to analyze with our model. Figure A.4 shows a sample network and the cuts edge cuts that were computed for the petal. Each edge cut is represented with a different color.

In the next section we use another existing technique to enumerate all minimal cutsets of a given graph.
A.3 Computation of Minimal Edge Cutsets

Enumerating all edge cuts of a graph is known to be an NP-Hard problem [54]. However, there exist many methods that take much less than the expected exponential time. In a graph $G = (V, E)$, a partition $(X, X')$ is defined as the two proper disjoint subsets of $V$. The complement of $X \subseteq V$ is denoted as $X' = V - X$. The induced subgraph $< X >$ is the graph $H = (X, F)$ where $F = \{(u, v) \in E | u, v \in X\}$. Note that the cutset problem can be alternately defined as a partition problem, where the goal is to find all partitions $X$ and $X'$ of a graph where $s \in X$ and $t \in X'$. In this section we present the use of an existing algorithm [65] to enumerate all
minimal edge cuts of a graph. We first define the terms used in the algorithm.

- **BFS Order**: Number assigned to each node, starting from 1, based on BFS traversal

- **Interior Vertex and Exterior Vertex**: For a graph $G$, a vertex $v$ is called an interior vertex if the graph $G/v = <V - v>$ is not connected, otherwise it is an exterior vertex

- **Cluster**: When we delete an interior vertex from graph $G$, more than one distinct component remains in the resulting graph. Each of such components is called a cluster and the deleted interior vertex is called the pivot vertex of these clusters

- **Absorbable Cluster**: A cluster in which all its vertices have a BFS order greater than the BFS order of its pivot is an absorbable cluster

- **Neighborhood Set** $\Gamma(v)$: The open neighborhood of a subgraph $X$ of $G$ is defined as $\Gamma(X) = \{v \in X'|(u,v) \in E \text{ for some } u \in X\}$

- **Edge Contraction**: A graph denoted as $G/uv$ is made by the contraction of an edge $uv$ in the graph $G$ in the following manner:
  
  - Delete vertices $u$ and $v$ in $G$ and replace them by a new contracted vertex $g$
  - Remove all edges incident to both $u$ and $v$ (i.e. $uv$)
  - For each edge incident to one of the vertices $u$ or $v$ (i.e. $uw$ or $vw$), there is an incident edge between $g$ and $w$ (i.e. $gw$) in $G/uv$

Algorithm 5 shows the steps to compute all minimal cuts as presented in [65]:

By the use of edge contractions the time complexity Algorithm 5 is reduced greatly. The time complexity of this algorithm is linear in the order of the number of cutsets. The proof of correctness for the algorithm has been provided in [65]. Note that with regard to minimal cuts there are approximation algorithms using the Monte Carlo approach as well [27]. Such approximation methods can also be used with the analytical model from Section 3.3.1.
Algorithm 5 Compute all minimal cutsets of a graph [65]

Input: graph $G$, seed vertex $v$, initially the source
Output: list of all cutsets $S(v)$ of $G$ that include vertex $v$
Initialization: $S(v) = \emptyset$, vertex list $F = \{v\}$, $dummyFlag = FALSE$

Find BFS ordering tree of graph $G$ with seed $v$ as its root.

$find\_all\_cutsets(G, v, F)$

if $v$ is an exterior vertex of the graph $G$ then
  if $F$ is not in $S(v)$ then
    add vertex list $F$ to $S(v)$
  end if
else
  find the set of clusters $clst$ for pivot vertex $v$
  if there is more than one inabsorbable cluster in $clst$ then
    return
  else
    if there is 1 inabsorbable cluster in $clst$ then
      $highest = \text{highest BFS order vertex among vertices of all absorbable clusters}$
      $dummyFlag = TRUE$
    end if
  end if
end if

find the neighborhood set $\Gamma(v)$ for vertex $v$
remove vertices in the vertex list $F$ from $\Gamma(v)$
remove vertices from $\Gamma(v)$ with BFS order lower than $v$
if $\Gamma(v)$ is empty then
  return
else
  for all vertices $u$ of $\Gamma(v)$ do
    copy vertex list $F$ to vertex list $H$
    contract the edge $(v, u)$ and find $G/uv$
    add the vertex $u$ to the vertex list $H$
    set the seed vertex $v$ to the contracted vertex $g$ of $G/uv$
    if $dummyFlag = FALSE$ then
      $find\_all\_cutsets(G/uv, v, H)$
    else
      if $v = highest$ then
        $dummyFlag = FALSE$
      end if
    end if
  end for
end if