TERMITE:
A SWARM INTELLIGENT ROUTING ALGORITHM FOR MOBILE WIRELESS AD-HOC NETWORKS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Martin Heinz Roth

January 2005

TERMITE:
A SWARM INTELLIGENT ROUTING ALGORITHM
FOR MOBILE WIRELESS AD-HOC NETWORKS

Martin Heinz Roth, Ph.D.

Cornell University 2005

A biologically inspired algorithm is presented to route messages in mobile wireless ad-hoc networks. Such computer networks are primarily characterized by their quickly changing topologies due to frequent node mobility. The principles of swarm intelligence are used to define a probabilistic algorithm for which routing through paths of maximum utility is an emergent property. This adaptive algorithm, dubbed *Termite*, uses stigmergy to reduce the amount of control traffic needed to maintain performance. Strong routing robustness is achieved through the use of multiple paths; each packet is routed randomly and independently. Once the basic operation of Termite is verified, alternative metric estimation techniques are tested via simulation. Optimal system parameters are selected by testing over orders of magnitude. A simple analytical model is built in order to explain the simulation results. The model also used to propose two heuristics for determining the optimal pheromone decay rate. All of the enhancements to Termite, now known as *ReTermite*, are consolidated and tested against Ad-hoc On-demand Distance Vector (AODV), a leading ad-hoc routing algorithm. ReTermite is shown to be superior in many primary metrics and the reasons for this explained. Previous heuristic models are also compared to simulation results.

BIOGRAPHICAL SKETCH

Martin Heinz Roth was born in Malaysia on October 18, 1978 as the first child to Heinz and Erika Roth. After learning to sit in Bali, Indonesia, Martin and his family returned to their native Switzerland in 1979. Shortly before Martin's fourth birthday, the Roth family (with the new addition of sister Anita Erika) moved to the United States of America and settled in Michigan. Martin attended Warwick Pointe Academy from 1983 until 1989, when the family (now one larger with Gerwin Niklaus) moved to North Carolina. Martin attended Charlotte Country Day School until graduating with a high school diploma and an International Baccalaureate (IB) degree in 1996. Departing the sunny south for reaches further north, the next stop on the educational soul train was Cornell University in New York. Majoring in Electrical Engineering and concentrating in electromagnetics, signal processing, and communications, Martin graduated cum laude and with honors in 2000 only to return to the same institution to begin graduate studies under Dr. Stephen Wicker.

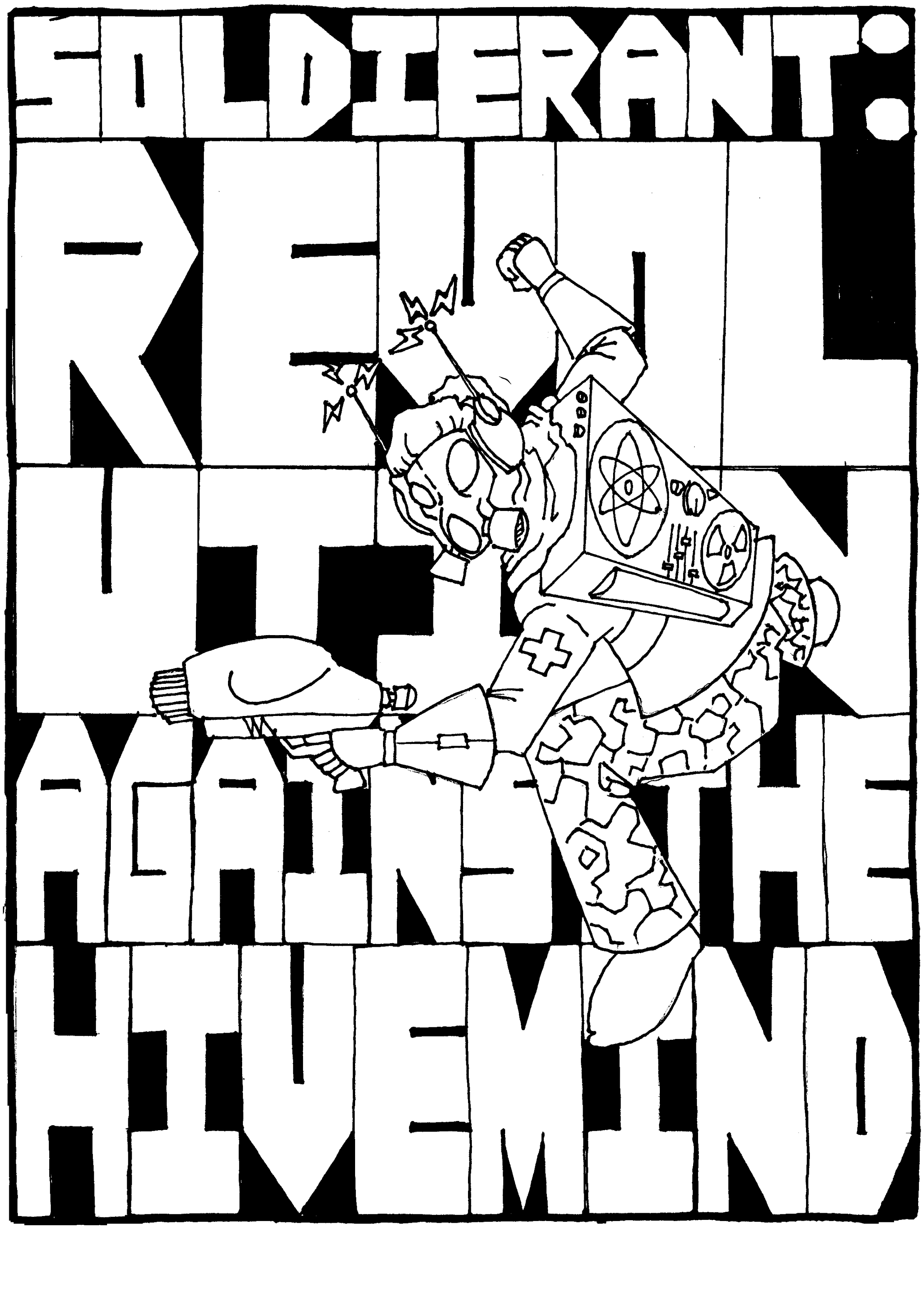

Figure 1: Soldier Ant: Revolution Against The Hive Mind [82]

ACKNOWLEDGEMENTS

The work presented in this thesis was made possible with the generous support of DARPA ITO under the project name, “Self-Configuring Wireless Sensor Networks,” and of NSF - CRCD, Microsoft, and Hewlett Packard under the project title, “The Ad Hoc Classroom: Integrating Emerging Wireless Communications and Networking Technologies into Mainstream Computer Science and Electrical Engineering Curricula.”

TABLE OF CONTENTS

1 Introduction **1**

1.1 Introduction . . . 1

1.1.1 Motivation . . . 2

1.2 Swarm Intelligence . . . 3

1.3 Mobile Wireless Ad-Hoc Networks . . . 4

1.4 Termite . . . 5

1.5 Pheromone Update in SI MANETs . . . 6

1.6 Analysis of Pheromone Update in SI MANETs . . . 7

1.7 ReTermite . . . 7

1.8 Conclusion . . . 8

2 Swarm Intelligence **9**

2.1 Introduction . . . 9

2.2 The Principles of Swarm Intelligence . . . 10

2.3 Swarm Intelligent Models of Biological Behavior . . . 13

2.3.1 Ant Path Optimization . . . 13

2.3.2 Termite Hill Building . . . 16

2.4 Applications of Swarm Intelligence . . . 18

2.4.1 Ant Colony Optimization . . . 18

2.4.2 Other Examples . . . 22

2.5 Relationship to Artifical Intelligence . . . 24

2.5.1 Reinforcement Learning . . . 24

2.5.2 Neural Networks . . . 26

2.5.3 Gradient Descent . . . 27

3 Mobile Wireless Ad-Hoc Networks **28**

3.1 Introduction . . . 28

3.2 Classic Network Routing . . . 29

3.2.1 Flooding . . . 29

3.2.2 Distance Vector Routing . . . 30

3.2.3 Link State Routing . . . 32

3.2.4 Discussion . . . 34

3.3 Introduction to Ad-Hoc Networking . . . 34

3.3.1 Applications . . . 35

3.3.2 Challenges . . . 37

3.3.3 Routing Metrics . . . 38

3.3.4 Performance Metrics . . . 39

3.3.5 Medium Access . . . 39

3.4 Traditional Ad-Hoc Routing Solutions . . . 40

3.4.1 Proactive Routing . . . 41

3.4.2 Reactive Routing . . . 43

3.5 Swarm Intelligent Routing . . . 47
3.5.1 Wired Networks . . . 47
3.5.2 Wireless Routing . . . 57
3.5.3 Discussion . . . 64
3.6 Related Approaches to Routing . . . 65
3.6.1 Agent-Based Routing . . . 66
3.6.2 Probabilistic Routing . . . 67
3.6.3 Q-Routing: Routing with Reinforcement Learning . . . 69

4 Termite **70**
4.1 Introduction . . . 70
4.2 Design of Termite . . . 70
4.2.1 The Pheromone Table . . . 71
4.2.2 Pheromone Update . . . 72
4.2.3 Pheromone Decay . . . 72
4.2.4 Pheromone Bounds . . . 73
4.2.5 Route Selection . . . 74
4.2.6 Packet Design . . . 75
4.3 Properties of Termite . . . 77
4.3.1 Probabalistic Routing . . . 77
4.3.2 Best-Effort Service . . . 78
4.3.3 Low Complexity . . . 78
4.3.4 Low Route Recovery Latency . . . 79
4.3.5 Adaptability . . . 79
4.3.6 Multipath Routing . . . 80
4.3.7 Stigmergy . . . 81
4.3.8 Mobile Agents . . . 81
4.3.9 Promiscuity in a Broadcast Medium . . . 81
4.3.10 Energy Consumption . . . 83
4.3.11 Directional Links . . . 83
4.4 Simulation and Results . . . 83
4.4.1 Simulation Environment . . . 83
4.4.2 Simulation Results . . . 86
4.4.3 Discussion . . . 88
4.5 Conclusion . . . 91

5 Pheromone Update in SI MANETs **92**
5.1 Introduction . . . 92
5.2 Termite . . . 93
5.2.1 Pheromone Accounting . . . 94
5.2.2 Packet Forwarding . . . 96
5.3 Pheromone Update Techniques . . . 99
5.3.1 Classic Pheromone Filter . . . 100
5.3.2 γ Pheromone Filter . . . 100

5.3.3 Averaging Filter . . . 101
5.3.4 Normalized γ Pheromone Filter . . . 101
5.3.5 Probabilistic Bellman-Ford . . . 102
5.3.6 Ant-Based Control + X . . . 102
5.3.7 Box Filter . . . 103
5.3.8 Oracle . . . 103
5.3.9 Random Routing . . . 104
5.4 Simulation and Analysis . . . 104
5.4.1 Environment and Parameters . . . 104
5.4.2 Evaluation Metrics . . . 105
5.4.3 Simulation with 50 Nodes . . . 107
5.4.4 Simulation with 100 Nodes . . . 110
5.4.5 Parameter Analysis of F and τ . . . 113
5.4.6 Analysis . . . 115
5.5 Conclusion . . . 117

6 Analysis of Pheromone Update in SI MANETs 119
6.1 Introduction . . . 119
6.2 The Model . . . 120
6.3 Pheromone Update Analysis . . . 121
6.3.1 Single Link Pheromone . . . 122
6.3.2 Two Link Pheromone . . . 126
6.3.3 Analysis . . . 132
6.4 An Optimal Pheromone Decay Rate Heuristic . . . 135
6.4.1 Maximum Pheromone Decay . . . 136
6.4.2 The Pheromone Filter Cutoff . . . 138
6.4.3 Heuristic Comparison . . . 140
6.5 Conclusion . . . 142

7 ReTermite 144
7.1 Introduction . . . 144
7.2 ReTermite . . . 144
7.2.1 Pheromone Table . . . 145
7.2.2 Packet Forwarding . . . 146
7.2.3 Pheromone Accounting . . . 147
7.2.4 Route Discovery . . . 150
7.2.5 Route Repair . . . 151
7.2.6 Packet Structure . . . 151
7.3 Simulation . . . 153
7.3.1 Simulation Environment and Parameters . . . 153
7.3.2 Evaluation Metrics . . . 154
7.3.3 Results and Analysis . . . 157
7.3.4 Discussion . . . 164
7.4 A Pheromone Decay Rate Heuristic . . . 165

7.4.1 τ^* Simulation . . . 166
7.4.2 τ^* Heuristic Analysis . . . 167
7.5 Conclusion . . . 169

8 Conclusion 170
8.1 Introduction . . . 170
8.2 Original Contributions . . . 170
8.2.1 The Termite SI MANET Routing Algorithm . . . 171
8.2.2 Evaluation of Pheromone Accounting . . . 173
8.2.3 Pheromone Decay Rate Heuristics . . . 173
8.2.4 ReTermite . . . 174
8.3 Future Work . . . 176
8.3.1 Termite . . . 177
8.3.2 Parameter Heuristics . . . 180
8.3.3 Networking with Linear Systems . . . 181
8.3.4 Artificial Intelligence . . . 181
8.4 Final Remarks . . . 181

A Pheromone Update Simulation Results 182

Bibliography 191

LIST OF TABLES

4.1 Simulation Parameters 84
4.2 Mean Link State Changes per Second per Node 89

6.1 Calculated Link Expiry Rate 142

LIST OF FIGURES

1 Soldier Ant: Revolution Against The Hive Mind [82] iv

2.1 Example of Path Optimization in Ant Foraging Behavior [9] 14

3.1 Example Network Diagram [24] 29

4.1 Example of a Pheromone Table 72
4.2 Data Goodput vs. Control Overhead vs. Average Node Speed . . . 86
4.3 Mean Packet Path Length vs. Mean Node Speed 87
4.4 Route Confidence vs. Mean Node Speed 88
4.5 Normalized Distance to Requested Destination vs. Mean Node Speed 89

5.1 Link Change Rate for 50 and 100 Node Simulations 107
5.2 Data Goodput vs. Node Speed (50 nodes) 108
5.3 Path Ineffiency vs. Node Speed (50 nodes) 110
5.4 Delivery Efficiency vs. Node Speed (50 nodes) 111
5.5 Data Goodput vs. Node Speed (100 nodes) 112
5.6 Path Ineffiency vs. Node Speed (100 nodes) 113
5.7 Delivery Efficiency vs. Node Speed (100 nodes) 114
5.8 τ^* vs. Node Speed (50 Nodes) 115
5.9 F^* vs. Node Speed (50 Nodes) 116

6.1 Diagram of the MANET Model 120
6.2 Goodput vs. Pheromone Update Method 122
6.3 Delivery Efficiency vs. Pheromone Update Method 123
6.4 Link Pheromone vs. Packets Received using the γ Pheromone Filter 128
6.5 Link Pheromone vs. μ_0 vs. K 129
6.6 Link Probability vs. μ_0 vs. K 130
6.7 Link Pheromone vs. Packets Received using the $\bar{\gamma}$ Pheromone Filter 131
6.8 Pheromone Decay Rate Heurisitcs 140
6.9 Pheromone Decay Rate Heurisitcs 141

7.1 Data Goodput vs. Node Speed 157
7.2 Average Path Length vs. Node Speed 158
7.3 Control Overhead vs. Node Speed 159
7.4 Packet Type Distribution vs. Node Speed 160
7.5 Medium Load vs. Node Speed 161
7.6 Medium Efficiency vs. Node Speed 162
7.7 Medium Inefficiency vs. Node Speed 163
7.8 Link Failure Rate vs. Node Speed 164
7.9 End-To-End Delay vs. Node Speed 165
7.10 Performance of NγPF @ 1 m/s vs. Pheromone Decay Rate 166
7.11 Performance of NγPF @ 10 m/s vs. Pheromone Decay Rate 167
7.12 Comparison of NγPF τ^* Heuristic and Simulation 168

A.1 Delivery Efficiency of γPF vs. F vs. τ (50 Nodes) 183
A.2 Delivery Efficiency of $\bar{\gamma}$PF vs. F vs. τ (50 Nodes) 184
A.3 Delivery Efficiency of pBF vs. F vs. τ (50 Nodes) 185
A.4 Delivery Efficiency of Classic PF vs. F vs. τ (50 Nodes) 186
A.5 Delivery Efficiency of Box Filter vs. F vs. τ (50 Nodes) 187
A.6 Delivery Efficiency of IIR Averaging Filter vs. F vs. τ (50 Nodes) 188
A.7 Delivery Efficiency of Oracle vs. F vs. τ (50 Nodes) 189
A.8 Delivery Efficiency of Random Routing vs. F vs. τ (50 Nodes) . . 190

CHAPTER 1

INTRODUCTION

1.1 Introduction

This thesis introduces, analyses, and develops a method for routing in mobile wireless ad-hoc networks (MANETs) which is designed following the principles of swarm intelligence. Packets are used to passively disseminate information about the network while choosing each next-hop probabilistically according to local link utility estimates. The routing algorithm is known as *Termite*.

The work is introduced with a review of all necessary background and previous work in the general fields of swarm intelligence and network routing. This develops an understanding of the foundations upon which Termite is built and how it extends the state-of-the-art. The Termite algorithm is introduced as a complete solution to the MANET routing problem, along with a discussion of the merits of a probabilistic routing approach and simulations to demonstrate its viability. The algorithm contains many parameters and it is not generally known what values will yield the best performance. Following chapters are dedicated to an analytical and simulation-based analysis of the parameter space in order to determine performance and parameter tradeoffs. The thesis is concluded with a resimulation of Termite updated with all of the knowledge gained from the preceding chapters. These simulations are compared against the Ad-hoc On-demand Distance Vector (AODV) routing algorithm, a highly regarded solution from the state-of-the-art.

1.1.1 Motivation

Current wireless communications technology is an advanced form of a basic concept which has existed for several decades. The basic model includes a single transceiver pair communicating over some distance. For example, the current cellular communications systems which are extensively deployed have a number of wire networked base stations, each of which services a specific area of wireless terminal (cellular telephone) users. If users should happen to move outside of the network's coverage area then all service is lost. Ad-hoc technology seeks to remove this constraint and extend the range of computer communications far beyond the range of a single access point. This will be accomplished by allowing nodes to act as routers in addition to their usual role as terminals. The promise of this technology is great, however no standard exists to facilitate the deployment of production ad-hoc networks. One of the greatest difficulties in reaching this goal is that the ad-hoc environment offers a number of challenging situations which network designers have not yet been able to overcome or even properly characterize. The most serious of these is node mobility. Because users are able to move freely, the topology of the network can change significantly over a short period of time. It is a serious technical challenge for the network to recognize changes and automatically adapt. Oftentimes so much control traffic is generated to determine the state of the network that it is unable to successfully deliver any user data.

There exists an interesting analogy to the world of biology which may shed some light on more applicable solutions to the network routing problem. Social insect colonies exhibit many of the characteristics that networks should ideally have. For instance, they are completely distributed in their operation, they are robust against interference and component misfunction, they are adaptable, and they are composed of simple individuals. One might think of the relation between an ant and an ant colony, as well as how an ant colony is able to adapt to its surroundings in order to find food or to protect against predators.

By making simple analogies between ants and nodes, and ant colonies and computer networks, a formalism can be created in order to study how the characteristics of the former may be transferred to the latter. The work presented in this thesis develops the formalism of the biological analogy beyond previous work. The results progress engineering efforts in the field of mobile wireless ad-hoc networks.

1.2 Swarm Intelligence

Swarm Intelligence (SI) is a framework for designing systems of simple interacting individuals. It is often coupled with the study of biological systems exhibiting the characteristics of swarm intelligence, such as social insect societies. Characteristics of interest include the interaction of many individuals following a predetermined set of simple rules. Chapter Two provides a description of SI's four design principles and demonstrates qualitatively how they can be used to structure solutions to various problems. The principles are positive feedback, negative feedback, multiple interactions, and randomness. The use of stigmergy and emergence are also reviewed. Examples of behavior by social insects, such as ants, social bees, termites, or social wasps, are used to illustrate the use of the design principles. Once an understanding is established of how swarm intelligence works on an intuitive level, several engineering applications are given. SI has been most famously applied to the fields of optimization, robotics, and artificial intelligence. The optimization applications will be discussed in some detail. Their formalism will be reused to solve the MANET routing problem.

1.3 Mobile Wireless Ad-Hoc Networks

Mobile wireless ad-hoc networks (MANETs) are a collection of mobile computers which are able to wirelessly communicate with each other. Each computer forwards messages, or packets, in order to deliver them across distances larger than the range of a single communications link. This field encompasses a large range of topics, including many of those of its fifty year old parent, computer networks. One of the most pressing is the routing problem; which nodes should forward a data packet on its way to the destination? The problem is compounded because nodes are mobile; the network topology changes often and requires the routing algorithm to be able to find new routes quickly.

A large number of proposed routing algorithms have been published. They are generally based on traditional routing solutions to wired networks. Both proactive and reactive protocols are proposed. Examples of both types are given in order to provide a good understanding of well-known approaches to the field. Swarm intelligent solutions are then reviewed, both for wired and wireless networks. The variety and similarities between approaches will be highlighted, and a general implementation for SI routing is revealed as well.

Lastly, some approaches related to swarm intelligent routing are explained. These are ideas which come close to SI routing, but are not quite the same. This includes probabilistic routing, gossiping, routing with agents, and routing with reinforcement learning. The end of Chapter Three concludes the review of previous work. Future chapters will be concerned only with the development and analysis of the Termite routing algorithm.

1.4 Termite

Chapter Four introduces the subject of this thesis. Termite is a swarm intelligent routing algorithm for mobile wireless ad-hoc networks. Based on previous work in the field and the termite hill building analogy, Termite is designed to remedy many of problems commonly associated with MANET routing algorithms. Primarily, Termite aims to reduce the amount of control traffic necessary to maintain acceptable performance in the network. The total number of transmissions necessary to deliver a data packet should be minimized. One way in which this is done is to institute a local route repair scheme where packets are simply forwarded to another neighbor if the current communications link fails. If a node is entirely unaware of a destination, then a route discovery process is initiated. Unlike the traditional approach of flooding the network to search for a route, Termite takes advantage of the broadcast communications environment and uses a random walk to find routes. This approach manages to find routes to most destinations while avoiding the cost of many packet transmissions.

There are many considerations when using a swarm intelligent algorithm such as Termite for network routing. Termite's probabilistic approach departs from standard routing practice, which is based on deterministic algorithms on graphs. Instead, Termite depends on a global routing solution as an emergent behavior, the result of the interactions of many packets. Each of these packets collects and distributes information about the network, known as pheromone, as it moves through the network; exploring while traveling. A disadvantage of this approach is that packets may lose themselves while exploring and die in the network. Termite can only be considered a best-effort routing service and requires a reliable transport layer implemented on top of it.

Termite is shown to be an effective routing algorithm based on simple rules following the principles of swarm intelligence. The pheromone update strategies are controlled by positive and negative feedback, and the behavior of individual packets is random and dependant on multiple interactions.

1.5 Pheromone Update in SI MANETs

Chapter Four shows that Termite can function effectively in the ad-hoc environment. There are many parameters and other procedural issues for which the optimal setting is unknown. This is like having a car without a steering wheel; the algorithm works but it is not generally known how to control it. Chapter Five will first offer some updates to Termite based on previous work. This includes true continuous pheromone decay and source pheromone repel in the packet forwarding equation. The new algorithm is then tested rigorously in order to determine the effect of various pheromone update techniques and parameter settings. No such comparison has been performed before. The results of this chapter will show that the traditional method and its variants, generally known as pheromone filtering, perform the best out of a comprehensive list. The list includes techniques based on the combination of many ideas including time-based and time-independent pheromone decay, joint link pheromone estimation or link independent pheromone estimation, pure random routing, and routing with perfect information. The pheromone sensitivity and pheromone decay rate parameters are also varied in order to gain an intuition on their relative effect on the global routing performance. In general, high sensitivities and moderate decay rates are preferred. The results will set the stage to improve the performance of Termite.

1.6 Analysis of Pheromone Update in SI MANETs

With a host of empirical data from Chapter Five, the next step is to create an analytical model which can reproduce the results of the simulation tests. Another interest is to be able to extend on those results and generalize with the analytical models. Chapter Six develops a number of conclusions. The first is an explanation for the performance of two of the best pheromone update techniques tested in the previous chapter's simulations, the γ pheromone filter and the normalized γ pheromone filter. This is basically explained by the differing amounts of pheromone maintained on active links. The second conclusion is the development of two heuristics to determine the optimal pheromone decay rate. The pheromone decay rate is a measure of how long network information is retained in order to make routing decisions. The first model is based on the amount of time necessary to decay the average amount of pheromone found on a link. The second model determines this value based on a linear filtering perspective of the information update procedure. A decay rate is determined from the cutoff frequency of the pheromone filter.

1.7 ReTermite

All of the experience gained from working with Termite over the course of the thesis is brought together in order to update the routing algorithm and show how well it can really perform. Termite is enhanced with source pheromone repel, the normalized γ pheromone filter, probabilistic bellman-ford, true continuous pheromone decay, and the optimal pheromone decay heuristics from Chapter Six. The new algorithm is referred to as *ReTermite* and this chapter provides a complete redefinition. It is tested against AODV, a leading standards-track MANET routing algorithm. ReTermite is shown to be able to deliver more packets more efficiently, especially at low speeds. However, it suffers from a larger end-to-end delay as it spends more effort to bring hard-to-deliver packets to their destination. A comparison is also made to the decay rate heuristics with simulation results. These show that ReTermite performance is heavily influenced by the proper parameter selection, and also that the decay rate chosen for the majority of the simulations was significantly suboptimal.

1.8 Conclusion

The final chapter will conclude the thesis with a discussion of the original contributions to the state-of-the-art of the fields of swarm intelligence and ad-hoc networking developed by this work. These are numerous and include the development of Termite, the direct comparison of several different pheromone update methods, the characterization of parameter influence, the extension of an analytical model to analyze pheromone dynamics, the development of the pheromone filter and associated pheromone decay rate heuristics, and the creation of ReTermite and comparison with AODV.

Future work is discussed based on the development of Termite. Suggested are additional effort on ReTermite to allows its use with asymmetric links, better developed and articulated parameter heuristics, new ideas for working on the network routing problem with tools from linear systems, and also some trading of analysis concepts with artificial intelligence, such as reinforcement learning.

The chapter and the thesis are concluded with some final remarks.

CHAPTER 2

SWARM INTELLIGENCE

2.1 Introduction

Swarm Intelligence (SI) is a framework for designing and analyzing large scale systems composed of many simple locally interacting individuals [1]. The net result of their actions is an emergent property of the system. The emergent behaviors may be desirable or undesirable to the system designers, however they are the result of the interactions between individuals, and not the result of a preprogrammed or centrally controlled choreography.

This methodology is inspired and based upon observations of the behavior of social insects such as ants, termites, social bees, or social wasps. These biological systems embody many of the principles that man-made systems should have. They are composed of a large number of simple and cheap components (workers), cooperating locally and independently to accomplish a global task that any individual could not do alone. Colony behavior is often robust against a large number of parameters such as individual misbehavior or loss. Such systems are also able to adapt to the environment as is necessary.

Swarm intelligence is often considered a subfield to Artificial Life (AL) [2]. Artificial life is the study of designing machines with traits that mimic those of biological counterparts. This is also referred to as biomimetics. Where SI is primarily concerned with studying large systems of interacting individuals, AL is broader and also looks at questions of self-replication, evolution, natural system modeling, or borrowing ideas from nature in order to enhance engineering design. Both SI and AL are generally thought to be subsumed by the even broader field of Artificial Intelligence (AI), which is ultimately concerned with understanding "intelligent" behavior on all levels [3].

2.2 The Principles of Swarm Intelligence

There are four primary principles of the swarm intelligence design framework. They describe how systems can be created to exhibit emergent properties by taking advantage of the interactions of many individuals. The principles are positive feedback, negative feedback, randomness, and multiple interactions. Each of these principles describes a critical component of a swarm intelligent system, required to elicit emergent behavior from a large group of simple individuals. An additional principle, stigmergy, is also mentioned here due to its importance to the process, however it is not explicitly required in all SI systems. The following paragraphs introduce a qualitative explanation of each principle. Afterwards some examples will be introduced in order to clarify the role of each.

Positive feedback is used to reinforce good solutions present in the system. When a particular solution is found to be better than others, at least locally, a positive feedback mechanism is needed in order to encourage the use of that solution over all others.

Negative feedback is responsible for removing old or poor solutions from the system. Coupled with the use of positive feedback, the system tends to use only the best solutions available at any given time. It is important that positive and negative feedback be carefully balanced. If the former outweighs the latter, too many solutions will exist in the system and there will not be a clear indication as to how a problem should be solved. If the latter outweighs the former, then solutions will die out quickly and the problem cannot be solved at all.

There is a need to test different solutions as they become available or change in quality. The randomness principle enables this by allowing a swarm intelligent algorithm to explore options at random. This is an effective strategy when the number of possible solutions is large or the location and development of a good solution is unpredictable.

Swarm intelligent algorithms rely on the multiple interactions of the individuals in the system. Some amount of information that has been gathered locally may be communicated in each interaction. Consequently, information from one portion of the system may be transmitted to another. Multiple local interactions also allow for events to be coordinated on a local scale. In an ad-hoc network, each node may coordinate local routing information with neighbors during each communication.

Another perspective is to insist that swarm intelligence requires large populations of participating individuals. Because individual behaviors are often randomly chosen, it is necessary to have many interactions in order to reliably determine the information necessary to decide on a proper course of action. Having many individuals will increase the interaction rate and make locally sensing system parameters more reliable. Natural SI systems, such as social insects, often contain anywhere from tens to millions of individuals.

Stigmergy is a method of indirect communication through effects on the environment of the behavior of an individual, which another will use in deciding what to do; it is the use of an outcome of previous work to guide current work. This term was first introduced by Grassé in 1959 in order to describe a communication mechanism of termites engaged in nest construction [4]. For instance, while ants can communicate directly by feeling each other with their antennae, they also share information via effects on their shared environment. This is often done through the use of an excreted volatile chemical compound called pheromone. Different types of pheromone are used, varying concentrations of which will elicit different behaviors. Another example is with bees, who may decide to gather nectar or store food based on statistical information on the availability of food or food storers.

Stigmergy can become a useful communications channel as it offers a means to eliminate possibly expensive or complex explicit and direct communication between individuals. Moving communications to the environment can oftentimes maintain state for free and even increase the communications robustness; information transfer can be made asynchronous between individuals. The environment acts as a convenient broadcast medium for information, since all individuals exist and interact with it.

There are two types of stigmergy, passive and active. The former is often associated with the physical environment which will force a particular action despite the agent's intentions. The latter is the type of stigmergy that is considered in this thesis; the environment simply influences an agent's choice of behavior which it then carries out [5].

Emergent behavior is often the intended product of a swarm intelligent system. Individuals are intentionally simple, with the expectation that the net result of their many interactions will produce the desired behavior of the system. Emergence can be summarized by the intuition that "the whole is greater than the sum of the parts." A prime example of this principle is the behavioral differences in biological examples such as between ants and ant colonies or bees and bee colonies. The concept of emergence has also captured the attention of other fields, including philosophy, mathematics, and music [6].

2.3 Swarm Intelligent Models of Biological Behavior

This section will describe two common examples of natural emergent behavior which can be explained with the swarm intelligent framework. In practice the relationship is often reversed, where a natural system is analyzed and then modeled using SI. The two examples are the path optimization behavior of ants and the hill building behavior of termites. These examples will show how SI models can be created, giving a strong foundation for putting engineering problems into the SI framework. Examples of bees and wasps and further details may be found in [1] and [7].

2.3.1 Ant Path Optimization

A classic SI model is that of the path optimizing behavior of ants. It is widely used to motivate applications such as routing in computer networks and optimization. Ants have been shown to adapt to their environment and always find the most efficient path to their food source [8]. This is possible even when a shorter path becomes available at a later time. The problem requires a global solution since all paths must be continuously explored. These paths are often many meters long, however each individual ant can only interact locally with its surroundings. Ants must cooperate in order to find the shortest path. A solution to this problem is explained below.

Consider a number of ants all traveling between their nest along a single available path to a food source (Figure 2.1a). As each ant returns with food, it deposits a small amount of pheromone on the trail. Pheromone is a volatile chemical to which ants are attracted. It evaporates over time, requiring that new pheromone be continuously laid or else it will disappear completely and the trail will vanish. Ants leaving the nest are attracted to the pheromone trail laid by returning ants. They are biased to follow it when they leave the nest looking for more food. This example is uninteresting with only a single path. When an unevenly shaped obstacle is placed between the nest and the food source, the ants are forced to choose a way around it (Figure 2.1b). Because no pheromone exists around the obstacle, ants travel in equal proportions around each side until resuming the original pheromone trail. Since it takes longer for returning ants to travel around the longer side of the obstacle, pheromone has more time to decay in between ant arrivals (Figure 2.1c). The weakened pheromone trail along the longer route around the obstacle is offset by the relatively strong trail on the shorter route; ants are able to travel that route more frequently and thus reinforce the pheromone trail there more often. Since ants are attracted to stronger pheromone trail, the shorter trail is eventually favored by nearly all ants over the longer trail (Figure 2.1d). The global optimization problem is solved. The ants adapt to a new environment and are able to choose the shortest path.

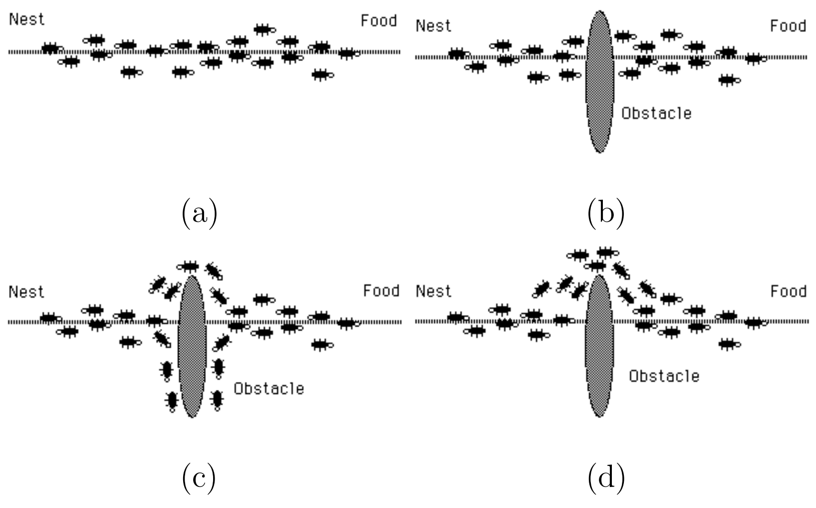

Figure 2.1: Example of Path Optimization in Ant Foraging Behavior [9]

Swarm Intelligent Framework

The attraction of ants to pheromone represents the positive feedback in this swarm intelligent system. A stronger pheromone trail is built on the shorter path, and this allows ants to be drawn away from the suboptimal solution. As more ants are attracted to the shorter path with the stronger pheromone trail, its intensity grows even stronger, thereby attracting even more ants.

Conversely, pheromone evaporation limits the amount of pheromone that can be placed on a trail. In the beginning of the example there are equal ants on both routes around the obstacle, however the longer path is doomed to obsolescence because the ants cannot possibly put enough pheromone on it fast enough to overcome the bias of the shorter path. This effect is due to a combination of the pheromone decay rate and the number of ants on that trail. Thus, pheromone decay represents negative feedback. Old or unacceptable solutions, the longer path, are slowly removed from the system.

This solution to the problem depends on both the many interactions of ants over time and their random decisions. Many ants are required in order to test the available paths to the food source. Each successive trip is a biased random choice by each ant. Each ant may have a different experience as it travels between the nest and the food source, and so these different experiences must be averaged out over time. The pheromone decay mechanism helps in this regard, however it is necessary for many ants to try the same path in order to accomplish this. Each trip represents a statistical test of the system. The sum of all trips and decisions by each ant is the eventual choice of the shortest path by the group.

The use of pheromone to communicate path quality is an example of active stigmergy. There is no need for the ants to communicate directly. All of the necessary information needed to solve the shortest path problem is resident in the pheromone laid on the ground.

Ultimately, the act of each outgoing ant choosing the stronger pheromone trail yields an emergent shortest path algorithm. Ants are able to adapt to new environments and then find the shortest path not by any sort of explicit instruction, but simply by being biased towards the pheromone gradient.

2.3.2 Termite Hill Building

The hill building behavior of termites is another example of how a large population of individuals can cooperate to solve global tasks. In this case, termites want to gather pebbles spread over an area into one place in order to build a hill. Individuals act independently and move only on the basis of the observed local pheromone gradient.

Each termite follows four rules. It is biased towards the locally observed pheromone gradient as are the ants. If no pheromone exists, a termite moves uniformly randomly in any direction. Each termite may carry only one pebble at a time. If a termite *is not* carrying a pebble and it encounters one, the termite will pick it up. If a termite *is* carrying a pebble and it encounters one, the termite will put the pebble down. The pebble will be infused with pheromone which then evaporates and creates a gradient for others to follow. With these rules, a group of termites can collect dispersed pebbles into one place.

Swarm Intelligent Framework

Positive feedback is represented by a termite's attraction towards the pheromone gradient. The termite is biased to add more pebbles to large piles. The larger the pile, the more pheromone it is likely to have, and more pebbles will be moved to it. The greater the bias to the hill, the faster termites are likely to arrive, further increasing the pheromone content of the hill.

As the pheromone evaporates, it consequently weakens and lessens the resulting gradient. A diminished gradient will attract fewer termites as they will be less likely to move in its direction according to the gradient. While this may seem detrimental to the task of collecting all pebbles into one pile, it is in fact essential. As the task begins, several small piles will emerge. Those piles that are able to attract more termites will grow faster. As pheromone decays on lesser piles, termites will be less likely to visit them again, thus preventing them from growing. Once all of the pebbles in the small piles have been picked up (by chance, which may well take a *long* time in this example), that pile will cease to exist and can never grow again; larger piles will grow instead. Negative feedback, in the form of pheromone decay, helps large piles grow by preventing small piles from continuing to attract termites.

It is essential that many individuals work together at this task. If not enough termites exist then the pheromone would decay before any more pebbles could be added to a pile. Termites would continue their random walk without forming any significant piles. Where and when piles are created or destroyed is determined entirely by chance since each termite makes independent and probabilistic decisions.

As in the ant example, termites use pheromone to coordinate their activities. This is an example of stigmergy. Termites are directed to the largest hill by the pheromone gradient. There is no need for termites to directly communicate with each other or even to know of each other's existence. The rule set is greatly simplified by allowing each individual to act independently of all others.

2.4 Applications of Swarm Intelligence

The previous section has reviewed how biological systems can be cast into the swarm intelligence design framework. This section will describe how these same principles and observations can be used to engineer systems with similar characteristics. Some examples include cellular automata, multi-agent systems, collective robotics, and optimization algorithms. The latter example will be reviewed in detail because its formalism will be used again to solve the network routing problem.

2.4.1 Ant Colony Optimization

The Ant System and the Ant Colony System are optimization techniques designed with the swarm intelligence framework in mind. They work by issuing many independent individuals to solve single instances of an optimization problem. These individuals communicate via the manipulation of a local metric in order to ultimately reveal the best solution. This type of algorithm is generally referred to as Ant Colony Optimization (ACO) [10].

Ant System

The Ant System (AS), first introduced in 1992 by Dorigo, Manziezzo, and Colorni, is an optimization algorithm built to solve the Traveling Salesman Problem (TSP) [11]. The TSP is a classic benchmark challenge for optimization algorithms because it is exponentially difficult to solve; TSP is NP-hard. The traveling salesman problem features a network of cities connected by roads and the shortest trip connecting all cities must be found. The problem may feature links with symmetric (TSP) or asymmetric (ATSP) costs. TSP is formalized as a graph $\mathcal{G}$ consisting of vertices, or nodes, and edges ($\mathcal{V}$, $\mathcal{E}$). An edge from node i to node j has cost $C_{i,j}$.

This cost may be thought of most simply as the euclidean distance between two nodes, such as the distance between two cities.

The Ant System works in the following way to solve the TSP. During each iteration $t < t_{max}$, up to m ants make a tour of the graph consisting of $|\mathcal{V}|$ steps. Ideally, this will be a complete tour of the graph with minimum total cost. Each ant decides on the next city to move to based on a random proportional transition rule shown in Equation 2.1.

$$p_{i,j}^{k}(t) = \frac{\tau_{i,j}^{\alpha}(t) \cdot \eta_{i,j}^{\beta}}{\sum_{l \in \mathcal{J}_i^k} \tau_{l,j}^{\alpha}(t) \cdot \eta_{l,j}^{\beta}} \tag{2.1}$$

The probability for ant k to move from node i to node j in the tth iteration of AS is $p_{i,j}^{k}(t)$. The parameter, $\eta_{i,j} = C_{i,j}^{-1}$, is called *visibility* and defines a local search heuristic determining the desirability of moving from node i to j. The set $\mathcal{J}_i^k$ contains all of the nodes that ant k has not yet been to when at node i; it may be thought of as a taboo list. The amount of pheromone on an edge during a particular iteration of AS, $\tau_{i,j}(t)$, is a global search parameter. The values of α and β are used to balance the effects of the local and global heuristics. The rule essentially defines a transition probability distribution on each unvisited neighbor city. This distribution will be different for each visiting ant because each ant will have visited a different set of cities before arriving at the current node i.

After each of the m ants has completed a tour, the pheromone on each link is updated as described in Equation 2.2.

$$\begin{aligned}
\Delta\tau_{i,j}^{k}(t) &= \begin{cases} \frac{Q}{L^k(t)} & : \quad (i,j) \in T^k(t) \\ 0 & : \quad (i,j) \notin T^k(t) \end{cases} \\
\Delta\tau_{i,j}(t) &= \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}(t) \\
\tau_{i,j}(t+1) &= (1-\rho)\tau_{i,j}(t) + \Delta\tau_{i,j}(t) \qquad (2.2)
\end{aligned}$$

Here, $\Delta\tau_{i,j}^{k}(t)$ is the amount of pheromone added by an ant k to the link from i to j. This is dependent on the total cost of the tour of ant k in iteration t, $L^k(t)$, as well if the ant actually used the particular link or not. Better (shorter) paths have more added pheromone. $T^k(t)$ is the set of all links traversed by ant k during iteration t. The parameter Q is a scaling constant. The total amount of pheromone added to any one link is the sum of all pheromones added by each ant. And of course the update equation would not be complete without accounting for pheromone decay, which prevents random fluctuations from reinforcing bad solutions. The parameter $0 \leq \rho < 1$ accounts for pheromone decay. The pheromone decay equation models the biological phenomenon.

The original Ant System proved useful for small test sets, while performance on larger tests dragged.

Ant Colony System

The 1997 Ant Colony System (ACS) by Dorigo and Gambardella was introduced in order to improve on the performance of the Ant System [12]. It has a similar operation to its predecessor, although with some differences in the transition rule, pheromone update rule, and exploration policy.

A parameter $0 \leq q_0 \leq 1$ is added to the transition rule in order to control the random decision making that ants make while on a tour. When an ant must decide the next city to go to, either it chooses to go directly to the best city with a probability of q_0, or else to randomly choose the next city according to a similar bias as seen in AS. This will later be known as probabilistic determinism. The next city j is chosen according to Equation 2.3. The parameter q is a random number uniformly distributed between zero and one.

$$j = \begin{cases} \arg\max_{u \in J_i^k} \left[\tau_{i,u}(t)\eta_{i,u}^{\beta}\right] & : \quad q \le q_0 \\ J & : \quad q > q_0 \end{cases} \tag{2.3}$$

If $q > q_0$, the next hop city is chosen probabilistically according to the distribution given by Equation 2.4,

$$p_{i,j}^{k}(t) = \frac{\tau_{i,j}(t) \cdot \eta_{i,j}^{\beta}}{\sum_{l \in \mathcal{J}_i^k} \tau_{l,j}(t) \cdot \eta_{l,j}^{\beta}} \tag{2.4}$$

which is almost the same as the AS transition rule, save for the lack of the exponent α. This system allows the algorithm to be tuned according to q_0 in order to favor the best known current solution or more exploration. The authors note that this is very similar to the temperature in a simulated annealing optimization strategy.

After each ant has made a tour, pheromone must be added to the links. Unlike AS, pheromone is added only to the edges belonging to the best tour to date according to Equation 2.5.

$$\begin{aligned} \Delta\tau_{i,j}(t) &= \frac{1}{L^{+}} \\ \tau_{i,j}(t+1) &= (1-\rho)\tau_{i,j}(t) + \rho \cdot \Delta\tau_{i,j}(t) \end{aligned} \tag{2.5}$$

Note that this equation is normalized with respect to the AS pheromone update, Equation 2.2. L^{+} is the length of the best tour so far. There is no normalization factor, Q, as there was in AS.

In addition to a global pheromone update, there is also a local pheromone update shown in Equation 2.6. This rule applies to each link visited by an ant not on the best tour.

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \rho\tau_0 \tag{2.6}$$

τ_0 is the initial pheromone on each link. The local pheromone rule causes the pheromone on visited links to return to the nominal level. Less pheromone causes a more uniform distribution across next hop cities, thus encouraging exploration. The ultimate purpose of this rule is to make sure that the same path is not followed consistently between tours, and that different variations are tested.

With regards to the taboo list, J^k, only when it is empty will an ant consider going to a city already visited.

2.4.2 Other Examples

Two other example applications of swarm intelligence are shown here. These are cellular automata and collective robotics. The first is a classic mathematical model which is capable of showing a wide variety of complex behavior despite its simplicity. Collective robotics is currently a hot field in which the lessons learned from social insect models are applied to social robots. These robot societies should be able to self organize themselves and complete cooperative tasks.

Cellular Automata

Cellular automata were originally introduced to study the interaction of simple computing elements. In the context of swarm intelligence, they are often used to simulate groups of simple celled organisms such as bacteria or tissue. A large number of independent locally interacting individuals produce an emergent behavior [13]. It has been shown that for a large variety of rule sets and starting configurations, behaviors ranging from deterministic, periodic, random, to chaotic can be produced. Some cellular automata are Turing machines that are able to compute any Turing-computable quantity [14]. Of course, this property depends on the proper selection of an underlying rule set which governs the operation of each cell.

Cellular automata are generally arranged on a regular grid. Each cell has some independent state and uses the same state machine as all others. The state machine is simply a table which tells each cell which state to move to in the next iteration of the automata based on its own state and the states of the neighboring cells. Each cell is initialized to a certain state and the system is iterated synchronously.

Early computer scientist John von Neumann originally began working with cellular automata in the 1940s in an effort to describe self-replicating machine code. Since then they have been used to study a diverse set of subjects, including chemistry, air/water flow, chaos, traffic modeling, and artificial life, among many others [15].

Collective Robotics

The area of collective robotics has also seen significant progress from the ideas of swarm intelligence. Researchers are either trying to directly reproduce biological behavior or trying to create swarm systems capable of accomplishing tasks useful to humans [1]. This is sometimes known as a biologically inspired testbed. Some of the many research directions include search and explore (to mimic foraging), construction, metmorphics (a group of robots combines to change the functional shape of the whole), or some other form of cooperative task. While the social insect metaphor is very dominant in the field, other examples such as the schooling behavior of fish [16] or the flocking behavior of birds [17] have also been explored.

A prime example of collective robotics is the SwarmBots project [18] in which a number of S-bots cooperate to solve collective tasks. The collection is known as a SwarmBot. The S-bots are wheeled robots with localized sensors and communications, and an extensible manipulator arm. They are designed to solve problems such as cooperative search, retrieval, and coordinated assembly to assist in overcoming environmental obstacles. The SwarmBot project is the most ambitious of its type. There is a large body of work by Japanese researchers in similar areas, primarily in individual robotics.

Robot testbeds have also been used in order to test models of social insect ant behavior in real-life settings, as well as to create coexistent robot-insect societies [19].

2.5 Relationship to Artifical Intelligence

The field of Swarm Intelligence is a subfield of Artificial Intelligence (AI). As seen in the last section, SI has been applied to many problems normally considered to be within the domain of AI. This includes areas such as robot control and optimization. There are a number of references available to highlight the similarities between the two fields; SI is AI.

2.5.1 Reinforcement Learning

Reinforcement learning is an artificial intelligence technique used to train an agent by only offering feedback upon the completion of a task [3]. The agent is required to learn the utility of a number of discrete states by keeping track of the feedback it has received after having visited each. There are many variations of this technique depending on the problem, including when reinforcement is received, how much knowledge of the environment exists, or whether the agent is required to actively manipulate the environment or passively learn from it.

The type of reinforcement learning most interesting to the routing application of swarm intelligence is known as temporal difference learning. In Equation 2.7,

$U(i)$ is the utility of being in state i, α is the learning rate, $R(i)$ is the reward received from being in state i, and $U(j)$ is the utility of moving to neighboring state j. The equation describes how to update the estimate of the utility of state i after moving to state j.

$$U(i) \leftarrow U(i) + \alpha[R(i) + U(j) - U(i)] \tag{2.7}$$

Equation 2.7 is reexpressed in Equation 2.8 in order to better show how utility estimates are averaged. Note that this expression has the same form as the averaging formula of ACS's Equation 2.5.

$$U(i) \leftarrow (1 - \alpha)U(i) + \alpha[R(i) + U(j)] \tag{2.8}$$

In order to speed the convergence of the utility estimates on the true values, the learning rate is reduced over time.

Reinforcement learning, and temporal difference learning specifically, have been applied with great success to applications such as backgammon [20] and robotic control [21]. Backgammon was one of the first "difficult" games to be conquered by computers. The program TD-Gammon used reinforcement learning in order to determine the utility of each state of the game, and how to move depending on the dice roll is receives.

The technique described here assumes that the environment is stationary (in the probabilistic sense). There must be time to apply enough tests to the system in order to have a statistically significant number of samples which will be used to make a good utility estimate. The following chapter will explain in detail how SI can be applied to the routing problem in computer networks, and it will be shown that the routing problem is most certainly *not* stationary. The statistics of the environment are always changing. Because they do not converge, the learning rate cannot be sent to zero. It should be clear from the introduction of reinforcement learning and the utility update equations (Equations 2.7 and 2.8), that reinforcement learning and all of the swarm intelligent routing approaches presented earlier are based on similar principles of statistically testing a state space. The learning rate is the pheromone decay rate.

The SI justification for pheromone decay is to remove old information from the system; to remove old solutions from the space of considered solutions. The mathematical model used to implement this functionality is an exponential decay, which is equivalent to the equation used by reinforcement learning to average many sample tests of the environment. It can be shown that the pheromone update is in fact a linear filter commonly used to average the input. Thus, the two systems really are accomplishing the same thing. A statistical signal analysis can be made of the averaging filter used in this application in order to characterize the statistics of the output of the filter.

2.5.2 Neural Networks

A neural network based solution to the TSP has been proposed by Chen in 1997 which is very similar to ACO [22]. The two approaches were developed independently during the mid 1990s. Since the TSP is a connected graph, similar to the usual representation of a neural network, it is suggested that the utility of each path between cities is represented by a synaptic strength which is changed over time according to successive stochastic trials of tours of the graph and a learning rate. The description of this process is nearly identical to the ACO. It is interesting to note that these two approaches, inspired by different fields, are able to converge on a unified algorithm for solving the same problem.

2.5.3 Gradient Descent

The similarities between ACO and stochastic gradient descent (SGD) have also been explored [23]. Gradient descent is a general optimization technique in which the local error surface of a problem is computed, and the algorithm takes a step in the direction which minimizes the error the most. A stochastic version of this is possible when an explicit expression for the error surface is not available, only a random sample. The stochastic gradient descent algorithm bases its decision either on that available random sample of the surface, or averages over many samplings.

The ACO algorithms described earlier are easily described in this context. Each ant makes a stochastic trip through the problem, such as making a tour around a graph with probabilistic next-hop choices in TSP, and reports its results by laying pheromone. An accurate view of the problem is not possible since there are a finite number of ants that make a tour in each iteration; only a random sample of the solution landscape is available. When all of the pheromone is accounted for and updated at the end of each turn, future ants will be biased by this new gradient which will hopefully lead in the direction of the optimal solution.

The similarities between ant colony optimization and stochastic gradient descent are clear. It is always helpful to be able to cast new algorithms in the light of more established ones. While it may not be fair to claim that ACO is simply a rehash of SGD, principles used to improve and understand the latter can be easily applied to the former.

CHAPTER 3

MOBILE WIRELESS AD-HOC NETWORKS

3.1 Introduction

This chapter will review the state-of-the-art of routing in computer networks. An introduction is presented to the general area in order to establish well-known solutions. These will then be developed in the context of mobile wireless ad-hoc networks (MANETs), and several current solutions will be explained. Finally, swarm intelligent MANET routing algorithms will be reviewed for both wired and wireless networks. This will provide the necessary background in order to understand the place of Termite in the field of network routing.

A network is traditionally formalized as a graph, $\mathcal{G}$, containing a set of vertices, or nodes, $\mathcal{V}$, and edges, or links, $\mathcal{E}$; $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. An edge is directional, starting at a vertex $x \in \mathcal{V}$ and ending at a vertex $y \in \mathcal{V}$. Thus, each edge may be described by an ordered pair of vertices, $e = \{x, y\} \in \mathcal{E}$. For the purposes of this discussion we require that x and y are distinct. There are no reflexive links; $x \neq y$. Each edge is also associated with a cost, or distance (or utility, depending on perspective), $c_{x,y}$; $e = x \overset{c_{x,y}}{\rightarrow} y$. A cost is generally assumed to be positive and finite. In some applications it may be convenient to define a non-existant link as a link with infinite cost. The neighbors of a node x are defined by the set $\mathcal{N}^x$ containing all other nodes for which an edge exists with x as the start point and the other node as the end point; $\mathcal{N}^x = \{y \in \mathcal{V} : \ \{x, y\} \in \mathcal{E}\}$; $x \notin \mathcal{N}^x$; $\mathcal{N}^x \subset \mathcal{V}$. A pair of links between two nodes can be thought of as a single *bidirectional* link if $x \in \mathcal{N}^y$ and $y \in \mathcal{N}^x$. A bidirectional link is *symmetric* if $c_{x,y} = c_{y,x}$ and *aymmetric* if $c_{x,y} \neq c_{y,x}$. This work assumes that there is at most one symmetric bidirectional between two nodes.

A small example of such a network is shown in Figure 3.1. Each numbered node might represent a computer, and each link between nodes represents a communications channel.

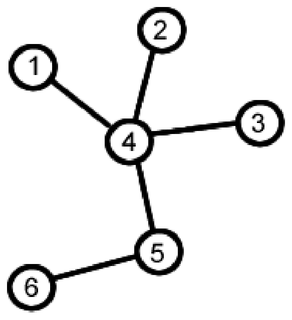

Figure 3.1: Example Network Diagram [24]

3.2 Classic Network Routing

In any computer network, a critical piece of the infrastructure is the routing protocol. It determines which nodes will forward a message from source to destination. There are two basic routing methods known as link state and distance vector. The simplest type of routing is known as flooding and will be discussed first.

3.2.1 Flooding

Flooding is the most primitive routing algorithm. When a node receives a message, it is rebroadcasted to all neighbors. All nodes in the network will receive the message, including the destination. This feature can be especially useful in failure prone networks or in situations where a message must be delivered network-wide. Flooding incurs a substantial overhead due to the large number of transmissions.

The network may not have enough communications resources, generally referred to as bandwidth, to deliver all of the messages. Since most networks are bandwidth constrained, flooding is usually only considered for network-wide announcements or for finding routes to unknown destinations.

A common optimization to control the number of retransmitted packets is to include a unique identifier in each packet. Nodes maintain a list of previously transmitted packets and will drop any that have already been processed.

3.2.2 Distance Vector Routing

Distance vector routing was the first approach used for routing in ARPANET, the primordial computer network. At each node, such algorithms maintain a next hop neighbor for each known destination as well as the cost to arrive at that destination via the neighbor. The distance to each next-hop neighbor is also assumed known. Next-hop distance can be measured in a variety of ways, generally by periodically sending a test short message. Distance and cost are interchangeable in the terminology of computer networks. Early work assumed that the shortest path between two nodes meant simply to minimize the number of intermediate nodes. The cost of the path was the distance measured in hops. Later work used path cost in the more general sense, taking into account factors such as throughput or delay. But the word “distance” has remained.

The Routing Information Protocol (RIP) used on the internet today is a distance vector based protocol [26] [27].

Bellman-Ford Algorithm

The Bellman-Ford algorithm was developed during the late 1950s in order to find optimal routes for logistical concerns (again, the concept of physical distance as cost) [28] [29]. The application to computer networking soon became apparent.

The algorithm may be described simply as follows. Each router in the network periodically advertises the cost of going to each destination in the network on a path through itself. Upon receiving this information, each router updates its own routing table with the minimum of the previously known cost to each destination and the cost of using the neighbor in order to get to the same destination. For example, suppose that x and y are neighbor nodes in the network. Suppose that C_i^x is the minimum cost that x has in its routing table to get to some other node, i. When y receives a routing update from x, it updates its table as shown in Equation 3.1. The current neighbor node through which y is sending packets to i, the current next-hop, is h_i^y. If the path through x is found to be more favorable, then y will store x as the next hop to i.

$$\forall\ i \in \mathcal{V},\ C_i^y \leftarrow \min(c_{y,h_i^y} + C_i^{h_i^y},\ c_{y,x} + C_i^x) \tag{3.1}$$

This algorithm requires a number of rounds of message passing equal to the diameter of the network; information from all members of the network must be received everywhere.

The Counting to Infinity Problem One of the disadvantages of this algorithm is that it is not very fault tolerant. It is susceptible to the well known *counting to infinity* problem. Consider a simple network topology such as, $x \leftrightarrow y \leftrightarrow z$. The problem arises when node x unexpectantly leaves the network, leaving y without a direct link. Node y will hear a routing update from z claiming a two hop distance to x. Not knowing that z's route is in fact through itself, y will update its routing table to reflect z's supposed route to x. In the next round, z will update its routing table to reflect y's change, since y is z's next-hop to x. Route distances will count to infinity because a loop has been established that cannot be broken.

There exists a standard solution to the counting to infinity problem known as *split horizon with poisoned reverse* [26]. This approach simply avoids sending routing updates about destinations that a node learned about from a neighbor, to that neighbor. The refers to the "split horizon." If a route update *is* sent to that neighbor, it may be sent with a distance of infinity. This refers to the "poisoned reverse."

3.2.3 Link State Routing

Link state routing is an alternative to distance vector routing. Instead of incrementally computing routes and waiting for intermediate updates until all routing information has been updated, the link state approach opts to broadcast the state of each router's local neighborhood to all other routers. All routers are then able to compute routes based on complete knowledge of the network.

When a routing update is due, which can be determined either by a specific period or by the occurrence of a significant event, each router floods its local connectivity information into the network. Connectivity information includes a list of neighbors and the cost to arrive at each. Each route update has a unique source dependant sequence number. This is used to control the flood, but also determines the relative age of a route update. A routing table should only be updated with information from update packets that have a sequence number greater than what is currently recorded. Only the newest route information is incorporated into the network. The requirement also highlights a problem with link state routing. If each router is not using the same information to compute routes, suboptimal routing, loops, or network disconnection may occur. Routers may not be aware of the existence or disappearance of links, or simply use incorrect link costs.

The Open Shortest Path First (OSPF) routing protocol is the current routing protocol used on the internet [30]. OSPF is a link state routing protocol.

Dijkstra's Algorithm

Dijkstra's algorithm is used for calculating the minimum cost path from one node in a network to all others [31]. This is known as the minimum spanning tree (MST). It requires complete knowledge of the network topology and all link costs. This is in contrast to the Bellman-Ford algorithm which can be run in a distributed fashion.

The algorithm starts by considering the source node of interest on the network graph, x. Consider the cost to each of x's neighbors. Add the neighbor with minimum cost, y, to the minimum spanning tree and label it with the total cost to x. This procedure is then continued. The neighbors of x and y are considered, and that neighbor, z, with the minimum cost to x is added to the MST. Suppose z is a neighbor of y, then since y is on the shortest path to x, and z is the closest neighbor to y, then the path $x \leftrightarrow y \leftrightarrow z$ must be the shortest path between x and z. The algorithm continues until all nodes in the network have been added to the minimum spanning tree.

3.2.4 Discussion

Flooding, distance vector, and link state routing protocols define the basis of all routing in networks. Other methods do exist, but those reviewed here are by far the most studied and used. The next section will introduce a specific type of network known as a mobile wireless ad-hoc network in which nodes are mobile and communicate over wireless links. The topologies of such networks can change quickly and thus require a large number of routing updates. The basic routing strategies presented in this section will be adapted to these needs.

3.3 Introduction to Ad-Hoc Networking

Computer networks are more relevant than ever in today's Information Age [64]. Since the first computer network became operational in 1969 as ARPANET, the number of computers connected to this network of networks has grown exponentially from a mere handful to billions of machines. In nearly the same amount of time, technology has progressed to allow wireless connectivity to the network at large. This approach culminated with the proliferation of one-hop wireless networks such as cellular telephone systems or wireless LANs. Over the last ten years, such wireless networks evolved to accommodate multi-hop wireless networks. Unlike earlier systems in which radio links merely replaced wires for long-haul routes or in difficult to develop terrain, modern multi-hop, or ad-hoc, networks assume node mobility and mutable topology.

A mobile wireless ad-hoc network (MANET) is a collection of mobile computers, or nodes, using only wireless communications links in order to send messages between themselves. Communications range is limited and thus each node has only a few neighbors. Node mobility causes dynamics on the topology of the network as nodes move in and out of communications range. Information may be sent between one node in the network to another by having intermediate nodes forward the message, exactly as in a typical wired network.

Ad-hoc networks represent a step forward in mobility, robustness, and availability of computer communications. Users are no longer restricted to a certain area in order to maintain a data link. This is a common issue with older wireless technologies such as cellular networks. MANET users are allowed to go where they please as long as they remain within range of another participating node. Ad-hoc networks are created and configured in real time, without the need for installed infrastructure or system configuration. This feature allows for a great deal of flexibility, and the same amount of effort is needed to create a network of any size.

3.3.1 Applications

There are a number of sample applications used to motivate the development of MANETs. Although there has not yet been any wide scale deployment of this technology to date, additional opportunities are sure to rise. In general, MANETs are used to extend computer communications range in areas where the deployment of permanent infrastructure is difficult or too costly.

Search and Rescue

A search and rescue scenario generally exists in regions that have little or no installed communications infrastructure. This may be because all of the equipment was destroyed, or perhaps because the region is too remote. Rescuers must be able to communicate in order to make the best use of their energy and to maintain safety. By automatically establishing a data network with the communications equipment that the rescuers are already carrying, their job is made easier.

Battlefield Communications

The military is interested in MANET development in order to augment the communications capabilities of their forces. Battlefields are generally located in areas with little installed or trusted communications infrastructure. Ad-hoc networks may be established quickly between any number of cooperating forces. Communications may be used to disseminate local information, or to send data to commanders.

Infrastructureless Networking

A civilian application for MANETs includes ubiquitous computing. By allowing computers to forward data for others, data networks may be extended far beyond the usual reach of installed infrastructure. Networks can be more widely available, easier to use, and cost less to deploy.

Sensor Networks

Sensor networks are composed of a very large number of small sensors which can be used to detect any number of properties of an area. Examples include temperature, pressure, toxins, pollutants, etc. The capabilities of each sensor are very limited, and each must rely on others in order to forward data to a central computer. Individual sensors are limited in their computing capability and are prone to failure and loss.

3.3.2 Challenges

There exist a number of challenges facing the development and deployment of ad-hoc networks [32]. Some examples include routing, scaling, energy efficiency and conservation, MAC layer transmission strategies, and quality of service (QoS). While each of these is critical to the performance of the network, this section will focus on routing.

The primary challenge to face routing in MANETs is the dynamic network topology. Some factors which influence the topology include mobility, node loss, and variable link quality. Mobility is the primary cause for topological dynamics in an ad-hoc network. As nodes move amongst each other, their limited communications range allows them to only send messages to their neighbors. These are lost and gained as nodes move. New paths must be discovered and maintained as the topology changes. The effort to do this can become prohibitive if the topology changes too quickly. Increasing the mobility of the nodes is known as scaling with respect to speed.

Node loss also has an effect on topology. If a node is removed from the network it will no longer be able to forward data. This may occur because the node is destroyed or because the machine is turned off. If too many nodes are lost there will not be enough remaining participants to maintain an operational network. There may be too few nodes between source and destination to send a message.

Because a wireless channel is used to communicate between nodes, each network link has similar properties. Properties include range, data rate, and bit error rate (BER). Link range determines a node's neighbors, while the latter properties affect the flow of traffic across those links. All of these attributes may also change over time, leaving it up to the routing algorithm to adapt to the new environment.

3.3.3 Routing Metrics

A variety of metrics may be used to optimize network performance. Some examples include throughput, delay, path length, energy consumption, and link lifetime. These are quantities that can be measured independently and locally at each node; it is not necessary to use global information.

The former three are general metrics that can be applied to any type of network. The latter two are more often used in MANETs due to the network's special characteristics. Throughput is the number of data bytes that can be delivered per second between any two nodes in the network; it is to be maximized. Delay is the number of seconds it takes for a message to arrive from a source to a destination; it should be minimized. Path length is a first order approximation to delay. If the number of transmissions, or hops, between source and destination is minimized, then presumably the amount of time that the packet will spend in transit will also be minimized.

Nodes in an ad-hoc network are often assumed to be energy limited; energy consumption has become an important metric. Paths requiring the minimum of energy to transmit packets should be found such that the network stays alive for as long as possible. The longevity of a network is defined as either the time until the first node dies, or the time until the network becomes disconnected due to node death. The quality of individual links and that of entire routes has also come under consideration. An example is link lifetime. Links which have been established for a long time may break soon, or conversely may be considered more reliable in the future. This depends on the mobility model used. The dynamics of the link attributes is also a factor. If its properties are changing quickly, it may be considered unreliable and avoided.

Protocols must be designed to take these parameters into account. The interactions between these parameters must be understood; the appropriate tradeoffs must be made when designing a system with these metrics in mind.

3.3.4 Performance Metrics

A variety of performance metrics exist to measure the performance of MANET. Some metrics may depend on a particular application, but the key measures are data goodput and control overhead.

Data goodput is the fraction of successfully delivered data packets and is a measure of the reliability of the network. This metric ranges between zero and one, where values close to unity are desired.

Control overhead is the fraction of non-data packets to the total number of packets transmitted in the system. A network should expend the majority of its resources to deliver data packets; if these resources are used to move control, or network management, packets instead, fewer data packets can be delivered. Control overhead is a measure of amount of effort required by the network to remain aware of its own state.

3.3.5 Medium Access

There is a specific terminology associated with the wireless communications environment and its capabilities. Oftentimes communications capabilities will be dictated by the specific medium access technology being used. Radio is usually used in MANETs, and transmissions are assumed to be omnidirectional.

The most common medium access (MAC) layer used in MANETs is the standard IEEE 802.11 [33], which is a Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA) protocol. This widespread technology is primarily used for one-hop wireless networking and is also known as "wireless LAN" or WiFi. Usually communications are between an access point (AP) which acts as a bridge between a wired network and the wireless medium. Participating nodes are within one hop of the AP. The standard specifies two basic transmission modes, broadcast and unicast. If a packet is broadcast, the transmitter sends the packet according to the rules of CSMA. There is no destination for a broadcast packet and the transmitter does not wait for an acknowledgement of reception. Each receiver can only receive one packet at a time, so if another terminal transmits in range of a receiving node then a collision may occur; neither packet will be received. A unicast packet is transmitted somewhat more reliably to a specific destination. The transmitter first broadcasts a Ready-To-Send (RTS) packet to the intended receiver. If the receiver successfully receives it and senses a free medium, it will reply with a Clear-To-Send (CTS) message. Upon receiving the CTS, the transmitter sends the data packet, to which the receiver replies with an acknowledgement (ACK) packet. If this RTS/CTS/DATA/ACK exchange is not successful, it may be due to a collision in the medium, a previously reserved medium, or because the receiver is not within communications range of the transmitter. An unsuccessful unicast exchange is generally used to indicate a lost neighbor to the routing layer.

3.4 Traditional Ad-Hoc Routing Solutions

Traditional ad-hoc routing solutions are naturally modeled on solutions to routing in wired networks. After all, it is only because the classic solutions to wired routing were found not to work in the ad-hoc domain does the field of ad-hoc routing exist at all. Well known solutions such as OSPF are found to generate so much control traffic as to be unusable [34].

Ad-hoc routing algorithms are necessarily much more efficient in their use of control traffic and network resources. They can be generalized into two categories, proactive, or table-driven, and reactive, or on-demand. Proactive protocols maintain a route to every node in the network at any given time. When a route is needed, it is immediately available. The proactive approach minimizes end-to-end packet delay because a packet can be immediately transmitted without needing to find a route first. A great deal of control traffic is also generated as the network continually updates its routes, even if no data traffic is available. Reactive protocols only find routes to a destination when there is data available for it. They often generate much less control traffic than a proactive protocol, but require that a route to a destination is discovered before packets can be sent. Reactive protocols incur greater end-to-end delay.

The following sections will provide some examples of currently available proactive and reactive protocols for ad-hoc networks.

3.4.1 Proactive Routing

Proactive protocols were the first to be developed for ad-hoc networks. Well known routing protocols designed for wired networks were adapted to the mobile setting. The design methodology carried over as well, as it was felt that routes should be on hand to any destination at any given time since traffic would always be present. Traditional approaches such as the link-state or distance vector algorithms flood their routing information to all routers in order to maintain a complete and current snapshot of the network. It was in this spirit that the first proactive routing protocols were designed for use in ad-hoc networks.

Dynamic destination-Sequenced Distance Vector

The Dynamic destination-Sequenced Distance Vector (DSDV) routing protocol proposed by Perkins and Bhagwat in 1994 is one of the first to solve the routing problem in ad-hoc networks [35]. DSDV is modeled as a distance-vector protocol in which each node periodically broadcasts its routing table to its neighbors. Each neighbor then updates its own routing table with the new information, and if any route has changed, that new information is subsequently broadcast. This procedure allows route information to propagate throughout the network. Sequence numbers are used for each destination route in order to determine its freshness. Routes are not updated if the local route to a destination has a higher sequence number than the advertised route.

DSDV does not make explicit a route discovery or route repair mechanism. It is assumed that the route advertising procedure will maintain valid paths to all nodes in the network. It is noted that route updates are broadcast when a link break or a change in topology are noted. In order to prevent a flood of possibly large route updates, each broadcast is delayed in order to anticipate any more updates which may further update the table. This is a simple information aggregation scheme.

Optimized Link-State Routing

The Optimized Link-State Routing algorithm was introduced in 2001 by Jaquet, Mühlethalter, Clausen, Laouiti, Qayyum, and Viennot as a proactive routing protocol for mobile wireless ad-hoc networks [36] [37] [38]. It optimizes the traditional link-state protocol by reducing the amount of information in a route update, as well as introducing an optimized flooding procedure. Both of these are based on the use of multipoint relays (MPRs), which are a subset of the neighbors of each node. They are responsible for forwarding and compiling routing messages. The timeliness and ordering of route information is guaranteed by sequence numbers.

OLSR works by having each node first designate a subset of its neighbors as multipoint relays. MPRs are chosen such that if each MPR of a node n rebroadcasts a message, then all two-hop neighbors of n will receive the message. Message flooding is optimized in this way. In order to establish routes, each node periodically floods (via the use of MPRs) a list of its MPRs, similarly to other link-state routing algorithms. Each node then has a list of everyone in the network and their preferred neighbors. A complete path to any destination can be calculated by chaining together node/neighbor pairs.

Multipoint relays are determined when a node broadcasts hello messages to establish and verify a radio link with all of its neighbors. A received hello message is always replied to. Thus, it is possible to generate a list of neighbors to whom a bidirectional communications link exists. Each hello message also contains a list of all neighbors of a node. In this way, information about the local two-hop topology is collected. This information can be used for each node n to specify its set of MPRs; $\text{MPR}(n) \subseteq \mathcal{N}^n$. This MPR list is included in subsequent hello messages so that each node knows for which neighbors it should forward messages.

3.4.2 Reactive Routing

It was not long until even proactive solutions were found not to perform terribly well in the ad-hoc environment [39]. Networks using proactive protocols could not tolerate constant routing updates. In light of these failings, on demand, or reactive, protocols discover routes to destinations only when they are needed. The advantage of this approach is that control traffic is only generated when necessary, releasing system resources for moving user data. The disadvantage is that there is some delay associated with the route discovery procedure.

Anatomy of a Reactive MANET Routing Protocol

All reactive routing protocols have a general structure, including a mechanism for routing to a known destination, for the discovery of a route to an unknown destination, for handling an error in a route, and finally also some optimizations. Routing to a known destination is straightforward because route information is already available. When a packet arrives for an unknown destination, a route discovery procedure is initiated. Since only reactive protocols are considered here, this situation may often be the case. Usually the network will be flooded with packets searching for the destination. These route request packets will collect and establish route information. Once a route has been found, it is used until it breaks again. A route error procedure is then initiated in order to update the relevant nodes of the change in network structure. Some optimizations are always implemented depending on the specifics of the previous mechanisms. Classic optimizations include locally repairing broken routes instead of rediscovering the route, or cleverly directing the route discovery process.

Ad-hoc On-demand Distance Vector

Originally based on the proactive DSDV, the Ad-hoc On-demand Distance Vector (AODV) routing protocol is currently one of the most popular on demand routing protocols for ad-hoc networks [40]. It is in the continuing stages of being standardized by the Internet Engineering Task Force (IETF) MANET working group [41] [42]. AODV was originally introduced in 1999 by Perkins and Royer, continuing in the footsteps of DSDV with the use sequence numbers to guarantee route freshness. It has seen extensive testing and development since its introduction, and is used as a model for many new proposals.

When a data packet arrives at a node with no route to the destination, a route request procedure is initiated. There are two variations of this. If a source node has no route to the destination, an expanding ring search is started. A local flood of route request (RREQ) packets is broadcast with a limited time-to-live. If no route reply (RREP) packet is received within a timeout period, another local flood is issued with a larger time-to-live, allowing it to search a larger area. This process is retried a certain number of times until the destination is found or is declared unreachable. Sequence numbers are embedded in the RREQs in order to distribute current information about the source, and to specify a minimum freshness for the information about the destination. Route metrics are also included in RREQ packets in order to create reverse routes to the source. If an intermediate node must search for a destination, this is known as a local repair and is comprised of only a local RREQ flood. If the destination cannot be found, a route error (RERR) is returned to the source to inform it that a new route must be discovered.

Route replies (RREP) are unicast from an intermediate node with an active route to the requested destination or from the destination itself. An active route is guaranteed back to the source since reverse routes are created by the RREQs. An active route to the destination should include a sequence number at least as large as that in the route request. Once a RREP is received, any packets that had been buffered for the destination can be sent to the destination.

A route error (RRER) message is needed to inform the source that an irreparable break has occurred in the route to the destination; a new route should be found.

The route error propagates from the node immediately upstream of the break back to the source. The route to the destination is deactivated and each intermediate node will require an equally fresh new route to the destination.

Dynamic Source Routing

The 1996 Dynamic Source Routing (DSR) protocol by Johnson and Maltz shares the popularity of AODV [43]. There is also a continuing IETF standardization effort [44]. It is a contemporary of AODV and approaches the problem from a source routing perspective rather than a distance vector basis.

DSR stores the entire route to the destination in the routing table rather than just the next hop as in AODV. Data packets contain their entire route to the destination in their header. When a data packet is received at a node and a route for the destination does not exist in the routing table, a route request is initiated. As in AODV, the RREQ packets are flooded throughout the system. RREQs record each visited node in their header in order to build a route to the destination. Overheard RREQs may also be examined for new routes. A route reply is generated by the destination itself or an intermediate node possessing a valid route to the destination. The RREP is returned to the source using the reverse route found in the RREQ header. Packets buffered at the source are released once the RREP arrives and a destination route is made available.

If a node receives a packet and the next hop is found to be broken, an alternative route may be chosen for the packet if it is available. This is known as salvaging the packet. Alternative routes are discovered by overhearing RREQ broadcasts. The packet is dropped if no route is available. In either case, a route error message is returned to the source noting that the RRER originator no longer has a link to the lost neighbor. All intermediate nodes should update their routing tables accordingly.

3.5 Swarm Intelligent Routing

The traditional routing solutions presented in the previous section can be generally characterized as deterministic algorithms on graphs. Consider the mathematics of the distance vector Bellman-Ford algorithm presented earlier in the chapter. The algorithm will always find the same routes in the same network in the same way. In contrast, the swarm intelligent (SI) routing algorithms presented in this section are random algorithms on graphs. Such algorithms may not have the same properties as their deterministic relatives, however they can oftentimes approximate the same behavior with a variety of interesting savings elsewhere. SI routing does away with firm rules and instead allows the system to self-organize on its own accord.

Swarm intelligent routing protocols for wired networks are first presented, with ad-hoc protocols following.

3.5.1 Wired Networks

A number of proposals exist which apply the principles of swarm intelligence to routing in wired networks. These include Ant-Based Control (ABC), AntNet, Cooperative Asymmetric Forwarding (CAF), Virtual-Wavelength Path Routing (VWPR), Multiple Ant Colony Optimization (MACO) [51], Mobile Ants-Based Routing (MABR) [52], among others. ABC is the oldest with its introduction in 1996. AntNet is to be credited as the most well known due to its solid performance and design by the inventors of the related and extremely successful Ant Colony Optimization (ACO) algorithm. CAF optimizes some approaches used by

AntNet and also extends a distance vector methodology. VWPR is included for its introduction of the pheromone repel feature in routing, a vastly underappreciated technique in this field borrowed from ACO. The Multiple Ant Colony Optimization algorithm is a more straightforward adaptation of ACO to routing. It uses the idea of having multiple ant colonies discover paths, each being repelled by the pheromone of the other. This ensures that a larger search space will be covered and thus better routes should be found faster. MABR is based on AntNet and describe a hierarchical location-based routing scheme. The network topology is broken into a regular grid by an abstraction layer, and pheromone based routing is executed on the grid.

Ant-Based Control

Ant-Based Control (ABC) was introduced by Schoenderwood, Holland, Bruten, and Rothkrantz in 1996 and is considered to be the first biologically inspired routing algorithm [45]. It is a routing algorithm for circuit-switched (eg. telephone) networks which routes calls based on the local interaction of mobile agents. Mobile agents (ant packets) traverse the network, updating routing tables at each node depending on the observed path quality. Routing tables consist of next hop probabilities for each destination. Ants traveling in one direction influence the placement of calls in the opposite direction. Symmetric bidirectional links are assumed.

ABC follows the ant food foraging analogy very closely. Ant packets are routing, or control, packets. Ants update the local pheromone table at each node they visit according to Equation 3.2. Pheromone tables are routing tables, and contain the probability of choosing a particular next hop to arrive at a certain destination. Probability values are manipulated directly as shown; pheromone is not used as an intermediate value as it is in biology. The value $p^n_{i,d}$ is the probability at node n for using the link to neighbor i to get to destination d. The source of a packet is s, and the previous hop of the ant is r.

$$\begin{aligned} \forall i \in \mathcal{N}^n - r,\ p^n_{i,s} &\leftarrow \frac{p^n_{i,s}}{1+\varphi} \\ p^n_{r,s} &\leftarrow \frac{p^n_{r,s}+\varphi}{1+\varphi} \end{aligned} \tag{3.2}$$

The value φ is known as the *learning rate.* The fact that the used route is increased and the others decreased in probability is an example of positive and negative reinforcement in ABC.

ABC also describes a method for adjusting the learning rate based on the path cost. Equation 3.3 shows how ABC accomplishes this in general [46].

$$\varphi = \frac{k}{f(cost)} \tag{3.3}$$

Here, f is a non-decreasing function of the path cost, and $k > 0$ is a constant. Since ABC optimizes for low delay routes, it is clear that routing probabilities are adjusted less as the ant spends more time in the network. The age of a packet includes not only the actual amount of time spent in the network, but also the expected delay that the ant would experience if it were a call waiting to be connected. The age of a packet includes network congestion.

Ant packets are launched on regular intervals by each node in the network to a random destination. They are routed randomly according to the probabilities present in the pheromone table for their particular destination. In order to encourage ants to try new paths, a noise factor is added to the routing decision. This is necessary in case the network conditions should change or better routes become available. If a high probability route exists, there is little incentive to stop using it, even if its quality drops. The probability update equation only allows routing probabilities to be positively reinforced when a link is used. Probabilities cannot decrease, no matter how bad the achieved path might be. The noise factor, on the order of 5% in ABC, routes ants uniformly over all outgoing links rather than according to the probabilities in the pheromone table.

Calls are routed such that they only follow the link with the highest probability at each node.

AntNet

AntNet is an approach from 1997 by Dorigo and Di Caro to using the social insect analogy to solve the routing problem in wired packet switched networks [47]. Unlike ABC, AntNet assumes asymmetric bidirectional links. Routes from destination to source, also known as reverse routes, cannot be influenced by an ant moving from source to destination as was the case in ABC. An ant moving in the forward direction will experience different costs than an ant moving in the reverse direction.

The algorithm works as follows. At regular intervals, each node in the network sends a *forward ant* packet to a randomly chosen destination. Forward ants are routed probabilistically as in ABC. Each node contains a routing (pheromone) table indicating the utility of a particular link to arrive at a destination. This utility is described by a probability. In order to ensure that the entire network is explored consistently, a forward ant chooses a next hop uniformly according to an exploration probability. This feature serves the same purpose as that of noise in ABC. As the forward ant is routed to its destination, it maintains certain information on a stack, S which includes a list of all visited nodes as well as corresponding metric information. The original AntNet description optimizes for low delay paths and so path costs are equal to the time necessary to travel between nodes. Forward ants are routed with the same priority as data packets and experience the same traffic conditions as the rest of the network that they are measuring. If the forward ant travels in a loop (which can be detected based on the contents of its stack), those nodes are popped and the packet continues. Upon arrival at the destination, the stack is saved, the forward ant is killed, and a *backward ant* is created to return to the source.

The purpose of the backward ant is to return the network information to the forward node's path. This is done simply by reversing the path on the stack. In order to update the network as quickly as possible, backward ants are routed with priority over data packets. They follow the forward ant's reverse path deterministically. The backward ant updates the local routing table at each node with information about the trip time from the source to the current node.

For example, if the forward ant has travelled on the path $s \to w \to x \to \ldots \to y \to z \to d$, then it's cost stack is, $S_{cost} = \{c_{s,w},\ c_{w,x}, \ldots,\ c_{y,z},\ c_{z,d}\}$, and it's visited stack is, $S_{visited} = \{s,\ w,\ x, \ldots,\ y,\ z,\ d\}$. Since links are asymmetric, eg. $c_{s,w} \neq c_{w,s}$, the backward ant must return this information to the nodes that can use it. A node can use the stack to determine the cost of a path to all of the intermediate nodes between itself and the destination by adding intermediate costs.

The AntNet algorithm can be considered as a sort of machine learning algorithm meant to learn the best routes in a dynamic environment. It is best characterized as a type of reinforcement learning in which the backward ant represents a feedback signal regarding the quality of the current routing solutions. Each path measurement is relative; there is no absolute measure of the goodness of a particular link or route. Routing probabilities are updated similarly to those in ABC, however the reinforcement parameter, φ, is calculated according to a much more complex statistical method based on achieved trip time. The mean and variance of trip times are compared in order to determine the quality and reliability of the current data. Note that φ is different for each next-hop/destination pair, since it is based on the measured cost of that path. The method used to update routing probabilities in AntNet is shown in Equation 3.4.

$$\begin{aligned} \forall i \in \mathcal{N}^n - r,\ \forall d \in S_{visited},\ p^n_{i,d} &\leftarrow p^n_{i,d} - \varphi^n_{i,d} \cdot p^n_{i,d} \\ \forall d \in S_{visited},\ p^n_{r,d} &\leftarrow p^n_{r,d} + \varphi^n_{r,d} \cdot (1 - p^n_{r,d}) \end{aligned} \tag{3.4}$$

The equations are rewritten in Equation 3.5 in order to make clear that this update is simply an averaging mechanism.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ \forall d \in S_{visited},\ p^n_{i,d} &\leftarrow (1 - \varphi^n_{i,d}) \cdot p^n_{i,d} \\ \forall d \in S_{visited},\ p^n_{r,d} &\leftarrow p^n_{r,d} + \varphi^n_{r,d} \end{aligned} \tag{3.5}$$

Note that they also have the same form of Equations 2.5 and 2.6 with $\tau_0 = 0$.

AntNet 1.1 The original AntNet network routing algorithm is updated by Barán and Sosa in 2000 with AntNet 1.1 [48]. The improvements are reviewed briefly below.

- AntNet assumes uniform link probabilities upon initialization. AntNet 1.1 proposes to positively bias the probabilities of using neighbor links to get to those neighbors; $\forall i \in \mathcal{N}^n,\ p^n_{i,i} > \frac{1}{|\mathcal{N}^n|}$.

- AntNet does not describe the case of link failure. It is assumed that when a link fails, the probability of using that link is then distributed uniformly among the remaining links. AntNet 1.1 proposes a non-uniform update of remaining link probability after link failure.

- When a previously lost link returns, AntNet treats it as a brand new link with no previous history. AntNet 1.1 suggests that the probability of this returning link should be initialized based on the probability that it had previously and a generic uniform probability. Thus if a link is dropped at time t_d and returns at time t_r and had uniform probability at time $t = 0$, the probability of the dropped link to i to get to destination d, $p^n_{i,d}$, is shown in Equation 3.6.

  $$p^n_{h,d}(t_r) = (1 - \rho) \cdot p^n_{h,d}(0) + \rho \cdot p^n_{h,d}(t_d) \tag{3.6}$$

  This reinitialization method assumes that the returning link retains characteristics similar to those it had before it dropped. The *coefficient of memory* is ρ.

- The use of probabilistic determinism is also proposed. Instead of forward ants choosing next-hops according to the probability distribution listed in the routing table, they choose the best link deterministically with some probability. The ant chooses randomly whether to decide on the next hop probabilistically (as described previously), or deterministically and only go with the best current solution. This technique is also used by ACO and generally helps direct packets toward their destination.

- AntNet 1.1 suggests to set the number of ants sent into the network to the number of links in the network. This approach should allow the for the entire network to be updated on average every time forward ants are issued. A tradeoff is made between control overhead and timely network updates.

- When a backward ant cannot continue its trip to the forward ant source due to a broken link, it is destroyed. Because the network topology has changed, the information that it is carrying is now worthless. AntNet does not consider this situation.

Cooperative Asymmetric Forwarding

Cooperative Asymmetric Forwarding (CAF) Routing, introduced by Heusse, Snyers, Guérin, and Kuntz in 1998, is a parallel to AntNet [49]. It solves the routing problem in networks with bidirectional asymmetric links. CAF builds on AntNet's forward/backward ant system by collecting forward information from existing data traffic and then uses a reverse ant to deliver routing information back to the source along the reverse route. CAF does not make use of the biological analogy that motivates AntNet. A unified distance vector perspective is presented where link probability is proportional to path utility.

The scheme works as follows. Since links are asymmetric, traffic moving in one direction cannot update routing tables meant to direct traffic heading in the opposite. As data traffic moves from source to destination, each node n estimates the cost of moving from its neighbor p to itself, $c^n_{p,n}$ based on the measured cost, $c_{p,n}$. This data is essentially collected for free from existing traffic. Each node periodically sends a routing agent to a randomly selected destination. This agent must move backwards along the flow of arriving data from that destination. Routing data is only available (and relevant) along the path from the destination to source. Each node maintains a reverse routing table, V, for the purpose of sending a routing agent from destination (itself) to source. This reverse routing information is collected by observing the rate of packet arrival and updating the routing table accordingly. This is shown in Equation 3.7. The previous hop of the arriving data is r and the source is s.

$$V^n_{r,s} \leftarrow (1-\varphi)\cdot V^n_{r,s} + 1 \tag{3.7}$$

The routing agent is sent to a randomly selected next hop, i, in order to arrive at the traffic source, s, based on the distribution calculated by Equation 3.8.

$$p^n_{i,s} = \frac{\left(V^n_{i,s}\right)^F}{\sum_{j\in\mathcal{N}^n}\left(V^n_{j,s}\right)^F} \tag{3.8}$$

The exponent F is the *sensitivity* and will be seen again later.

The routing agent follows a random path based on forward traffic flow to the source. It collects forward route statistics saved at each node (based on measurements from packets moving from agent destination to the agent source) in its stack. The agent updates the forward routing table at each visited node, as in AntNet. The agent is killed once it arrives at its destination. The CAF routing update method is somewhat more efficient than that used by AntNet because it takes advantage of existing traffic to measure path performance. However, it requires that the destination of a traffic flow send routing agents back to the source in order to update the routing tables.

Suppose that a routing agent is sent from a flow destination to its source following the path $d \to w \to x \to \ \ldots\ \to y \to z \to s$. Upon arrival at the flow source, the stack of the route agent will be $S_{cost} = \{c^d_{w,d},\ c^w_{x,w},\ \ldots\ ,\ c^y_{z,y},\ c^z_{s,z}\}$. Each node can update its routing table with the estimated cost to all intermediate nodes.

Actual routing updates are made similarly to previous schemes. When an agent arrives at node n from previous hop neighbor r, the routing table is updated as shown in Equation 3.9.

$$\forall d \in S_{visited},\ C^n_{r,d} \leftarrow (1-\varphi) \cdot C^n_{r,d} + \varphi \cdot c^S_{r,d} \tag{3.9}$$

The cost to use node r to get to node d according to the route agent's stack S, $c^S_{r,d}$, is the sum of all costs between node n and d on the stack. Since the route agent is arriving from the previous hop r, that node is also on the stack.

Data moving in the forward direction are routed randomly according to the distribution calculated by Equation 3.10.

$$p^n_{i,d} = \frac{\left(C^n_{i,d}\right)^{-F}}{\sum_{j \in \mathcal{N}^n} \left(C^n_{j,d}\right)^{-F}} \tag{3.10}$$

This cost normalization approach to determining route probabilities is common. As in Equation 3.8, F is used to bias routes according to their relative differences. The inverse of each cost is used to calculate routing probability since packets should be forwarded over minimum cost paths.

Virtual-Wavelength Path Routing

Virtual-Wavelength Path Routing was introduced by Varela and Sinclair in 1999 [50]. Ant Colony Optimization is applied to the problem of routing and wavelength allocation in multi-wavelength all-optical virtual-wavelength path routed transport networks. A typical application of this system is in long haul fiber optical networks requiring not only that packets be routed, but also that the minimum number of wavelengths be used to transport those packets. Since each fiber link can only hold a limited number of wavelengths, a constraint satisfaction element is added to the traditional routing problem.

The authors propose a number of approaches using ACO, however the most significant development is the idea of pheromone repel. In order to ensure that the algorithm does not try to place too many wavelengths on one fiber, or alternatively that it tries to minimize the number of wavelengths used in the network, different ants are sent into the network that are repelled by the pheromone of others. Pheromone may be thought of as a routing metric.

This is implemented below. The probability of each link is first determined based on the the local ant's resident pheromone. This is a simple normalization of the link pheromone.

$$p^n_{i,d} = \frac{P^n_{i,d}}{\sum_{j \in \mathcal{N}^n} P^n_{j,d}} \tag{3.11}$$

All of the pheromone belonging to other ants is summed to $\ddot{P}^n_{i,d}$ and a link probability mass distribution is calculated. A meta forwarding probability is then computed based on these two distributions such that,

$$\hat{p}^n_{i,d} = \frac{p^n_{i,d} \cdot \left(\ddot{p}^n_{i,d}\right)^{-R}}{\sum_{j \in \mathcal{N}^n} p^n_{j,d} \cdot \left(\ddot{p}^n_{j,d}\right)^{-R}} \tag{3.12}$$

In this way the link probability is adjusted away from large competing pheromone concentrations. The *pheromone repel constant*, $R \geq 0$, determines the strength of the pheromone aversion.

This idea will be used later in order to direct the movement of packets through the network.

3.5.2 Wireless Routing

The routing problem in mobile wireless ad-hoc networks has received some attention from the swarm intelligence community. The properties of SI algorithms are very well suited to the problem, which requires optimization in a dynamic environment. The Probabilistic Emergent Routing Algorithm (PERA) is the first routing algorithm to be proposed in 2002. This was quickly followed by the Ant-like Routing Algorithm (ARA) and Termite in 2003, with Ad-Hoc Networking with Swarm Intelligence (ANSI) in 2004. Many of these algorithms extend AntNet or ABC to the ad-hoc problem; the forward/backward ant architecture is used for route discovery and then various route repair and error schemes are proposed.

Probabilistic Emergent Routing Algorithm

The Probabilistic Emergent Routing Algorithm (PERA) designed by Baras and Mehta in 2002 is the first swarm intelligent MANET routing algorithm [53]. It operates very much in the way of AntNet, using forward and backward ants to collect and distribute information about the network. It also borrows elements from more traditional contemporary routing algorithms such as AODV and DSR. The reinforcement parameter is computed based on metric mean and variance.

The primary features which set PERA apart from other implementations is its handling of forward and backward ants to manage the network. Forward ants are sent from each node on regular intervals to random destinations, or in the event that a route discovery must be made. They operate as in AntNet, collecting metric information as they are routed randomly to the destination. However, PERA broadcasts all forward ants, effectively flooding the network to find the destination. This is a similar scheme as proposed by DSR. Each packet has a sequence number which is unique to the source node and so each forward packet can be uniquely identified. All nodes maintain a list of maximum sequence numbers last seen from every other node in the network. If a packet is received with a sequence number less than the last seen, it is discarded. This prevents forward ant floods from growing out of control, as well as preventing old information about the network from being used in route updates. When a forward ant finally reaches its destination, a backward ant is generated with the information carried by the forward ant. It should be noted that multiple forward ants may reach the destination if there are multiple paths between it and the source. This will generate multiple backward ants along each path to the source. The backward ant is unicast back to the source along the reverse route of the forward ant and updates local routing tables as described; the forward ant sequence number is maintained. PERA allows routing updates to be done either according to the ABC (Equation 3.2) or AntNet (Equation 3.4) method. It is not made clear how exactly the reinforcement parameter is calculated.

Data packets are routed deterministically to the next hop neighbor with the largest probability in the routing table for the particular destination.

Ant-like Routing Algorithm

This algorithm presents a detailed routing scheme for MANETs, including route discovery and maintenance mechanics [54]. It was developed by Günes, Kähmer, and Bouazizi between 2002 and 2003. The Ant-like Routing Algorithm (ARA) is the first MANET routing algorithm to use pheromone directly as a means to measure path quality. It very closely mimics the classic ant foraging algorithm instead of the more abstract ACO. ARA uses the same route discovery algorithm as PERA. Forward ants with unique identifiers are flooded into the network. Forward ants arriving at the destination are destroyed and a backward ant is returned to the source along the forward path. When a node receives a forward ant, it is allowed to update it routing table, or create a new entry for the source of the ant, with some constant amount of pheromone, γ_c.

In contrast to previous algorithms, routes are maintained by data packets themselves. Whenever a data packet is received or sent, a node will note the neighbor and add pheromone, γ_c, to that link. This is made clear in Equation 3.13,

$$\begin{aligned} P^n_{h,d} &\leftarrow P^n_{h,d} + \gamma_c \\ P^h_{r,s} &\leftarrow P^h_{r,s} + \gamma_c,\ r = n \end{aligned} \tag{3.13}$$

In order to balance this positive feedback, each node n in the network periodically decays the pheromone on each link for each known destination, $\mathcal{D}^n$, as shown in Equation 3.14.

$$\forall i \in \mathcal{N}^n,\ \forall d \in \mathcal{D}^n,\ P^n_{i,d} \leftarrow P^n_{i,d} \cdot e^{-\tau} \tag{3.14}$$

If a loop is detected by checking data packet sequence numbers, an error message is returned to the previous hop. The previous hop then removes all pheromone to the duplicate hop, making the loop impossible to complete in the future.

ARA presents a unique local repair method. If a link is found to be dropped, perhaps due to a MAC level communications failure, the pheromone on that link is set to zero. As in the loop detection case, this makes it impossible for this (now non-existent) link to be used. If another link is available, the packet is resent. If not, the packet is returned to the previous hop for similar processing. If a packet is eventually returned to its source with this method, a new route discovery process is initiated using the forward ant flood method.

Data is forwarded deterministically over the links with the most pheromone on them.

ARA2 The original Ant-like Routing Algorithm was improved in 2003 by the same authors [55]. This version incorporates several changes including, probabilistic data forwarding, continuous pheromone decay, ant packet prioritization, backward ant flooding, data packet buffering, and neighbor broadcast.

The original ARA had forwarded data packets deterministically, the same as all of the algorithms to come before it. ARA2 allows the use of probabilistic data packet forwarding. Next hop probabilities are computed by a normalization of the destination pheromone on outgoing links, as shown in Equation 3.15.

$$p^n_{i,d} = \frac{P^n_{i,d}}{\sum_{j \in \mathcal{N}^n} P^n_{j,d}} \tag{3.15}$$

The authors cite that the main advantage of this approach is that the traffic load is balanced across available paths.

ARA2 also introduces continuous pheromone decay. Previous proposals always decayed pheromone on a fixed regular period, such as one second. This feature is introduced in order to avoid oscillation between two paths with similar pheromone concentrations. Unfortunately no specific implementation details are given.

Both forward and backward ant packets are prioritized over data traffic in ARA2. This is a change from ARA where ants shared the same queues as data. Such a scheme allows network control information to be distributed faster into the network, as in previous schemes. It is noted that the implementation of ARA2 presented in [55] optimizes for hop count, where this approach works very well. Other metrics may be better served if the forward ants have the same priority as data. An example includes delay. This subtlety is mentioned in previous work.

Another effort to speed the dissemination of control information is the use of backward ant flooding. This should cover a larger portion of the network and cause network updates to be better distributed. Unfortunately no specific implementation details are given and this feature remains vague. A similar idea is presented in PERA where backward ants are unicast for each received forward ant.

ARA2 also works to limit the number of forward ant floods sent into the network by buffering received data packets which need a route request, rather than sending one out for each data packet.

Finally, ARA2 introduces the concept of neighbor broadcast to spread control information. Past SI algorithms have relied on the MANET broadcast environment solely to facilitate the use of packet flooding. Although, the use of packet eavesdropping has been used in algorithms such as DSR as well. ARA2 allows nodes to eavesdrop on their neighbor's transmission in order to gather information about where they are transmitting to. This information is added in the context of pheromone. No specific information is given on how this feature works exactly.

Termite

Termite, the subject of this thesis, is a contemporary of the original ARA. Work started on its development in the Fall of 2001 and the first report published in 2003 [56]. Many of the features presented in this chapter are also found in Termite, with future developments through 2004. A full treatment continues in the following chapters. Some features found in future publications by other authors reflect on those developed originally for Termite.

Ad-Hoc Networking with Swarm Intelligence

The Ad-Hoc Networking with Swarm Intelligence algorithm (ANSI) by Rajagopalan and Shen is the most recent addition to the SI MANET routing family in 2004 [57]. This algorithm is introduced with the intention of creating an algorithm with complementary proactive and reactive components, able to adapt to more flexible metrics, such as a node capability.

In contrast to many other SI routing algorithms, ANSI allows a proactive routing feature in addition to the standard reactive approach. This follows in the footsteps of other dual mode MANET routing algorithms such as the Zone Routing Protocol (ZRP) [58] and the Sharp Hybrid Ad-hoc Routing Protocol (SHARP) [59]. Such systems are generally defined such that a node proactively keeps track of routes to all destinations within a certain zone, and routes to the rest are maintained reactively. A separate proactive routing protocol and reactive routing protocol are defined to handle each case. The hybrid protocol is often designed such that any previously defined proactive and reactive algorithms can simply be plugged in. The hybrid protocol then defines how zones are managed. The zone of a node is an area defined by a given hop diameter. The zone may be predefined or calculated online according to the needs of the network [60]. In ANSI, proactive ants are sent periodically by each node in order to establish routes to their source. This is similar to the route discovery approach used by PERA. The impact of the flood is minimized by using packet identifiers, a limited time-to-live in each packet, and a probabilistic flooding (gossiping) scheme. As each proactive ant moves through the network, it updates the current node with information from its stack.

ANSI follows the PERA method for reactive routing to non-local destinations. Forward ants are flooded while keeping their path and metric information on a stack. When a forward ant arrives at the destination, a backward ant is unicast along the forward ant's reversed path to the source. However, the forward ant flood radius is increased iteratively, equivalent to AODV's optional expanding ring search. Full route discovery floods are typical. When a backward ant is received at a node, local pheromone is updated according to Equation 3.16.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ \forall d \in S_{visited},\ P^n_{i,d} &\leftarrow P^n_{i,d} \cdot e^{-\left(t-t^n_d\right)\tau} \\ \forall d \in S_{visited},\ P^n_{r,d} &\leftarrow P^n_{r,d} + \gamma^S_{r,d} \end{aligned} \tag{3.16}$$

Here, $\gamma_{r,d}^{S}$ is the amount of pheromone to be placed on the link to the previous hop r for destination d by the backward ant. This is equivalent to the value $\left(c_{r,d}^{S}\right)^{-1}$ from CAF. Pheromone is interpreted as utility, the inverse of path cost. In the context of the decay parameter, $e^{-\Delta t \cdot \tau}$, the current time is t, and the last time that the pheromone from destination d at node n was updated is t_d^n. This is an example of continuous pheromone decay.

The utility, u, of using a link is computed according to Equation 3.17. This is the same equation used in ACO, incorporating two measures of link utility. Pheromone, P, measures global path utility, and η is the estimated metric distance to the destination. Data packets are routed over the link with the highest utility.

$$u_{i,d}^{n} = \frac{[P_{i,d}^{n}]^{\alpha}[\eta_{i,d}^{n}]^{\beta}}{\sum_{j \in N^n}[P_{j,d}^{n}]^{\alpha}[\eta_{j,d}^{n}]^{\beta}} \tag{3.17}$$

ANSI also describes a detailed route error and recovery procedure. In the event of route error, an intermediate node performs a number of tasks. These include buffering the data packet, sending a route request forward ant flood to search for the destination, and sending a route error ant back to the source of the packet. The packet will be buffered until the route request returns a result. The route error ant is unicast back to the source, letting it and all intermediate nodes invalidate routes to the destination. This is accomplished by setting the pheromone on the offending link to zero. The source route must then issue a new route request to find a route to the destination for all subsequent packets.

3.5.3 Discussion

It should be clear by this point that all of the SI routing proposals are basically equivalent. Whether they assume symmetric link costs like ABC or asymmetric link costs like AntNet, whether routing updates affect link probabilities directly or through an intermediate pheromone metric, each algorithm is essentially doing the same thing. Each approach may have a different name for different parameters, however analogies can be drawn to show equivalency. In fact, not only are they equivalent between themselves, but as the previous chapter showed, there is also an equivalency to algorithms found in other fields such as optimization and artificial intelligence.

It may be difficult to differentiate between proposals and find the best solution for a particular application. But as the principles of swarm intelligence posit, there is strength in a diversity of approaches. The strongest solution will emerge.

3.6 Related Approaches to Routing

A number of approaches to routing have been developed in the past that bear a resemblance to swarm intelligent designs. For this reason they merit mention and understanding. For example, the idea of using roaming software agents in order to perform network services has been around for some time. While the ant packets of SI are small and simple, agent-based networking suggests using much more capable mobile programs. Simple probabilistic routing was developed for the first computer networks some forty years ago. Next-hop probabilities were precomputed and static. A probabilistic broadcast scheme called gossiping has been studied in an effort to reduce the load of traditional flooding. Artificial Intelligence has also made some attempts to solve the routing problem. Reinforcement learning techniques have been applied using the concepts of link utility and feedback signals.

3.6.1 Agent-Based Routing

A related approach to SI routing is known as agent-based routing. In this paradigm, a discrete number of agents traverse the network and measure and update various parameters at each node that they visit. Clearly this idea is very similar to the ant packets used in SI algorithms. The difference is that agents are generally considered to be more capable in terms of their execution and data storage than simple ants packets. Agents may be able to carry with them program code as well as program data as they move through the network. Ant packets are generally only considered to carry some small amount of data. Ants may be considered to be a simple agent.

An example of agent-based routing is the Agent-based Distance Vector Routing (ADVR) [61]. It is introduced in 2001 as a generalized distributed distance vector routing algorithm in which network data is moved about by independent agents. In retrospect, this scheme is similar to CAF which uses ant packets to distribute network control data and to perform distance metric updates. ADVR presents these same ideas, however it is able to generalize the mathematics somewhat with regards to forwarding strategies.

Another example is the 2002 Ant-AODV which follows the AODV routing rules, but uses agents, also called ants, to distribute routing updates [62]. The algorithm does not use any sort of randomness as may be implied by the name. Instead, the ant packets are only used to pass route information around the networks. The AODV based portion can still use RREQ floods for route discovery. A RREP may be more readily available due to the updating activity of the ant packets. The ants are thus intended to reduce the route discovery latency, however it really just makes the algorithm proactive, similar to DSDV.

Agents need not be used only for routing. They can be used to manage any sort of service on a network [63]. Other examples include network diagnostics, remote maintenance of heterogeneous elements, service provisioning, component provisioning, and performance management.

3.6.2 Probabilistic Routing

The fact that probabilistic routing is such a large component of the SI based routing algorithms, and that SI is a relatively new optimization paradigm, does not mean that probabilistic routing has not been used in the past. In fact, probabilistic routing was first introduced as a means of load balancing in the early 1960s. Since a packet's path is not deterministic, it has also been proposed to the use the technique as a security measure. In general, by exploiting the geometric properties of computer networks, messages can be delivered reliably even with probabilistic retransmission.

Stochastic Routing

The idea of using random routing has been around since the first computer networks [64]. When packets are routed deterministically, certain paths may become congested because so much traffic is being sent on it. Stochastic routing is a simple solution to this problem where packets are sent to a random next-hop, allowing packets to be evenly spread over a larger number of paths and thus reducing congestion. The disadvantage to such an approach is that it is increasingly likely that packets will be received out of order and with a larger delay spread. There has not been a large amount of work in this area because deterministic routing schemes are much easier to understand and design, and their performance is good enough. Early stochastic routing schemes used static predetermined probabilities, while more recent work has allowed for dynamic next-hop probabilities.

Multipath Routing Stochastic routing may also be considered a form of multipath routing; packets are sent across multiple paths to a destination. More packets could be sent at a time since more network resources are available, the delivery could be more reliable since there are redundant delivery options, and security could be improved because an attacker would have to monitor multiple points in the network in order to intercept all packets of a message.

Gossiping

Gossiping is a probabilistic optimization of flooding which attempts to have every node receive some information without requiring each node to transmit it as well [65]. The basic gossiping scheme in a broadcast communications medium, such as an ad-hoc network, requires each node receiving a message to rebroadcast it with some probability. While the modification is slight, the reduction in the number of transmissions can be significant, up to 35% for random topologies [66]. Gossiping in a wired network requires each node to randomly select a number of neighbors to retransmit the received message to.

Gossiping also exhibits an interesting feature known as a phase transition [67]. As the network becomes large, the probability that all nodes or no nodes will receive a message becomes extremely sensitive to the retransmission probability. There is a critical probability below which almost no nodes will receive the gossiped message, and above which almost all will. This threshold is dependant on many factors including network size and density.

3.6.3 Q-Routing: Routing with Reinforcement Learning

Boyan and Littman introduce Q-routing in 1993 as an application of the Q-learning reinforcement learning technique to the problem of network routing [68]. The authors' perspective is entirely from that of reinforcement learning and artificial intelligence. With the hindsight of the development of swarm intelligent routing in the late 1990s, Q-routing may well be thought of as one of the first swarm intelligent routing algorithms. It predates ABC by three years.

Q-routing works simply as follows. Each node maintains an estimate of the time necessary to reach every other node in the network through each neighbor, $C^n_{i,d}$. When a packet is sent from a node n to the next hop h for destination d, n receives a feedback signal from h consisting of h's best estimate of arriving at d. Node n will then update its cost estimate to arrive at d according to Equation 3.18.

$$C^n_{h,d} \leftarrow (1-\varphi)\cdot C^n_{h,d} + \varphi \cdot (c_{n,h} + \min_{j\in\mathcal{N}^h} C^h_{j,d}) \tag{3.18}$$

The authors note that this is very similar update rule found in Bellman-Ford algorithm. The authors of CAF make the same observation regarding their algorithm some five years later. Since Q-routing is designed from the reinforcement learning perspective, φ is explicitly referred to as the learning rate. This again makes the similarities between the swarm intelligent algorithms and reinforcement learning very clear.

CHAPTER 4

TERMITE

4.1 Introduction

Termite is a distributed routing algorithm for mobile wireless ad-hoc networks [56]. It is designed using the swarm intelligent framework in order to achieve better adaptively, lower control overhead, and better packet delivery than contemporary solutions. The algorithm is inspired by the hill building behavior of termites.

A social insect metaphor suggests a probabilistic routing algorithm. Information about the network environment, such as topology, link quality, and traffic congestion, is determined from the amount of pheromone, or path utility estimate, contained on each arriving packet at each node. Packets are considered to route themselves and are able to influence the paths of others by updating routing parameters throughout the network. The collection of these parameters from all nodes across the network constitute the environment in which the packets exist. This Termite environment is a representation of the collective knowledge of all nodes. The interaction between packets and their environment implicitly spreads information about network conditions and thus reduces the need to generate explicit control traffic. In general, the method of communicating information indirectly through the environment is known as stigmergy.

4.2 Design of Termite

The Termite algorithm is explained in detail below, however it may be described simply as follows. Each node in the network has a specific pheromone scent. As packets move through the network on links between nodes, they are biased towards the pheromone of their destination. Packets follow this pheromone gradient while laying pheromone for their source on the links that they traverse. The amount of pheromone deposited by a packet on a link depends on the utility of the path that it has traversed. Packet pheromone is proportional to the utility of the path followed up to the current node in the network. Using this method of pheromone updating, consistent pheromone trails are built through the network. Changes in the network environment, such as topological or path quality changes, are accounted for by allowing pheromone to decay over time. This requires paths to be continuously reinforced by new traffic; new information about the network is added to links. Each network node records the amount of pheromone that exists for each destination on each of its links.

4.2.1 The Pheromone Table

In order to manage the pheromone in the network, each node maintains a table recording the pheromone on each neighbor link from each destination node. Each node has a distinct pheromone scent. The table may be visualized as a matrix with neighbor nodes listed along the side and destination nodes listed across the top. Rows correspond to neighbors and columns to destinations. An entry in the pheromone table of node n is referenced by $P^n_{i,d}$, where $i \in \mathcal{N}^n$ is a neighbor node, and $d \in \mathcal{D}^n$ denotes a destination. In other words, $P^n_{i,d}$ is the amount of pheromone from node d on the link with neighbor i at node n. An example of a pheromone table is shown in Figure 4.1. $\mathcal{N}^n$ and $\mathcal{D}^n$ are the sets containing the current neighbors and destinations known to node n, respectively.

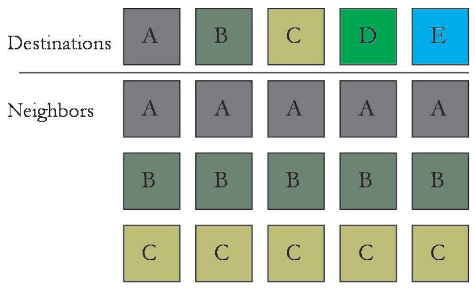

Figure 4.1: Example of a Pheromone Table

4.2.2 Pheromone Update

When a packet is received at a node n from source s and previous hop r, the pheromone entry $P^n_{r,s}$ is updated in the pheromone table according to Equation 4.1 with a constant amount of pheromone, γ_c. Each receiving node should update its pheromone table in this way, even if it is not the intended receiver of the packet. Using such promiscuous mode reception allows routing information to be spread more quickly.

$$P^n_{r,s} \leftarrow P^n_{r,s} + \gamma_c \tag{4.1}$$

4.2.3 Pheromone Decay

To account for pheromone decay, each value in the pheromone table is periodically multiplied by a decay factor, $e^{-\tau}$. The decay rate is $\tau \geq 0$, and is a global parameter. A high decay rate will quickly reduce the amount of remaining pheromone, while a low value will degrade the pheromone slowly. The nominal pheromone decay interval is one second; this is called the decay period. Equation 4.2 describes pheromone decay.

$$\forall i \in \mathcal{N}^n,\ \forall d \in \mathcal{D}^n,\ P^n_{i,d} \leftarrow P^n_{i,d} \cdot e^{-\tau} \tag{4.2}$$

If all of the pheromone for a particular node decays, then the corresponding row and/or column is removed from the pheromone table. Removal of an entry from the pheromone table indicates that no packet has been received from that node in quite some time. It has likely become irrelevant and no route information must be maintained. A column (destination listing) is considered decayed if all of the pheromone in that column is equal to a minimum value. If that particular destination is also a neighbor then it cannot be removed unless all entries in the neighbor row are also decayed. A row is considered decayed if all of the pheromone values on the row are equal to a minimum value.

Neighbor nodes must be specially handled because they can forward packets as well as originate packets. A decayed column indicates that no traffic has been seen which was sourced by that node. Since neighbors can also forward traffic, their role as traffic sources may be secondary to their role as traffic relays. Thus, the neighbor row must be declared decayed before the neighbor node can be removed from the pheromone table.

If a neighbor is determined to be lost by means of communications failure (the neighbor has left communications range), the neighbor row is simply removed from the pheromone table.

4.2.4 Pheromone Bounds

There are three values governing the bounds on pheromone in the table. These are the *pheromone ceiling*, the *pheromone floor*, and the *initial pheromone*. When a packet is received from an unknown source, a new entry for that node is created in the pheromone table. In the case of a neighbor node, a new column and row will be created (neighbor nodes are also potential destinations). If the source is not a neighbor only a column is entered into the table. Each pheromone value in the new cells will be assigned the initial pheromone value. During the course of pheromone decay, no value is allowed to fall below the pheromone floor. This allows unused nodes to be easily detected. Likewise, no pheromone value is allowed to exceed the pheromone ceiling. These bounds prevent extreme differences in pheromone from upsetting the calculation of next hop probabilities. Each parameter may be tuned for a particular network environment.

4.2.5 Route Selection

In order to forward a packet towards its destination, the forwarding equation is used to determine the next hop neighbor. This formula transforms the pheromone for destination d on each outgoing link i, $P^n_{i,d}$, to the probability that that link will be used to forward the packet, $p^n_{i,d}$. The specific next hop neighbor is chosen randomly according to this distribution, though no packet is ever returned to the node that it arrived from, r. Thus, i is chosen from $\mathcal{N}^n - r$. The forwarding equation is shown in Equation 4.3.

$$p^n_{i,d} = \frac{(P^n_{i,d} + K)^F}{\sum_{j \in \mathcal{N}^n - r}(P^n_{j,d} + K)^F} \tag{4.3}$$

The constants F and K are used to tune the routing behavior of Termite. The *pheromone threshold*, K, determines the sensitivity of the probability calculations to small amounts of pheromone. If $K \geq 0$ is large, then large amounts of pheromone will have to be present before an appreciable effect will be seen in the routing probability. The nominal value of K is zero. Similarly, the *pheromone sensitivity*, $F \geq 0$, may be used to modulate the differences between pheromone amounts. For example, $F > 1$ will accentuate differences between links, while $F < 1$ will deemphasize them. $F = 1$ yields a simple normalization.

4.2.6 Packet Design

There are five types of packets used by Termite. These are *data*, *hello*, *seed*, *route request (RREQ)*, and *route reply (RREP)*. The latter four types are control messages. Each packet type contains at least six fields, including *source address*, *destination address*, *previous hop address*, *next hop address*, *message identification*, and *Time-To-Live (TTL)*. The previous hop address and next hop address fields may be removed from the packet and the information instead extracted from the MAC header which encapsulates the routing data. Issues of address resolution may arise; MAC addresses are rarely the same as routing addresses. The message identification field allows each packet in the network to be uniquely identifiable. This feature is not required, however it enables loop detection as well as allowing the nodes to operate in a limited promiscuous mode. Data packets may contain additional fields such as *data length* and *bulk data*.

Data Packets

Data packets are routed normally through the network. If a node does not know how to forward a packet, which is the case when the node's pheromone table does not contain the packet's destination, the packet is buffered and a route request is issued. If a reply is not received within a given time period, *rreq timeout*, the data packet is dropped and considered lost.

A route reply will ensure that pheromone for the packet's destination exists in the pheromone table, thus enabling proper routing. A reply also ensures that there exists at least one pheromone trail from the requestor to the node which replies to the route request.

Route Request Packets

Route request (RREQ) packets are sent when a node needs to find a path to an unknown destination. Route requests perform a random walk over the network until a node is found which contains some pheromone for the requested destination. In a random walk, a packet uniformly randomly chooses its next hop, except for the link it arrived on. If a route request cannot be forwarded, it is dropped. Pheromone is disregarded during a random walk. Any number of RREQ packets may be sent for each route request; the exact number of which may be tuned for a particular environment.

A route request is not looking for an explicit route to the destination. Rather it is searching for the beginning of a pheromone trail to the destination. The route will be strengthened by future communications.

Route Reply Packets

Once a route request packet is received by a node containing pheromone to the requested destination, a route reply (RREP) packet is returned to the requestor. The RREP message is created such that the source of the packet appears to be the requested destination and the destination of the packet is the requestor. The reply packet extends pheromone for the requested destination back to the requestor without any need to change the way in which pheromone is recorded at each node. The reply packet is routed normally through the network by probabilistically following a pheromone trail to the requestor. Intermediate nodes on the return path automatically discover the requested node.

Hello Packets

Hello packets are used to search for neighbors when a node has become isolated. Hello packets are broadcast at a regular interval until a reply is received. A reply is sent by all nodes who hear the original hello. Any replies will create an entry in the pheromone table (since the table was previously empty) and thus make the node aware that it is not alone. The node will stop sending hello packets until the pheromone table is again empty.

As has been suggested in previous work, hello broadcasts may be avoided at the routing layer by an analogous mechanism at the MAC layer.

Seed Packets

Seed packets are used to actively spread a node's pheromone throughout the network. Seeds make a random walk through the network and serve to advertise a node's existence. They can be useful for reducing the necessary number of explicit route request transactions.

4.3 Properties of Termite

This section highlights some of the properties and variations of Termite.

4.3.1 Probabalistic Routing

All routing decisions in Termite are random. This simplifies the processing of each packet, and allows new paths to be explored at a rate inversely proportional to the quality of the current routing solutions. The use of tuning parameters F and K allows a tradeoff between route exploration and exploitation.

4.3.2 Best-Effort Service

The random routing of packets allows known routes to be used while still exploring the network for new and better routes. There is always a chance that poor routing decisions will be made and a packet will never arrive at its destination. A packet may be routed in a loop or perhaps in an entirely wrong direction. A Time-To-Live (TTL) field is used in order to prevent the propagation or reinforcement of bad routing decisions. The probability of a packet persisting in a loop tends to zero with the length of the loop; the number of packets that are lost to looping will not adversely affect the overall routing performance. If loops cannot be tolerated, the message identification field in each packet may be used to record previously processed or overheard packets. Packets in this list will be dropped if received a second time.

Termite is designed to quickly find an *acceptable* routing solution and to adapt gracefully as the network changes.

4.3.3 Low Complexity

Termite is a simple algorithm. The memory footprint can become large, storing a maximum number of pheromone values equal to the square of the total number of nodes in the network. The is the same as any distance vector based routing algorithm. Computation time for next-hop probabilities may be reduced by caching probability results or reducing the frequency with which updated probabilities are calculated. Each packet is processed in one pass, including the updating of the pheromone table and the next hop computation. It is not necessary to keep track of special purpose information such as sequence numbers or route setup attributes.

4.3.4 Low Route Recovery Latency

Termite provides low route recovery latency. If a neighbor link is lost, future next hop calculations will simply not consider the pheromone that was on that link. Packets transmitted to neighbors that are no longer able to communicate may be retransmitted to those that can. Unless a node is entirely unaware of a particular destination, a next hop can be computed immediately.

Given a sufficient amount of traffic from each node, little pheromone will decay to the point of being lost entirely from a pheromone table. Few control packets must be sent, which could delay the next hop calculation.

4.3.5 Adaptability

Termite is able to adapt to the aggregate effects of all factors affecting throughput. Many effects influence the rate at which a message may be transmitted, and thus the rate at which messages arrive at their destinations.

In order to find a shortest hop path, packets traversing a shorter path will arrive at the destination sooner than those on longer paths. This will influence packets traveling in the opposite direction to travel the shorter path, which will in turn bias more traffic onto the shorter path.

In the case of variable quality links, the network will learn to avoid low quality links. For instance, in order to maintain an acceptable bit error rate (BER) across low quality links, a medium access protocol can be expected to reduce the bit rate. This will result in a lower throughput, and perhaps also increased traffic congestion at that node as transmit queues are filled. Fewer arriving packets leads to a lower regeneration rate of pheromone and thus lower bias for packets traveling in the opposite direction.

The MAC layer may take a different approach to solving the link BER problem, such as increasing the transmit power. This may maintain an acceptable bit and error rates, but the larger transmit radius will influence traffic patterns in the area. The node will prevent others from transmitting while it is doing so. This will alter the rate at which packets are able to be transmitted between nodes and thus change the routing probabilities.

Protocols have been suggested that base their routing strategies on measured topology volatility, especially that of the local topology [70]. Termite automatically measures this; many transmissions will be overheard from long-time neighbors, resulting in high pheromone levels on the links to those neighbors. Stable neighbors will be preferred for routing.

4.3.6 Multipath Routing

Termite provides multipath routing. Each routing decision is probabilistic and independent from all other decisions. No two packets are guaranteed to take the same path through the network. After all, data packets are expected to explore the network.

As a network topology becomes more stable, routes between nodes may collapse to a single high probability path. This will be the case if there is one path which is more optimal than all others, or it may simply be a matter of happenstance. In the latter scenario, the forwarding function can be tuned to equalize the link probability.

4.3.7 Stigmergy

Termite is built on the strength of the traffic flowing through the network. No direct communication between packets, nodes, or any sort of higher level network abstraction exists or is even necessary. Packets are routed simply by interacting with the pheromone environment around them. Stigmergy is fundamental to Termite.

4.3.8 Mobile Agents

In the mobile agents paradigm, agents traverse the network while updating and optimizing routing information at each node. Termite may be viewed in this framework, where each packet is an agent updating each node which overhears it. Information conferred includes where the packet came from immediately and originally, as well as how quickly it has traveled through the network relative to other packets. Each node to which a packet is directly forwarded will execute the agent; the packet will forward itself by detecting its environment and making a decision.

Because every packet in the network would be considered an agent in this framework, we do not believe Termite to fall into this category. Rather, routing solutions are an emergent property of the interaction of all packets. Little information must be stored in each packet in order for the packets to route themselves.

4.3.9 Promiscuity in a Broadcast Medium

Termite can take advantage of the fact that many MANETs exist in a broadcast medium. While this algorithm ultimately abstracts communications with neighbors to discrete links, it is possible for each node to promiscuously listen to all transmissions. Routing information can be gained from listening to all traffic, rather than only to specifically addressed traffic. New nodes can quickly be detected when their transmissions are overheard. Also, a great deal of information about the network can be gained from the destinations that neighbors are forwarding for.

While promiscuity can boost the performance of Termite, it also creates some problems. If the same packet is overheard more than once, the routing information derived from it may result in misleading pheromone gradients. In order to prevent the double counting of packets, a message identification field is included in Termite packets. Suppose a packet from node s to d is forwarded from x to neighbor y, and then y again forwards the packet to z. Node x will overhear the transmission from y to z (since x and y are neighbors), and updates its pheromone table with the overheard packet. This would imply that there exists a path through y to s. But the path through y is already through x! If x naively records the packet in this manner, it might be tempted to forward a packet destined for s to y, which then may pass the packet back to x. This is somewhat related to the distance vector counting to infinity problem. In order to prevent such a scenario, each node should maintain a list of previously forwarded packets as well as packets that it has overheard (but were not addressed specifically to it). Packets in this list are dropped if they are received again; they cannot be used to update the pheromone table. This process will also prevent packets from following loops in the network; previously forwarded packets will be destroyed.

4.3.10 Energy Consumption

Eavesdropping on all communications requires a great deal of energy. With current technology, the energy cost of receiving a message is almost as high as that of transmitting one [71]. As many ad-hoc applications are power limited, this can become a major drawback to the Termite approach.

4.3.11 Directional Links

Termite assumes symmetric bidirectional communications links between nodes. When a packet is received directly from a particular node, it is assumed that packets may be forwarded in the opposite direction. Unidirectional links will break Termite as it has been described in this paper. Sub-Layer schemes have been proposed to abstract unidirectional links into bidirectional links [72].

4.4 Simulation and Results

Termite has been simulated using Opnet Technologies Inc.'s, Opnet Modeler [74]. A series of tests are run in order to determine the viability of Termite as an effective scheme for MANET routing. Data goodput and control overhead are the primary metrics; Node speed is the independent parameter.

4.4.1 Simulation Environment

Simulations are executed in scenarios containing one hundred nodes and lasting 600 seconds. Each scenario uses the same simulation parameters as listed in Table 4.1. Only node speed is varied and is indicated in the results.

Table 4.1: Simulation Parameters

simulation area	50 x 50 [meters^2]
transmission range	10 [meters]
channel bit rate	1 [Mbps]
initial pheromone	1
pheromone ceiling	10000
pheromone floor	0.1
rreq timeout	2 [seconds]
τ (decay rate)	0.105
decay period	1 [second]
Data TTL	32 [hops]
RREQ TTL	32 [hops]
RREP TTL	32 [hops]
Seed TTL	4 [hops]
Seed period	30 [seconds]
Hello period	1 [second]
RREQs per Route Request	2

Random Waypoint Mobility Model

The random waypoint mobility model is used in this simulation. Movement is restricted to a rectangular area. Generally, each node remains stationary for a uniformly distributed period of time, chooses a uniformly random destination in the area, and then moves to that destination in a straight line at a uniformly random speed. In this simulation, there is a 60 second pause time at the beginning of the simulation in order to allow the network to reach an equilibrium state. Otherwise there is no pause time and all nodes have the same speed. These parameters are not subject to the decreasing average speed problem encountered when setting the bounds on speed too far apart [75].

Traffic Source Model

A constant bit rate (CBR) traffic source model is used. Each node sends one data packet every half second to a random destination which is selected at the beginning of the simulation. The data size of each packet is 64 bytes in order to minimize the effects of congestion on the network. A reply is returned to the source when a packet arrives at its destination. This is to simulate an acknowledgement.

MAC Layer Model

A perfect MAC layer is used in this simulation. If two nodes are within communications range, then they may communicate immediately without error or transmission collision. Transmission and receive queues are modeled; a packet must wait until all packets ahead of it in the queue are serviced before it is processed. Such a simple model is used in order to test the behavior of the algorithm without the influence of extra-layer effects.

4.4.2 Simulation Results

In order to make an initial assessment of the algorithm, data goodput and control overhead are measured against average node speed. Results are shown in Figure 4.2. Data goodput is the fraction of successfully delivered data packets. Overhead is measured in three ways; Bandwidth overhead is the fraction of all transmitted bits that belong to a control packet. Packet overhead is the fraction of all transmitted packets that are of the control type. Using these measures, packets transmitted multiple times will be counted in each instance. Control overhead is the fraction of all sourced packets that are of the control type. Packets are only counted when they are created and not again. Data goodput declines as

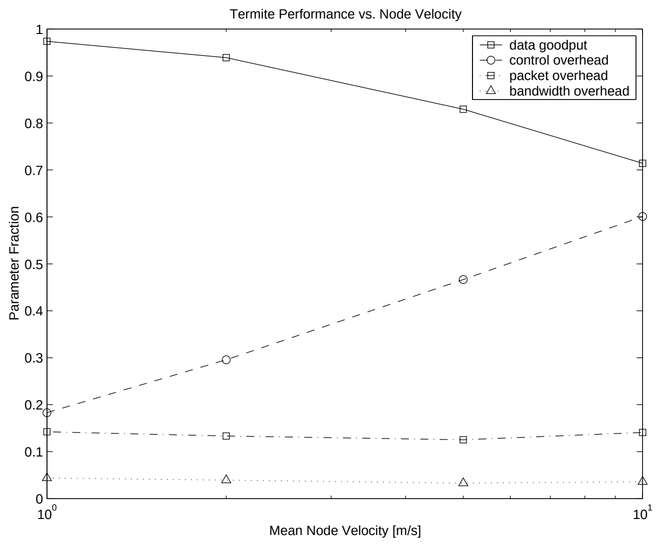

Figure 4.2: Data Goodput vs. Control Overhead vs. Average Node Speed

node mobility increases. Control overhead experiences a nearly linear increase on the semilog plot, suggesting a logarithmic control packet generation rate in the node mobility region of interest. Bandwidth and packet overhead remain constant throughout the simulation.

Figure 4.3 shows how the average hop count of data and route request packets varies with node mobility. The average path length of successfully delivered data packets experiences a sharp increase as the network becomes more volatile. However, the path length of both successful route request and route replies remains constant and quite low.

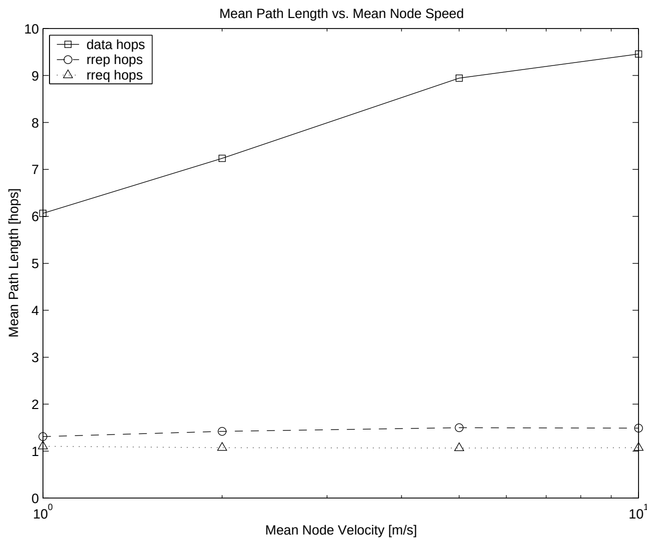

Figure 4.3: Mean Packet Path Length vs. Mean Node Speed

In order to more clearly illustrate how data packets take longer paths with higher node mobility, Figure 4.4 shows how route confidence varies against average node speed. Route confidence is the average next-hop probability. Routes are extremely well established in a static network; average next-hop probabilities are close to unity. As the network topology becomes unreliable, packets move through the network with significantly less confidence; the average next-hop probability drops and low probability next-hops are dominant. These conditions yield longer paths as packets wander through the network; they are no longer closely guided to their destination. More paths are explored, most of which are long.

Figure 4.5 demonstrates the distance of nodes being requested from the requestor with respect to average node speed. The results are normalized to the

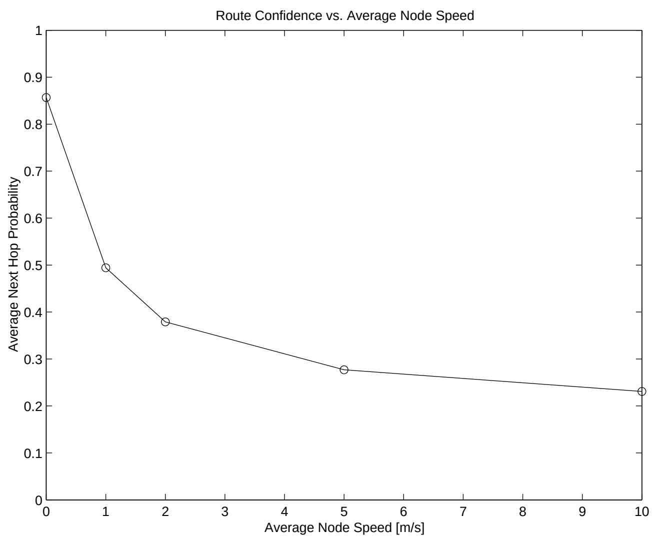

Figure 4.4: Route Confidence vs. Mean Node Speed

transmission range which is ten meters in this simulation. The requested destinations tend to be within two hops, however higher node speeds quickly find the requested nodes to be farther away.

Finally, an indication of network volatility is shown in Table 4.2. Volatility is measured by the average number of link state changes per node per second. A change in link state is defined as an existing neighbor link failing, or a new neighbor link discovered. This metric observes the rate with which the network topology is changing from an individual node's perspective. The increase in number of link changes is approximately linear with average node speed.

4.4.3 Discussion

Figure 4.2 confirms that Termite is able to perform well over a variety of volatile network environments. Bandwidth overhead remains constant regardless of node mobility. This is despite the fact that control overhead can become quite large. Packet overhead mirrors the characteristics of bandwidth overhead although it is

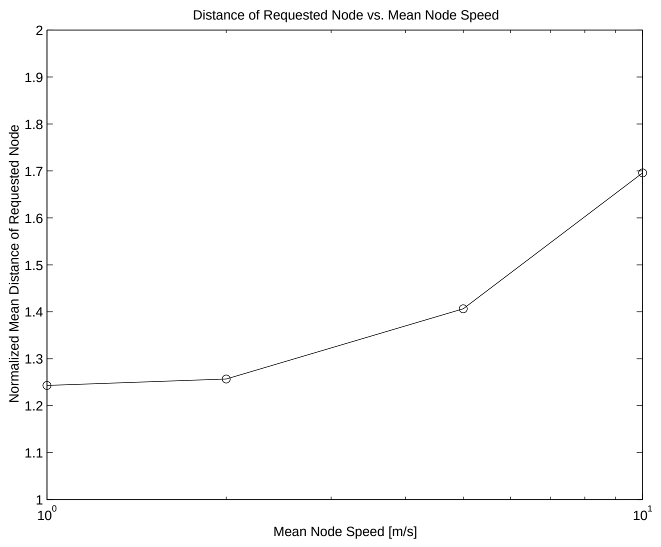

Figure 4.5: Normalized Distance to Requested Destination vs. Mean Node Speed

Table 4.2: Mean Link State Changes per Second per Node

0 [m/s]	0 [link changes/node/s]
1	1.18
2	2.41
5	5.88
10	11.25

somewhat larger. This characteristic is in stark contrast to other MANET routing algorithms where overhead attributed to control packets is nontrivial in general and can become a dominating factor in network resource consumption [73]. Figure 4.3 demonstrates that as the network becomes more volatile, data paths become longer while control paths remain small. The reason for this is explained by Figure 4.4, where data packets are quickly less confident in their next-hop selections as the network becomes unstable. Packets tend towards a random walk over the network as volatility increases. Simple calculations based on node density indicate that the average number of neighbors per node in this simulation is slightly less than thirteen. Thus, 0.08 is roughly the route confidence for a random walk. Though this metric is much higher than that in all cases, data goodput degrades greatly.

Even though many more control packets may be generated in volatile situations, they do not have to travel very far to meet their objective. One explanation for this behavior draws on the results of Figure 4.5. The majority of requested destinations are within two hops of the requestor. This suggests that paths are being broken only at the very end of a trip. The destination moves outside of the communications range of the penultimate node, which then must issue a route request for it. The destination node has not moved very far and can be reached within two hops. Likely an intermediate node between the penultimate and the destination is aware of the destination (since it is within communications range of both) and thus is able to reply to the route requestor. In this way, route requests are kept very short. A second possibility as to why control paths are so short, is that most nodes are aware of most destinations due to packet exploration of the network. Nodes record the source of all packets they receive; as long as enough traffic is received from that source to prevent its pheromone from decaying, a node will always maintain a knowledge of that source. If a route request is issued, it is highly likely that a nearby node will be aware of the requested destination; pheromone does not decay very quickly in this simulation. This characteristic of Termite is facilitated by all packets exploring the network on their way to a destination.

Another perspective on these results indicates a tradeoff. As network volatility increases, data paths become significantly longer and control paths stay short. Data packets are being used to explore the network while the bandwidth impact of control packets is minimized. But as data paths become longer, potentially large packets will be transmitted many times, negatively impacting energy use as well as network capacity. There is a tradeoff between the amount of bandwidth used by control packets and the additional amount of bandwidth used by transmitting large data packets over longer paths.

These results are promising; data goodput is maintained while packet overhead is kept constant, even in highly volatile networks. But there remain many open questions as to the specific operation of Termite. For instance, it is unclear what the optimal value of the system parameters are; the most notable of these parameters being the decay rate. How can parameter values be automatically determined based on locally available information. In these simulations, the parameters were chosen empirically but are not necessarily to be optimal. The optimal performance or even the optimal parameter values of this algorithm are unknown at this time.

4.5 Conclusion

A routing algorithm for mobile wireless ad-hoc networks has been presented. Swarm intelligence is used to build an emergent routing behavior. Packets probabilistically follow pheromone trails to their destination while laying pheromone for their source. Passive route marking reduces the need for explicit routing traffic and maximizes the network resources available to carry user data. Nodes determine network conditions by monitoring traffic flow and make adjustments to their routing tables. Simulations show that Termite is able to maintain reasonable data goodput over a variety of mobility conditions. Control bandwidth overhead is minimized and remains constant across several degrees of network volatility. Further work exploring the characteristics of the underlying algorithm is warranted. Termite is in the first generation of biologically inspired MANET routing algorithms.

CHAPTER 5

PHEROMONE UPDATE IN SI MANETS

5.1 Introduction

The previous chapter has shown Termite to be a viable MANET routing algorithm. Termite effectively applies the principles of swarm intelligence to provide a novel solution to the routing problem. The biological analogy of ants and pheromone is quite poetic. There is nevertheless an opportunity to improve performance even further through the precise and dedicated application of engineering acumen. By regarding pheromone based routing not just as an algorithm based on a strict metaphorical interpretation of biology, but as a general framework for probabilistic routing, performance is enhanced by moving away from a biological perspective [76].

This chapter tests and develops pheromone accounting strategies in the context of the Termite routing algorithm. Termite is briefly reviewed, and new features are added in order to improve performance. A number of pheromone accounting techniques are presented which can be used to determine pheromone levels and routing probabilities. Appropriately tuned parameters can approximate the deterministic approach used by ABC and AntNet to forward data packets and thus provide a smooth transition between hard and soft routing. In some of the techniques considered below, Termite is reformulated in order to make hard routing decisions and also include an explicit exploration probability. There is also a question of how certain parameters affect performance. Each of the accounting methods is tested across a wide range of pheromone sensitivity and decay rate.

5.2 Termite

This section will review the swarm intelligent Termite MANET routing algorithm. A number of new features have been added in order to improve performance over the original description in the previous chapter. Termite may be described simply as follows. Each node in the network has a specific pheromone scent. As packets move through the network on links between nodes, they are biased to move in the direction of destination pheromone gradients. Packets follow this gradient while laying pheromone for their source on the links that they traverse. The amount of pheromone deposited by a packet on a link may be equal to the utility of its traversed path. Using this method of pheromone updating, consistent pheromone trails are built through the network. Changes in the network environment, such as topological or path quality changes, are accounted for by allowing pheromone to decay over time. This requires paths to be continuously reinforced by new traffic; new information about the network is added to links. Each node records the amount of pheromone that exists for each destination on each of it's links. This creates a routing table similar to those found in traditional distance vector routing algorithms.

There are two major operations in Termite. These are pheromone accounting and packet forwarding. The former is responsible for maintaining current information about the network. This includes updating the routing table when packets are received as well as removing old information over time. The latter uses the available information about the network to route packets; the probability with which to forward a packet over a given link to its destination is determined.

5.2.1 Pheromone Accounting

The description of laying pheromone on a link is left vague in the previous description. The process of accounting for the amount of pheromone on a link, or even the eventual physical meaning of that number, can take many different forms.

The purpose of this process is to find a measure of the ability of a particular link to deliver a packet to a destination. This estimate is based strictly on information carried by local traffic. The traditional method for tallying pheromone mimics real pheromone in the sense of additive increase and exponential time-based decay. This allows old concentrations to be removed temporally while new ones are created as the network changes. The method is known as pheromone filtering. It is described in detail in the next section as the classic pheromone filter, and is analyzed in the following chapter. Alternative methods are also proposed.

Besides a specific pheromone accounting scheme, three general methods are described which assist in maintaining current and reliable utility estimates.

True Continuous Pheromone Decay

The SI framework requires that pheromone be decayed in order to remove old information from the system. Previous routing implementations such as Termite [56] and the Ant-like Routing Algorithm (ARA) [54] would decay pheromone on a regular predefined period. ARA2 introduces continuous pheromone decay [55]. Superior performance is reported by decaying pheromone based on packet interarrival times. Older information is removed immediately and proportionally upon the addition of new information.

This chapter introduces true continuous decay in which pheromone is decayed based on pheromone observation interarrival times. Not only does pheromone decay in between packet arrivals, but between any pheromone observations, such as when pheromone is used to determine link probability. In essence, pheromone is decaying all of the time and the update algorithm implements this by noting all times at which the pheromone has been observed for any reason.

Piggybacked Routing Information

In order to easily disseminate routing information, each packet contains the cumulative metric measurement experienced as it has travelled through the network. The cumulative metric measurement is simply a number detailing the total delay, throughput, hop count, energy use, or whatever the optimizing metric may be, that the packet has experienced enroute to the current node. On many platforms, this information results in the addition of only eight bytes of overhead.

While it is not new to piggyback routing information on data packets, it should be stressed that this information is piggybacked on *all* packets. Even data packets are used to actively update the network.

Neighbor Broadcast

This is a general technique used is many routing algorithms. Since MANETs exist in a multiuser broadcast environment, all neighbors can overhear a packet transmission. One of those neighbors will be the intended receiver, but all local nodes may eavesdrop and update their own routing information with data from that packet. Packets processed by unintended receivers are subsequently dropped. Neighbor broadcast substantially improves the dissemination of routing information without incurring any additional overhead. This technique is found to be particularly useful in Termite.

5.2.2 Packet Forwarding

As in the previous chapter, in order to forward a packet towards its destination, the forwarding equation is used to determine the next hop neighbor. This formula maps the destination $d \in \mathcal{D}^n$ pheromone on each outgoing link $i \in \mathcal{N}^n$ at node n, $P^n_{i,d}$, to the probability that that link will be used to forward the packet to the destination, $p^n_{i,d}$. The specific next hop neighbor is chosen randomly according to this distribution. Unlike the previous chapter, the previous hop, r, *is not* precluded from this calculation; a packet may be forwarded to the previous hop. The forwarding equation is shown in Equation 5.1 below.

$$p^n_{i,d} = \frac{(P^n_{i,d} + K)^F}{\sum_{j \in \mathcal{N}^n} (P^n_{j,d} + K)^F} \tag{5.1}$$

Source Pheromone Repel

Source Pheromone Repel (SPR) is an adaptation of the packet forwarding process in order to take advantage of additional routing information already available at each node. Packets usually only follow the pheromone gradient of their destination. SPR forces the packets away from their own source's pheromone, thus simultaneously pulling and pushing the packet towards its destination. SPR was originally reported in VWPR [50] and later used in MACO [51] as a means to find a ranking of good solutions. MACO expands on the ant colony metaphor for routing by placing two competing colonies in a network to find routes; each is repelled by the other, thus forcing them to alternative solutions.

SPR adds an extra dimension of directivity to the routing process and elegantly allows for the previous hop to be included in the next hop selection. The preclusion of the previous hop was intended to mitigate obviously bad routing decisions and loops. The addition of SPR is very effective at preventing packets from being routed back to where they came from. While not impossible, it is unlikely. The reason for this is that the arrival link is more likely to be useful to go to the source. The source pheromone on it will be higher than the destination pheromone. This will yield a smaller forwarding probability for the destination based on the SPR forwarding equation, Equation 5.2. If the packet happens to wander through the network and arrive instead on a link that is relatively useful to go to the destination, the amount of source pheromone on that link is likely to be small; source and destination nodes are likely to be on opposite sides of the node and thus best reachable over different links. Ultimately, complexity may be removed from the algorithm while maintaining performance.

Unfortunately it cannot be relied upon blindly. Poor parameter selection can keep packets behind topological bottlenecks because they cannot overcome the repelling force of the source pheromone at that point.

The forwarding equation is augmented in order to determine the new link bias based on source pheromone. Both the destination and source link biases are computed according to the original forwarding equation. The *source repel sensitivity*, $R \geq 0$, controls the amount by which the packet is repelled by the source pheromone. A meta link distribution, $\hat{p}^n_{i,d}$, is then created. This is similar to the equation introduced by Virtual-Wavelength Path Routing.

$$\hat{p}^n_{i,d} = \frac{p^n_{i,d}\left(p^n_{i,s}\right)^{-R}}{\sum_{j \in \mathcal{N}^n} p^n_{j,d}\left(p^n_{j,s}\right)^{-R}} \tag{5.2}$$

Route Discovery and Repair

In case there does not exist any destination pheromone at a node for a packet to follow, a route discovery procedure must be initiated. Previously, some number of route request packets (RREQ) would be sent on a random walk to find a node with the destination's pheromone. A route reply packet (RREP) is then sent back to the requesting node, marking its path with destination pheromone. A path with destination pheromone then exists for the data packet to follow.

The version of Termite used here does *not* use control traffic. If a destination is unknown, its pheromone on each link is zero and thus the packet sees a uniform forwarding distribution over each link (save for the effect of the source pheromone if SPR is used). Routing in a network with no pheromone at all is difficult; packets perform a random walk over the network. Ultimately, only data packets exist in the network; each carries a small amount of routing information with it, which is disseminated to each node on its path as it moves through the network.

There is no concept of route repair in Termite. Each next hop is computed online and every node has an estimate of the utility of each link to deliver a packet to its destination. If a link should fail, the neighbor row is removed from the node's routing table and the next-hop probabilities are recomputed for the remaining set. If all else fails, the packet will simply be routed back to the previous hop, or else dropped.

Loop Prevention

Preventing packets from being routed in loops is always a key priority for a routing algorithm. This behavior is clearly suboptimal. A typical approach to loop prevention is for each node to maintain a list of packets already routed. If the same packet is seen more than once, an error procedure is executed. This might include dropping the packet or sending control traffic into the network to fix routing tables.

Termite as presented here does not make any special effort to prevent loops. All received packets are forwarded as described, regardless of the number of times they have visited any particular node. SPR helps mitigate the number of routing loops by making hops towards the source less likely. Nodes closer to the source (measured according to the network metric) will have larger amounts of source pheromone on them. This makes travel towards a packet's source increasingly unlikely, as travel towards the source is generally also travel away from the destination. Looping is most probable at the midpoint between source and destination where the push and pull of the two is roughly the same. Packets fall into a random walk pattern when there is no significant pheromone gradient in any direction. This indeterminate directionality can lead to packet death as time-to-live (TTL) counters expire.

Evidence of this sort of behavior may be seen in the decreasing goodput of larger networks. These situations allow larger intermediate areas where neither source nor destination pheromone gradient is well defined.

Duplicate Packet Processing

No special effort is made to prevent the same packet from being processed more than once. This was the case in the previous description of Termite. The message identification feature is removed, and the pheromone table is updated any time a packet is received. This is true even when a neighbor retransmits a packet just forwarded by the local node. Because this version of Termite decreases the amount of pheromone on a packet as it moves through the network (the packet's total path utility decreases), the effect on the pheromone table will not be detrimental.

5.3 Pheromone Update Techniques

A total of eleven pheromone update techniques are tested. They represent a variety of approaches and philosophies to the problem of effectively maintaining reliable and current routing information at each node in the network. The classic swarm intelligent approaches are used, along with distance vector style, and link independent and dependant updates.

5.3.1 Classic Pheromone Filter

The classic pheromone filter (CPF) is the ant pheromone decay model used by the ant food foraging example. It is widely used in many routing applications such as Termite, ARA, MACO, as well as ACO. The particular variant used here features continuous decay which takes into account pheromone observation times.

Each arriving packet deposits a constant amount of pheromone, γ_c, on the link that it arrives on, regardless of the quality of the path that it has traversed. Pheromone constantly evaporates over time; pheromone evaporation is modeled as exponential decay where t is the current time, t_s^n is the last time that pheromone from node s was observed at node n. The pheromone decay constant is τ. Node r is the previous hop.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ P_{i,s}^n &\leftarrow P_{i,s}^n \cdot e^{-(t-t_s^n)\tau} \\ P_{r,s}^n &\leftarrow P_{r,s}^n + \gamma_c \end{aligned} \tag{5.3}$$

5.3.2 γ Pheromone Filter

The γ pheromone filter (γPF) is an extension of CPF only in that each packet deposits an amount of pheromone equivalent to the utility of the path that it has taken in order to arrive at the current node. The difference is small but the global effect is substantial. Where the classic pheromone filter makes decisions based on packet throughput, the γ pheromone filter bases them on path quality.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ P^n_{i,s} &\leftarrow P^n_{i,s} \cdot e^{-(t-t^n_s)\tau} \\ P^n_{r,s} &\leftarrow P^n_{r,s} + \gamma \end{aligned} \tag{5.4}$$

As the packet moves through the network, its utility measurement worsens. Basing routing updates on a worsening metric measurement is similar to ABC's method of making routing probability adjustments smaller based on the amount of time the packet has spent in the network; the packet age [45].

5.3.3 Averaging Filter

A one-tap infinite impulse response averaging filter is a basic averaging technique widely used in previous work such as CAF. It may also be viewed as an evolution of the γ pheromone filter. The averaging filter retains the idea of pheromone decay, but normalizes the result in order to produce an actual estimate of the utility of that link to the destination. Link utility is estimated independently for each link for every source-neighbor pair. The time $t^n_{r,s}$ is the last time that the pheromone for node s on the link to neighbor r was observed.

$$P^n_{r,s} \leftarrow P^n_{r,s} \cdot e^{-(t-t^n_{r,s})\tau} + \left[1 - e^{-(t-t^n_{r,s})\tau}\right] \cdot \gamma \tag{5.5}$$

5.3.4 Normalized γ Pheromone Filter

The normalized γ pheromone filter (NγPF or $\bar{\gamma}$PF) is a variation of the previous two methods. Each link is normalized with respect to incoming pheromone, but there is also the effect of joint pheromone decay.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ P^n_{i,s} &\leftarrow P^n_{i,s} \cdot e^{-(t-t^n_s)\tau} \\ P^n_{r,s} &\leftarrow P^n_{r,s} + \left[1 - e^{-(t-t^n_{r,s})\tau}\right] \cdot \gamma \end{aligned} \tag{5.6}$$

5.3.5 Probabilistic Bellman-Ford

With packets traveling the network and updating link metrics based on path utility, this behavior is reminiscent of a distance vector routing approach such as the Bellman-Ford algorithm. The Probabilistic Bellman-Ford (pBF) algorithm is designed as a distributed, asynchronous, probabilistic version of the Bellman-Ford algorithm where packets are routed probabilistically and update routing information at each node they visit. Unlike all other techniques tested here, this one is nonlinear.

$$\begin{aligned} \forall i \in \mathcal{N}^n, \quad & P^n_{i,s} \leftarrow P^n_{i,s} \cdot e^{-(t-t^n_s)\tau} \\ if\ P^n_{r,s} < \gamma, \quad & P^n_{r,s} \leftarrow \gamma \end{aligned} \tag{5.7}$$

5.3.6 Ant-Based Control + X

While it does not represent a fundamental difference, SI algorithms such as ACO feature a different mechanism for balancing exploration and exploitation than Termite does now. Earlier algorithms sent an ant over the best available link, and also had an explicit noise function which forced them to choose a probabilistic next-hop. This is probabilistic determinism. ABC does something similar where control packets are usually routed according to the pheromone table, but occasionally forwarded uniformly. ABC's data packets are always forwarded over the best link. Termite integrates this tradeoff between exploration and exploitation directly into the forwarding equation with the pheromone threshold and sensitivity.

The Ant-Based Control + X (ABC+X) algorithm is a combination of the ABC forwarding method described above and a selected pheromone update method which determines how pheromone is accounted for. That is, pheromone is accounted for normally, according to an accounting method such as γPF. When a packet must be forwarded, it is sent either on the link with the largest amount of pheromone, or else a next-hop is chosen uniformly. The use of the uniform forwarding method is chosen randomly according to a noise factor.

Three different variations of ABC+X are used in this work. They include ABC+γPF, ABC+$\bar{\gamma}$PF, and ABC+pBF.

5.3.7 Box Filter

The box filter is a simple length m averaging filter. If the pheromone filter can be approximated by the averaging filter, the box filter should be able to estimate the average utility of a link as well. Each packet travels independently through the network and this represents an independent stochastic sample of the network environment. The box filter is ideally suited to estimate the mean of a signal in this case [77]. The length of the filter must be adjusted in order to trade off between speed of adaptation and accuracy of the estimate. As with the averaging filter, each source-neighbor pair is filtered separately and independently. Here, $\gamma^n_{r,s}(t)$ is integer indexed as the most recent pheromone arrived at time t at node n on neighbor link r from source s. Equation 5.8 simply averages the last m received pheromone on each link for each source.

$$P^n_{r,s} \leftarrow \frac{1}{m} \sum_{i=0}^{m-1} \gamma^n_{r,s}(t-i) \tag{5.8}$$

5.3.8 Oracle

The Oracle sets the destination pheromone on each link equal to the minimum distance to the destination out of that link. Each node always knows the exact utility of each link. This is not a practical algorithm, rather a test case used to determine how well the system is able to perform with exact information.

5.3.9 Random Routing

Random routing is the baseline case in these experiments. Each link is considered uniformly regardless of observed traffic. Comparisons of the above algorithms to a pure random walk will show how the traffic flow information is exploited to produce reliable routing results.

In order to fit the random walk into the forwarding model, the pheromone on each link is maintained at zero. In this way the forwarding equation (assuming $K > 0$) will return a uniform distribution over the outgoing links, even with SPR. The random routing pheromone update equation is shown in Equation 5.9.

$$\forall i \in \mathcal{N}^n,\ P^n_{i,s} \leftarrow 0 \tag{5.9}$$

5.4 Simulation and Analysis

A number of different scenarios are simulated in order to compare the performance of the various pheromone accounting techniques over a variety of system parameters. Simulations are designed to test the effect of node mobility, pheromone sensitivity (F), and pheromone decay rate (τ) on the global performance metrics.

5.4.1 Environment and Parameters

Two scenarios of varying network diameter are tested. This includes 50 nodes distributed uniformly over an area 100 meters square, and 100 nodes distributed over 141.2 meters square. Network population is increased while maintaining node density. Common to each scenario is a node transmission radius of 10 meters. As the nodes move according to a random waypoint mobility model, a single source and destination pair transmit packets to each other at a rate of two packets per second. Packets are only sent when a path exists between source and destination. This is determined by the simulation. All tests are performed with single communication pairs since pheromone is superimposable. Performance is tested with nodes moving at 1, 2, 5, and 10 meters per second, presenting a wide range of node mobility. It is unknown what values of the pheromone sensitivity and pheromone decay rate give the best performance. The sensitivity is tested with six values, 1, 2, 5, 10, 20, and 50, and the decay rate is tested with 0.1, 0.2, 0.5, 1, 2, and 5 seconds^{-1}. Each combination of mobility, sensitivity, and decay rate is tested for 5000 seconds and averaged over five runs. This gives a total of 720 simulation runs to test each pheromone update method. The optimization metric is based on a simple energy model, where the cost of a link is equal to the distance between nodes squared. In all scenarios, the pheromone threshold is chosen as the utility of a hypothetical link spanning the diagonal of the simulation area; $K_{50} = 5 \times 10^{-5}$, and $K_{100} = 2.51 \times 10^{-5}$. All packets have a time-to-live (TTL) of 32 hops. The ABC+X algorithms have a 5% exploration probability.

The purpose of these simulations is to test the pheromone update methods. A perfect MAC layer is assumed; nodes are able to communicate over an error and contention free medium.

5.4.2 Evaluation Metrics

A number of metrics are used to determine the utility of the proposed algorithms. These include data goodput, path ineffiency, delivery efficiency. The link change rate is a measure of the network volatility, or rate of change of the network topology.

Data Goodput

Data goodput is a classic evaluation metric for routing algorithms. It is the fraction of successfully delivered data packets. This metric should remain as high possible under any circumstances.

Path Inefficiency

This metric is a measure of how many times larger the achieved per packet path metric is, as compared to the smallest available path metric at the time. The lower this number, the better; the path length ratio must be greater than or equal to one, since the achieved performance is compared to the optimal.

Delivery Efficiency

This metric aims to characterize the value of the algorithm in one number. It is the ratio of the goodput to path inefficiency. It is a number between zero and one, and values approaching unity are desired.

Link Change Rate

The link change rate measures the average number of links that change state, new or leaving, every second at each node. It is a relative measure of how fast the network topology is changing, and thus the time available to acquire new information about the network. This metric is shown in Figure 5.1 for each tested speed and node count. Note that the 100 node case sees a larger slope. This is due to fewer boundary effects resulting from a larger simulation area.

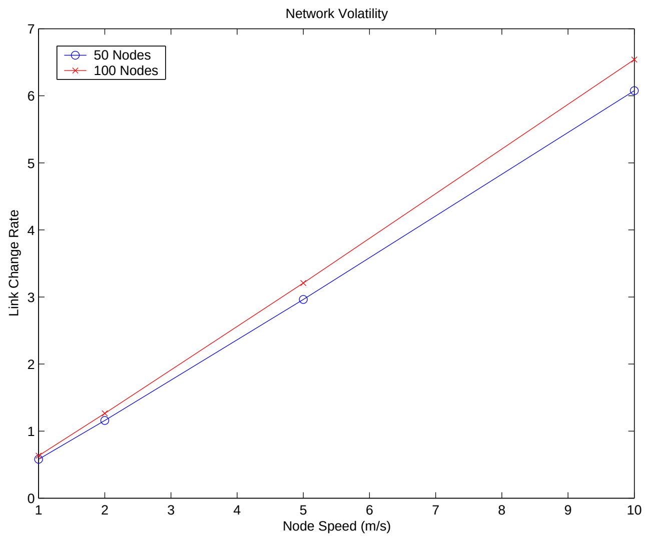

Figure 5.1: Link Change Rate for 50 and 100 Node Simulations

5.4.3 Simulation with 50 Nodes

In this simulation series, fifty nodes are placed in a square area of 100 meters on a side. Pheromone sensitivity, pheromone decay rate, and node speed are the independent parameters. The results here only compare with node speed for reasons of clarity. The values of F and τ giving the best results are used. The complete results are shown in Appendix A.

Data Goodput

The results for data goodput are shown in Figure 5.2. The Oracle is able to do extremely well, but only with a very large pheromone sensitivity (Figure A.7). Note that it does not actually achieve full goodput; a finite sensitivity and non-zero pheromone threshold allow enough randomness in the forwarding process as to let packets wander. The Oracle is also able to achieve constant performance regardless of node speed, but this is expected since it always has perfect information. The γ pheromone filter is the best performing realizable implementation. At low speeds it is able to achieve near perfect goodput, while suffering only a small performance loss at high speeds and still achieving over 96% packet delivery. $\bar{\gamma}$PF and pBF are close contenders in the low node speed regime, but lose substantially more performance as speeds increase. $\bar{\gamma}$PF achieves roughly 95% goodput at 10 m/s while pBF drops to 91%. The box, averaging, and classic pheromone filters fall so far behind the primary three that the results only serve to show that these methods are unsuitable. More interesting are the results for the ABC+X class of algorithms. The results are comparable with those of the pheromone filters, but ABC+X generally lags behind the performance of the original algorithm. Not shown on the graph for reasons of clarity are the results for random routing, which is a constant 36% at all speeds.

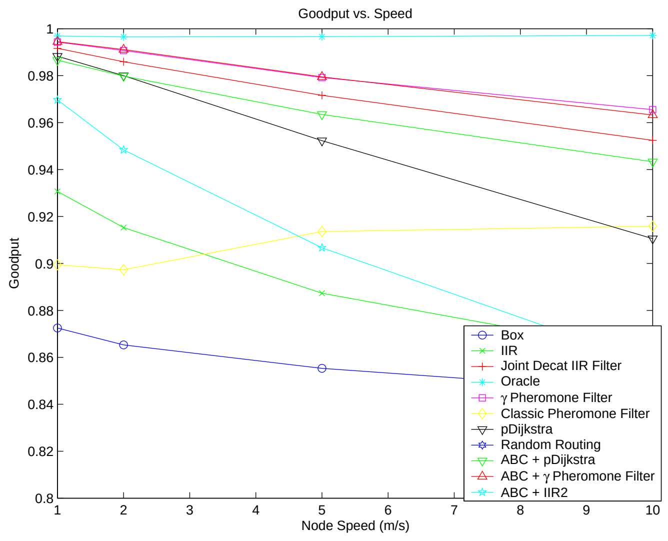

Figure 5.2: Data Goodput vs. Node Speed (50 nodes)

Perhaps the most interesting feature of these results is that the classic filter does *better* at high node speeds. The reason for this is the high constant value of pheromone that was used in these simulations compared to the pheromone threshold; $\gamma_c = 1$. At low speeds, when it is easy to find paths, such a large amount of pheromone on many paths creates an ambiguous gradient to the destination. Packets are not well directed. However, at high speeds it is much more difficult to find a stable path. With such a large amount of pheromone being laid down over short-lived paths, strong gradients are created and packets can more easily find their way through the network. The effect of using such a large pheromone constant was unknown before these experiments. The constant was chosen based on previous work.

Path Inefficiency

The path inefficiency results in Figure 5.3 are somewhat more unexpected. γPF performs quite poorly compared to the other tests. pBF is the best contender of the three pheromone based techniques and performs equally well as the box and averaging approaches. $\bar{\gamma}$PF uses less optimal paths than pBF. ABC+X performs comparably to γPF; poorly.

The fact that the link independant estimators, the box and averaging filters, perform so well in this metric is somewhat misleading. The metric can only be measured with packets that have successfully arrived at their destination. Figure 5.2 shows that these pheromone measures have very poor goodput, and thus are likely to successfully deliver packets only when an easily discovered path is available. The shorter the path, the easier to find. The box and averaging filters achieve a very good path inefficiency but overall their performance is still rather poor.

By constrast, the classic pheromone filter achieves a low goodput and is also path inefficient. Since this filter can establish a strong pheromone gradient quickly using a large pheromone constant, even long paths can be followed. And because longer paths are likely to be established first by a wandering packet, this filter will continue to use that path until it breaks.

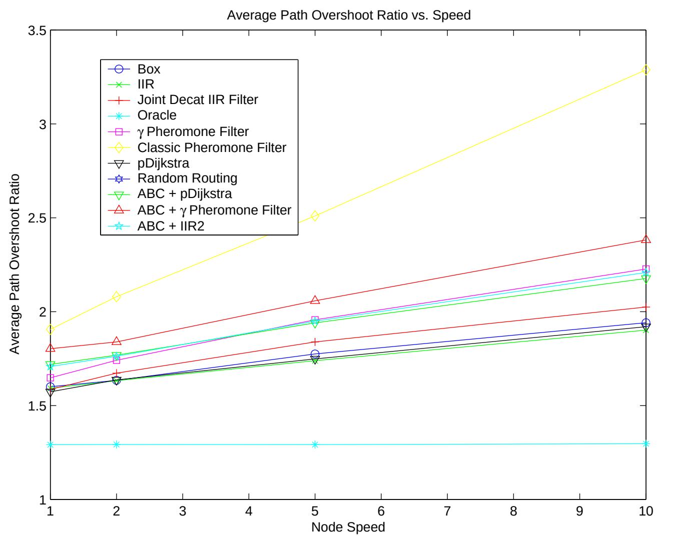

Figure 5.3: Path Ineffiency vs. Node Speed (50 nodes)

Delivery Efficiency

The delivery efficiency metric gives an alternative view of the results, shown in Figure 5.4. $\bar{\gamma}$PF is now the top performer regardless of speed. pBF and γPF follow closely, however the former is superior in lower speed regimes while the latter in higher. ABC+X has comparable performance. As seen in the goodput results, the box, averaging, and classic pheromone filters perform dismally.

5.4.4 Simulation with 100 Nodes

In order to test the algorithm in larger diameter networks, a simulation environment of 100 nodes is generated with the area increased to 141.2 meters squared in order to keep the node density constant. All other simulation parameters are retained, except for the pheromone threshold which is changed to reflect the new

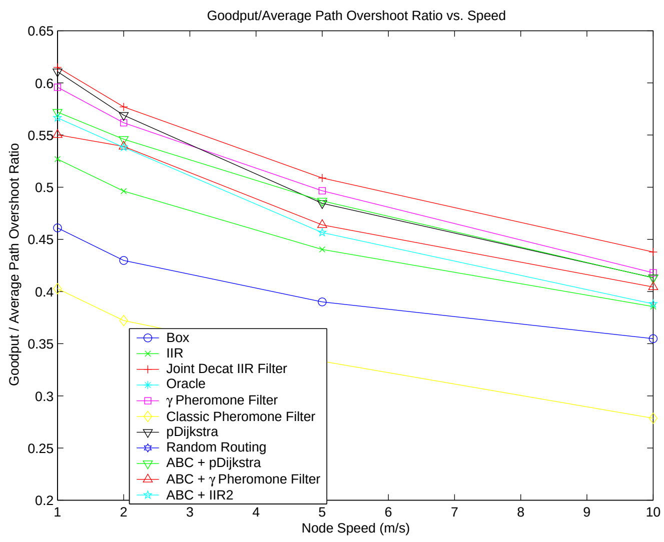

Figure 5.4: Delivery Efficiency vs. Node Speed (50 nodes)

area. This environment tests how well the algorithm is able to scale with network size; the diameter of the network is roughly doubled. Only the best performing algorithms are retained for clarity of presentation and timeliness of simulation, including the γ pheromone filter, the normalized γ pheromone filter, and the probabilistic Bellman-Ford. Oracle and Random Routing are kept for purposes of comparison. The results of the 100 node simulations mirror those of the smaller test. The absolute results drop, however the ranking remains the same.

Data Goodput

γPF is able to perform the best across all tested speeds, followed closely by $\bar{\gamma}$PF and pBF in the low speed regime, and becoming clearly superior in high speeds. All methods have roughly 96% goodput at low speeds, however large speeds differentiate the results significantly with ranging from 87% to 85% to 78% for γPF, $\bar{\gamma}$PF, and pBF. The Oracle is able to do very well at 98% regardless of node speed; random routing achieves only a contant 19%. The results are shown in Figure 5.5.

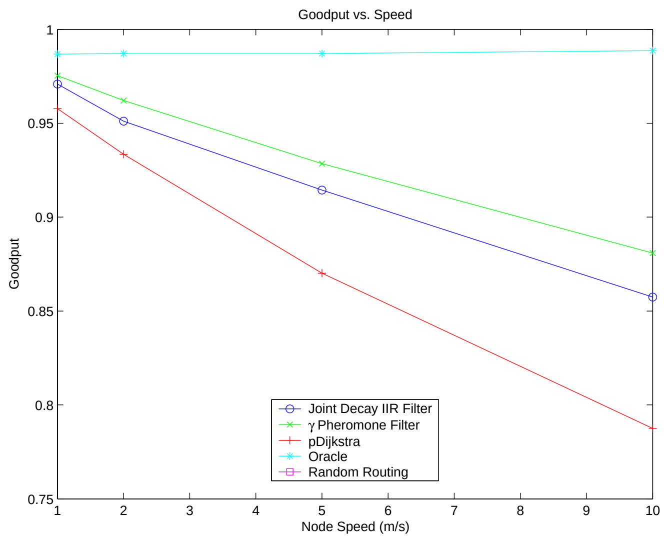

Figure 5.5: Data Goodput vs. Node Speed (100 nodes)

Path Inefficiency

The path inefficiency ranking in Figure 5.6 remains the same as in the 50 node case. The algorithms are able to maintain the ratio of achieved path length to the minimum available path length independent of network size. This is a very fortunate property from the perspective of scalability. γPF, $\bar{\gamma}$PF, and pBF each manage a ratio of about 1.75 at low node speed and spread to a range between 2.1 and 2.5 at high speed between $\bar{\gamma}$PF and γPF. Oracle achieves a ratio of approximately 1.3 and random routing performs very poorly at 15.

Delivery Efficiency

The delivery efficiency results of the 100 node simulations reflect those of the 50 node simulations. $\bar{\gamma}$PF performs the best at all speeds, followed by γPF and pBF. The only surprise here is that pBF outperforms $\bar{\gamma}$PF slightly at very low node

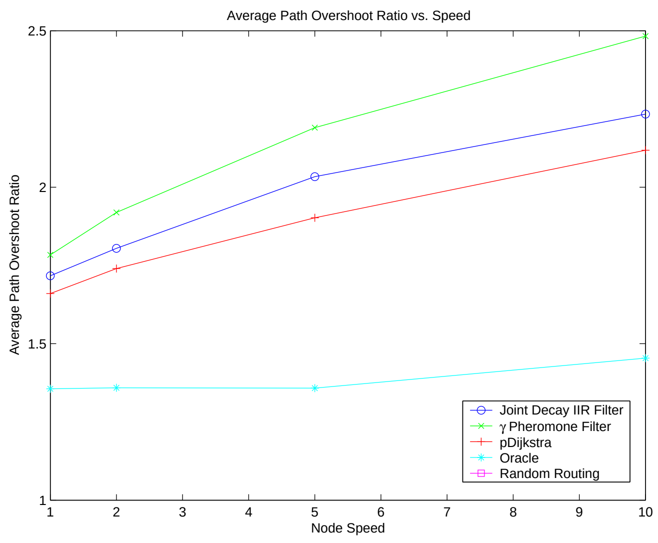

Figure 5.6: Path Ineffiency vs. Node Speed (100 nodes)

speeds. Oracle achieves 72% delivery efficiency and Random Routing manages only 1.3% efficiency. These results are shown in Figure 5.7

5.4.5 Parameter Analysis of F and τ

Both the pheromone sensitivity, F, and pheromone decay rate, τ, are varied in order to examine which values give the best performance with respect to node speed. Results for the optimal pheromone decay rate, τ^*, are shown in Figure 5.8. The different values of the pheromone decay rate tested are 0.1, 0.2, 0.5, 1, 2, and 5 seconds^{-1}. As expected, the decay rate achieving the best performance, with respect to delivery efficiency, increases with node speed. Note that the decay rate does not need to be too high even with high node movement. The figures of Appendix A show that the effects of the pheromone decay rate and the pheromone sensitivity can offset each other to some degree. Global performance does not necessarily change dramatically as decay rate increases. Should the decay rate be too aggressive, the pheromone sensitivity can be raised in order to accentuate the

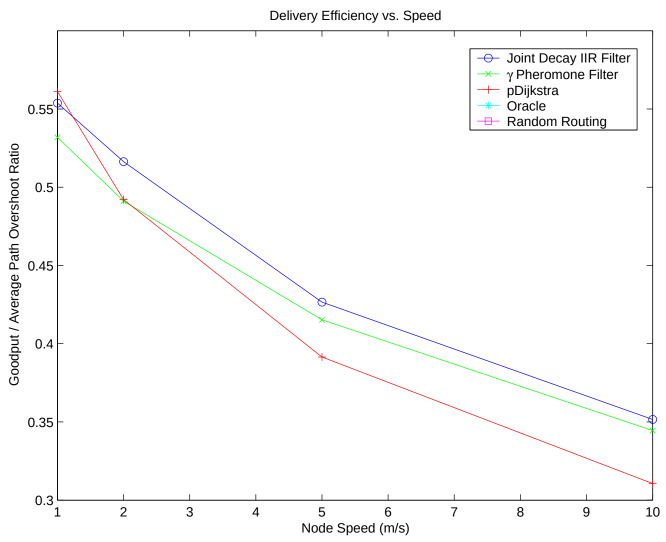

Figure 5.7: Delivery Efficiency vs. Node Speed (100 nodes)

remaining differences in pheromone and force a high probability trail.

The different values of F that are tested are 1, 2, 5, 10, 20, and 50. The results are in Figure 5.9. It is difficult to draw any serious conclusions from this graph without also looking at the complete data set in Appendix A. The data shows that the optimal pheromone sensitivity remains constant with node speed. The truth of the matter is that this parameter makes little to no difference in global performance above a certain level, shown to be $F = 10$ in these simulations. This threshold exists because the relative differences between resident pheromone values have been exaggerated to an extreme above this value. Exaggerating them more does not make any practical difference in the link probabilities.

The results are more affected at lower levels of the sensitivity when the decay rate is also varied. For instance, Figure A.1 shows how the decay rate can have a substantial effect on the delivery efficiency when the γ pheromone filter is used. In contast, Figure A.2 shows how the performance of the normalized γ pheromone filter is nearly independent of the sensitivity above this threshold. The reason for this is that the each of the tested decay rate values are much too large in the normalized case; the unnormalized filter requires larger optimal decay rates. τ does not seem to make a difference for normalized filters because even $\tau = 0.1$ already decays pheromone much too quickly based on the amount of pheromone already on the link. The exact mechanics for this are discussed in detail in the following chapters.

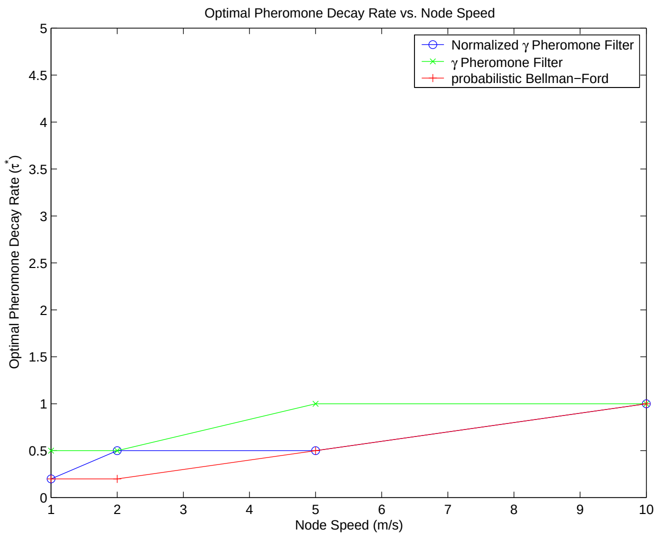

Figure 5.8: τ^* vs. Node Speed (50 Nodes)

5.4.6 Analysis

While the random routing approach fares particularly poorly, it serves as an excellent standard against which the other algorithms can be compared. In both the 50 and 100 nodes simulations, all pheromone update methods are able to perform much better than random choice. This is not to say that the methods perform near optimally; this case has also been disproven by comparison to the oracle. But the results are certainly much better than they are worse.

The γ pheromone filter is able to perform extremely well with regards to the

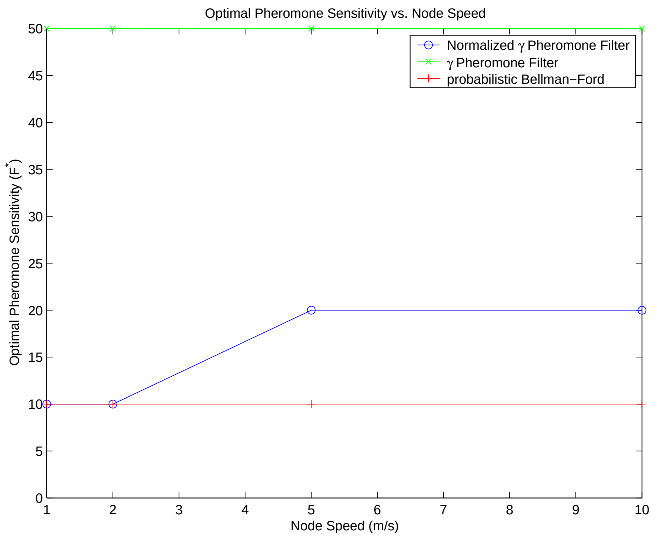

Figure 5.9: F^* vs. Node Speed (50 Nodes)

goodput metric. Even at 10 m/s, where a node can cross another's transmission range in two seconds, and with six links changing state per second per node on average, the algorithm is able to deliver over 96% of the data packets successfully. This level of performance is likely due to the fact that the γ pheromone filter maintains a relatively large amount of pheromone on its links. A strong pheromone gradient is created between source and destination for packets to follow. Unfortunately, the large pheromone gradient is also slow to decay, which limits the rate at which the algorithm can adapt to changes in the network environment. Thus, the γ pheromone filter shows poor path efficiency and ultimately suffers in the delivery efficiency metric.

For similar reasons, the normalized γ pheromone filter shows superior performance in the latter metric to the unnormalized version. The former maintains less pheromone and thus a weaker gradient. Goodput suffers, but path flexibility grows stronger. The probabilistic Bellman-Ford algorithm also maintains less pheromone than the γ pheromone filter.

The link independent estimators are all but worthless. While it may not be a bad idea to estimate the utility of each link independently, the implementation is certainly flawed. By adding joint link pheromone decay to the averaging filter and creating NγPF, performance is improved dramatically. This implies that a method for measuring the freshness of the available information about the network consistently across links is critical. The pheromone techniques use pheromone decay in order to achieve this. The algorithm supposes that information about the network loses relevance exponentially with time. The linear filter techniques only update their estimates upon packet arrival. The box filter has no way of incorporating time, such as the packet interarrival time, into its estimate. A dynamic determination of filter length is necessary to properly account for time-based correlation.

It is important to remember that the absolute numbers shown in the results may not be accurate under more realistic situations. It is the ranking of pheromone update methods that is important. The qualitative results provide intuition as to the true operation of such swarm intelligent or stochastic algorithms. There is a great deal to be learned from the type of updates that work well and why.

5.5 Conclusion

A routing algorithm for mobile wireless ad-hoc networks has been presented. The principles of swarm intelligence are used to build an emergent routing behavior. Packets probabilistically follow pheromone trails to their destination while laying source pheromone. Passive route marking reduces the need for explicit routing traffic, thereby maximizing the network resources available to carry data traffic. Nodes determine network conditions by monitoring traffic flow and make adjustments to their routing tables.

Various pheromone accounting methods are tested by simulation. A pheromone decayed version of a simple averaging filter is shown to be superior to the traditional pheromone metaphor approach. The former maintains less pheromone on each link due to its normalization process. This allows it to adapt faster than the latter filter, although it does not achieve as high a goodput. Performance of each accounting method scales with network size although there is a general negative trend. The improved performance is gained by moving away from the biological perspective and borrowing ideas from traditional linear filtering. The pheromone sensitivity and decay rate also have an effect on performance. There seems to be a sensitivity threshold above which performance is unchanged, while the pheromone decay rate should be more carefully determined. Their relative effects can offset each other to some extent as well.

Results are generally very good, delivering the large majority of packets to their destination without any control traffic under very high mobility conditions. The implementation is very simple, letting the routing behavior emerge from the simple interaction of many randomized agents.

CHAPTER 6

ANALYSIS OF PHEROMONE UPDATE IN SI MANETS

6.1 Introduction

The previous chapter has shown that modified pheromone update schemes can improve routing performance over the traditional approach. It remains unclear as to exactly why the new schemes work as well as they do, and how parameters affect their performance in general. This chapter presents a simple analytical model of Termite [79]. The purpose of this model is to discover how individual parameters are related to each other and how they affect global metrics, such as the reliability of message delivery and adaptability to changes in the network environment. The critical element under study is pheromone. Because Termite is based on a model of social insect behavior, much of the biological terminology remains. Pheromone is a measure of the metric that the network is optimizing for; it is a measure of route utility.

The model will first be used to characterize the behavior of pheromone on a single communications link. This will establish an intuition for determining the dynamics of pheromone in a system of two links. An understanding of pheromone behavior is then used suggest two heuristics on the optimal selection of the pheromone decay rate. This parameter must be carefully chosen such that information about the network is retained only as long as it needs to be, without disregarding it too quickly. The heuristics are known as Maximum Pheromone and Filter Cutoff. Their predictions of the pheromone decay rate are then compared to simulation results from the previous chapter.

Swarm intelligent routing algorithms lend themselves to mathematical analysis.

Their routing update and decision procedures are mathematical functions themselves. One of the earliest works on swarm intelligent routing is the Ant Based Control (ABC) algorithm. ABC has been modeled analytically in [46]. This work demonstrates and gives analytical justification for the behavior of pheromone and its effect on global system performance in ABC.

A great deal of work has been done with the original biologically inspired models. This work spans several fields, including experimental biology, theoretical biology, and the various disciplines of engineering which apply the models derived by the former. Some summaries may be found in [1] and [7].

6.2 The Model

A model of an ad-hoc network is presented which will be used to evaluate the behavior of the pheromone update methods. The network is modeled as two communicating nodes with two independent paths available between them. These paths abstract all other connections between the two nodes, including additional nodes, mobility issues, or communications effects. The physical structure is shown in Figure 6.1, and is the same as that used in [46].

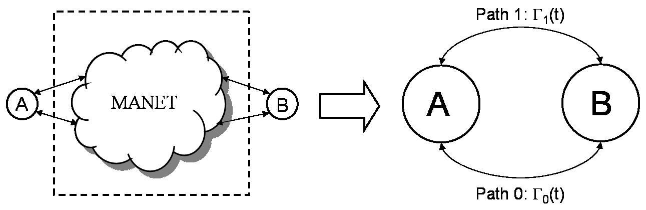

Figure 6.1: Diagram of the MANET Model

Each node sends packets to the other with independent exponentially distributed interarrival times. The average rate at which node A sends packets to B is λ_A, and λ_B in the opposite direction. Each node is also able to decay the pheromone on its links independently. The decay rates at each node are τ_A and τ_B, respectively.

Each path, indexed by v, has a utility characterized by a non-negative random process $\Gamma_v(t)$ with mean $\mu_v(t)$. The pheromone contained in a packet arriving on a link, γ, is a sample of that process. Γ is non-stationary since link utilities change over time due to mobility and other effects. Γ is considered to be stationary in this work for ease of analysis. Since each packet moving in the same direction passes through the network independently of all other packets, there is no correlation between successive samples of the link utility process. The forwarding equation independently considers each packet.

6.3 Pheromone Update Analysis

This section analyzes the amount of pheromone found on a link. The results will explain the performance of the γ pheromone filter (γPF), the normalized γ pheromone filter (NγPF or $\bar{\gamma}$PF), and the probabilistic Bellman-Ford (pBF) algorithms, which are reprinted in Figures 6.2 and 6.3 from the previous chapter. Specific answers to be answered include, why the γ pheromone filter achieves such high goodput while being slow to adapt at high mobility, why pBF adapts better, and why the normalized γ pheromone filter is able to achieve the highest performance with regards to the delivery efficiency metric.

Goodput is the fraction of successfully delivered data packets, while the latter metric is the goodput divided by the ratio of the achieved path metric to the best possible available at the time.

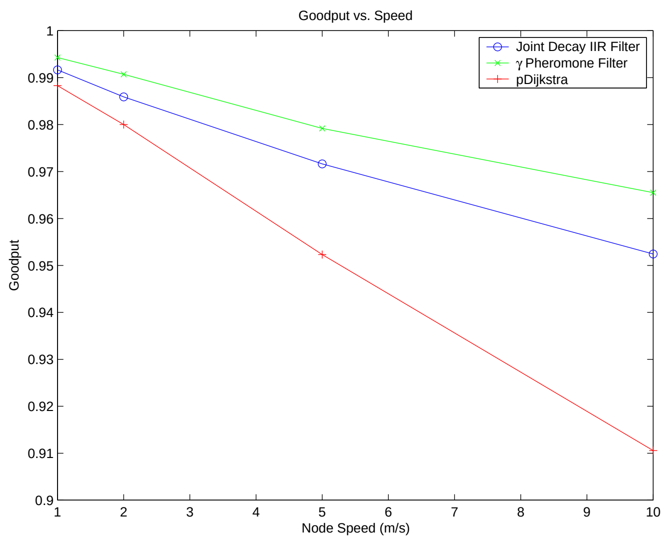

Figure 6.2: Goodput vs. Pheromone Update Method

6.3.1 Single Link Pheromone

A system of one link is first considered in order to give insight on the more interesting system of two links. A formula is established for the time average pheromone deposited on a link, given that the number of packets arriving per second is poisson distributed with mean λ [packets/second]. The decay rate, τ [1/second], remains constant. The poisson packet arrival rate assumption implies that the packet interarrival times are exponentially distributed with mean, λ^{-1} [seconds/packet]. The average value of the received pheromone is $E\Gamma_v = \mu_v$. The amount of pheromone on the link before packets begin arriving is P_0.

By applying the pheromone update equation n consecutive times, an expression is derived for the amount of pheromone on a link given that n packets have arrived, $P(n)$. The packet interarrival time of the nth packet is independently and identically distributed, t_n.

$$P(n) \quad = \quad (((P_0 e^{-t_1\tau} + \mu) \cdot e^{-t_2\tau} + \mu) \cdot \ldots) \cdot e^{-t_n\tau} + \mu$$

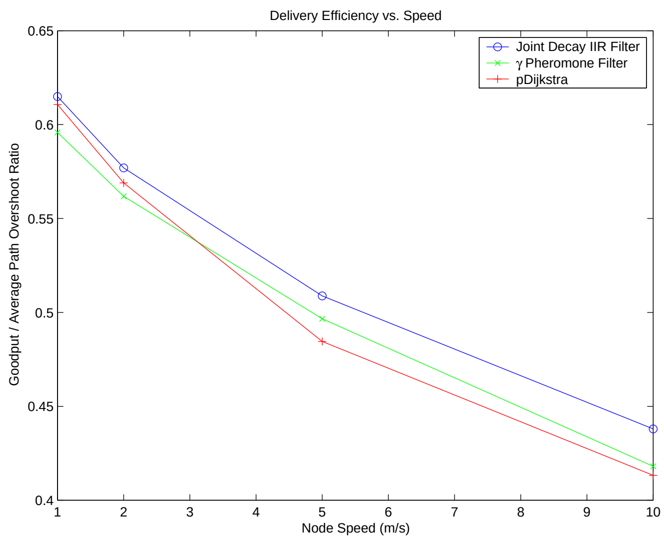

Figure 6.3: Delivery Efficiency vs. Pheromone Update Method

$$\begin{aligned} &= P_0 e^{-(t_1+\ldots+t_n)\tau} + \mu e^{-(t_2+\ldots+t_n)\tau} + \ldots + \mu e^{-t_n\tau} + \mu \\ &= P_0 e^{-\left(\sum_{i=1}^{n} t_i\right)\tau} + \mu \left[\sum_{i=2}^{n} e^{-\left(\sum_{j=i}^{n} t_j\right)\tau} \right] \end{aligned} \tag{6.1}$$

The expectation of $P(n)$ with respect to packet interarrival time is found according to standard methods.

$$EP(n) = \frac{P_0 \lambda^n}{(\lambda+\tau)^n} + \sum_{i=0}^{n-1} \frac{\mu \lambda^i}{(\lambda+\tau)^i} \tag{6.2}$$

In order to simplify Equation 6.2, substitute $\beta = \frac{\lambda}{\lambda+\tau} = \left(1 + \frac{\tau}{\lambda}\right)^{-1}$, and further reduce the expression.

$$EP(n) \quad = \quad P_0 \beta^n + \mu \left(\frac{1-\beta^n}{1-\beta} \right) \tag{6.3}$$

To arrive at an expression for the expected amount of pheromone on a link over time, note that the number of packet arrivals, n, within a given time, t, is distributed according to the poisson distribution with parameter, λt.

$$EP(t) \quad = \quad \sum_{n=0}^{\infty} \left[poisson(\lambda t, n) \right] \left[EP(n) \right]$$

$$\begin{aligned} &= \sum_{n=0}^{\infty} \left[e^{-\lambda t} \frac{(\lambda t)^n}{n!} \right] \left[P_0 \beta^n + \mu \left(\frac{1-\beta^n}{1-\beta} \right) \right] \\ &= \frac{\mu}{1-\beta} + e^{-\frac{\lambda \tau t}{\lambda+\tau}} \left(P_0 - \frac{\mu}{1-\beta} \right) \end{aligned} \tag{6.4}$$

The long term behavior of the link pheromone is defined as its mean.

$$\begin{aligned} \lim_{t\to\infty} EP(t) &= \frac{\mu}{1-\beta} \\ &= \frac{\mu(\lambda+\tau)}{\tau} \\ &\overset{\text{def}}{=} EP \end{aligned} \tag{6.5}$$

A similar analysis shows the variance of the link pheromone,

$$VAR(P) = \frac{\mu^2 \lambda}{2\tau} \tag{6.6}$$

Scale Invariance The ratio of λ and τ is a scale invariant parameter in this system. Primarily characterized by the expected decay factor in Equation 6.2, the expected pheromone on a link may be held constant as long as $\frac{\tau}{\lambda}$ remains the same. The scale invariant parameter is proportional to β.

The Pheromone Filter

The expected link pheromone may be represented by the sum shown by Equation 6.7, which is a reexpression of Equation 6.2. Indices represent packet arrivals; $\gamma(n)$ is the amount of pheromone contained in the nth packet, and $P(n)$ is the expected pheromone after its arrival. Assume $P_0 = 0$ and $B(n) = \beta^n = \left(\frac{\lambda}{\lambda+\tau}\right)^n$, $n \geq 0$.

$$\begin{aligned} P(n) &= P_0\beta^n + \sum_{i=0}^{n-1} \gamma(n-i)\beta^i \qquad (6.7) \\ &= \gamma(n) * B(n) \end{aligned}$$

The sum is equivalent to the convolution of the incoming sequence of pheromone values and the expected decay factors. The decay process may thus be interpreted as a linear filter with random weights, and the received pheromone as a signal which is passed through the filter. The resulting pheromone on the link is the output of the pheromone filter, B.

Following standard techniques, the complex fourier transform of the filter is shown,

$$B(\omega) = \frac{1}{1 - \beta e^{-j\omega}} \tag{6.8}$$

as well as its magnitude,

$$|B(\omega)| = \frac{1}{\sqrt{1 + \beta^2 - 2\beta\cos(\omega)}} \tag{6.9}$$

The mean, or DC component, of an input signal has frequency $\omega = 0$. It is noted that the peak of this filter is at the mean, and is equal to $\frac{1}{1-\beta}$. If the mean of the received pheromone process is μ, then the mean of the pheromone on the link will be equal to $\frac{\mu}{1-\beta}$. This is a rederivation of Equation 6.5. This analysis assumes a normalized frequency, $-\frac{2\pi}{\lambda} \leq \omega \leq \frac{2\pi}{\lambda}$. The sampling rate of this system is λ samples per second, which is the packet arrival rate.

The cutoff frequency of the pheromone filter is controlled by adjusting the pheromone decay rate. An optimal pheromone decay rate can be computed by selecting to remove a certain portion of the frequency content from the spectral power density of the path utility stochastic process, Γ. The bandwidth of the pheromone filter is defined in the usual way, and $0 \leq s \leq 1$ is the fraction of the peak value at the cutoff. For example, a -3dB cutoff implies $s = \frac{1}{2}$. The cutoff frequency is calculated,

$$\lambda_c = \frac{\lambda}{2\pi}\cos^{-1}\left(\frac{\lambda^2 + \lambda\tau - \left(\frac{1-s^2}{2s^2}\right)\tau^2}{\lambda^2 + \lambda\tau}\right) \tag{6.10}$$

as well as the value of τ corresponding to that cutoff frequency.

$$\tau = \left[\frac{\lambda(1-s^2)}{1 - s^2 \cos\left(\frac{2\pi\lambda_c}{\lambda}\right) + \sqrt{\left(1 - s^2 \cos\left(\frac{2\pi\lambda_c}{\lambda}\right)\right)^2 - (1-s^2)^2}} \right] - \lambda \tag{6.11}$$

The physical interpretation of the cutoff frequency is twofold. The first is that the cutoff frequency represents the maximum frequency of events to which a node will adapt. If changes in the network occur faster than the cutoff frequency, a node will be unable to accurately adapt to this change. Network changes are reflected by the value of received pheromone. Alternatively, unwanted noise in the system or unreliable variability in the arrived pheromone stochastic process will be reduced.

6.3.2 Two Link Pheromone

The following analysis shows the average value of pheromone on each link in a two link system. The γ pheromone filter, the normalized γ pheromone filter, and probabilistic Bellman-Ford update methods are reviewed. Each generates different pheromone dynamics and maintains varying amounts of link pheromone in equilibrium. Each method falls into a one-zero pheromone distribution which echoes previous results from an analysis of ABC [46].

A system of equations is presented to recursively compute the mean pheromone at each node on each link, $P^A_{0,B}$, $P^A_{1,B}$, $P^B_{0,A}$, and $P^B_{1,A}$. Pheromone changes when it is checked in order to send a packet, or when it is updated due to packet arrival. The total pheromone observation rate is $\lambda = \lambda_A + \lambda_B$. The *Pheromone Check* procedure only decays the pheromone and accounts for the fraction of the instances when a packet must be sent. During *Packet Arrival*, which accounts for the fraction of instances that a packet arrives, the pheromone is not only decayed, but also incremented if the packet arrives on the correct link.

The average amount of inter-observation pheromone decay has already been implicitly derived in Equation 6.2. Suppose random variable Y is the interarrival time between packets and is distributed exponentially with mean λ^{-1}, as described in the model definition. Random variable X is defined such that $X = e^{-Y\tau}$, which describes the fraction of pheromone decayed in between packet arrivals. Its probability distribution function is, $f_X(x) = \frac{\lambda}{\tau} x^{\left(\frac{\lambda}{\tau}-1\right)}$ where $0 \leq x \leq 1$. $EX = \frac{\lambda}{\lambda+\tau} = \beta$.

γ **Pheromone Filter**

The γ pheromone filter is shown in Equation 6.12.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ P^n_{i,s} &\leftarrow P^n_{i,s} \cdot e^{-(t-t^n_s)\tau} \\ P^n_{r,s} &\leftarrow P^n_{r,s} + \gamma \end{aligned} \tag{6.12}$$

The example below develops the average link pheromone equation for $P^A_{0,B}$ based on the previous description of pheromone influences. The expected pheromone for the remaining links is then shown as well.

$$\begin{aligned} P^A_{0,B} &= \mathit{Pheromone\ Check} \\ &+ \mathit{Update\ Not\ On\ This\ Link} \\ &+ \mathit{Update\ On\ This\ Link} \end{aligned} \tag{6.13}$$

$$\begin{aligned} P^A_{0,B} &= \frac{\lambda_A}{\lambda}\left[\left(\frac{\lambda}{\lambda+\tau_A}\right)P^A_{0,B}\right] \\ &+ \frac{\lambda_B}{\lambda}\left\{p^B_{1,A}\left[\left(\frac{\lambda}{\lambda+\tau_A}\right)P^A_{0,B}\right]\right\} \\ &+ \frac{\lambda_B}{\lambda}\left\{p^B_{0,A}\left[\left(\frac{\lambda}{\lambda+\tau_A}\right)P^A_{0,B} + \mu_0\right]\right\} \\ &= \left(\frac{\lambda}{\lambda+\tau_A}\right)P^A_{0,B} + \left(\frac{\lambda_B}{\lambda}\right)p^B_{0,A}\mu_0 \end{aligned} \tag{6.14}$$

$$\begin{aligned}
P^A_{1,B} &= \left(\frac{\lambda}{\lambda+\tau_A}\right) P^A_{1,B} + \left(\frac{\lambda_B}{\lambda}\right) p^B_{1,A}\mu_1 \\
P^B_{0,A} &= \left(\frac{\lambda}{\lambda+\tau_B}\right) P^B_{0,A} + \left(\frac{\lambda_A}{\lambda}\right) p^A_{0,B}\mu_0 \\
P^B_{1,A} &= \left(\frac{\lambda}{\lambda+\tau_B}\right) P^B_{1,A} + \left(\frac{\lambda_A}{\lambda}\right) p^A_{1,B}\mu_1
\end{aligned} \qquad (6.15)$$

The link probability described by the forwarding equation is $p^B_{0,A}$. The evolution of pheromone on a link as packets arrive is illustrated in Figure 6.4. The parameters for this particular run include $K = 0$, $F = 5$, $R = 0.5$, $\lambda_A = \lambda_B = 2$, $\tau = 1$, $\mu_0 = 0.99$, and $\mu_0 = 1.0$.

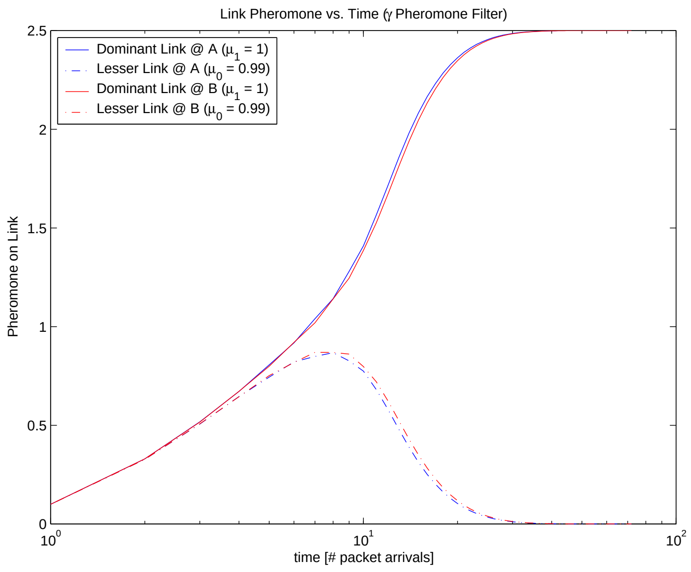

Figure 6.4: Link Pheromone vs. Packets Received using the γ Pheromone Filter

Note that when $K = 0$, the asymptotic probability mass function between the two links follows a one-zero distribution; the algorithm uses the better link exclusively. The reason for this is that when a packet arrives at a node, its link is positively reinforced, while all other links at that node are negatively reinforced; pheromone decays on all links but is only replaced on one. Because packets are also biased towards pheromone, this produces a strong positive feedback which eventually transfers all traffic to the link with the highest utility. With no pheromone threshold there is no incentive to use a lesser link. Consequently the probability of using it disappears. Under these conditions, the dominant link follows the behavior of the single link. This is why ABC and AntNet must include a non-zero noise or exploration parameter, and why Termite must include $K > 0$. Only when $\mu_0 = \mu_1$ do the links have equal probability. An example of this behavior is shown in Figures 6.5 and 6.6, where $\mu_1 = 1$ and μ_0 and K are varied. The former describes the pheromone on each link, and the latter shows the corresponding probability.

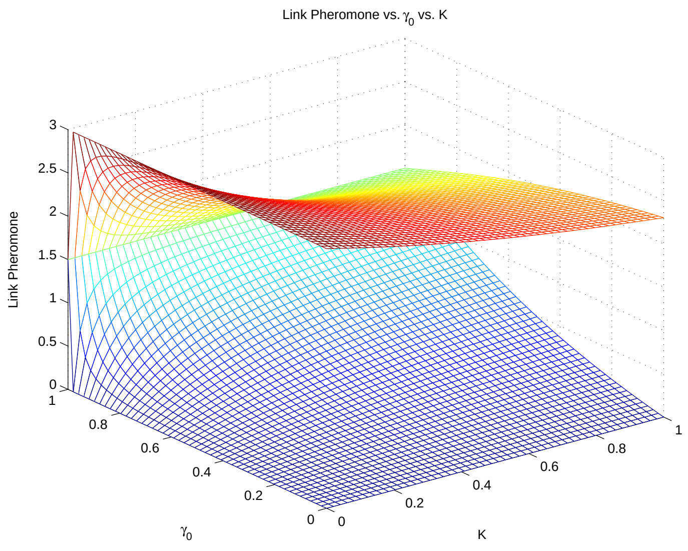

Figure 6.5: Link Pheromone vs. μ_0 vs. K

With $K > 0$, traffic is allowed to be forwarded over all links, regardless of their utility. Currently bad links may be tested on occasion for a change in utility. This allows for a measure of adaptivity, however K must be set appropriately in order to allow for links to be tested *often enough*. Too few tests of other links will not overcome the positive feedback towards to the dominant link. In this way, the function of K is equivalent to that of noise in ABC or AntNet. It is also noted that value of the pheromone on the link and the corresponding probability does

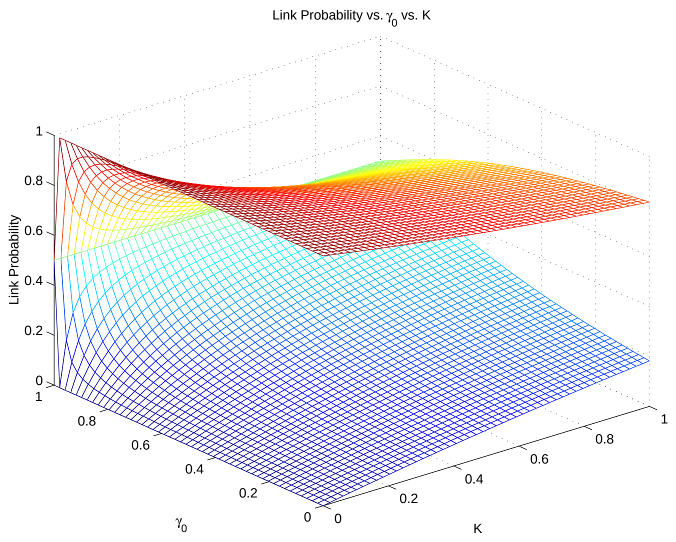

Figure 6.6: Link Probability vs. μ_0 vs. K

not have a physical meaning. The amount of pheromone or the link probability can be set arbitrarily by manipulating parameters such as τ, K, or F.

The mean link pheromone on the dominant link in the case of a one-zero pheromone distribution can be found by setting the probability of using the dominant link to one and solving for the remaining pheromone. For example,

$$\tilde{P}^{A}_{0,B} = \frac{\lambda_B\left(\lambda + \tau_A\right)}{\lambda \tau_A}\mu_0 \tag{6.16}$$

Normalized γ Pheromone Filter

The normalized γ pheromone filter implements a simple one-tap infinite impulse response averaging filter with joint pheromone decay. The inspiration for such an approach comes from the single link pheromone analysis and the corresponding pheromone filter. It also closely mirrors traditional averaging filters.

The pheromone analysis is done similarly to the previous. This filter requires that arriving pheromone be normalized according to the time since a packet last arrived on that link, as shown in Equation 6.17.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ P_{i,s}^n &\leftarrow P_{i,s}^n \cdot e^{-(t-t_s^n)\tau} \\ P_{r,s}^n &\leftarrow P_{r,s}^n + \left[1 - e^{-(t-t_{r,s}^n)\tau}\right] \cdot \gamma \end{aligned} \tag{6.17}$$

The amount of pheromone on each link is shown in Equation 6.18.

$$\begin{aligned} P_{0,B}^A &= \left(\frac{\lambda}{\lambda+\tau_A}\right) P_{0,B}^A + \left(\frac{\lambda_B}{\lambda}\right)\left[1 - \left(\frac{p_{0,A}^B \lambda_B}{p_{0,A}^B \lambda_B + \tau_A}\right)\right] p_{0,A}^B \mu_0 \\ P_{1,B}^A &= \left(\frac{\lambda}{\lambda+\tau_A}\right) P_{1,B}^A + \left(\frac{\lambda_B}{\lambda}\right)\left[1 - \left(\frac{p_{1,A}^B \lambda_B}{p_{1,A}^B \lambda_B + \tau_A}\right)\right] p_{1,A}^B \mu_1 \\ P_{0,A}^B &= \left(\frac{\lambda}{\lambda+\tau_B}\right) P_{0,A}^B + \left(\frac{\lambda_A}{\lambda}\right)\left[1 - \left(\frac{p_{0,B}^A \lambda_A}{p_{0,B}^A \lambda_A + \tau_B}\right)\right] p_{0,B}^A \mu_0 \\ P_{1,A}^B &= \left(\frac{\lambda}{\lambda+\tau_B}\right) P_{1,A}^B + \left(\frac{\lambda_A}{\lambda}\right)\left[1 - \left(\frac{p_{1,B}^A \lambda_A}{p_{1,B}^A \lambda_A + \tau_B}\right)\right] p_{1,B}^A \mu_1 \end{aligned} \tag{6.18}$$

The time pheromone evolution is described by Figure 6.7 which has the same parameters as those from Figure 6.4. Mean link pheromone in the one-zero distribution

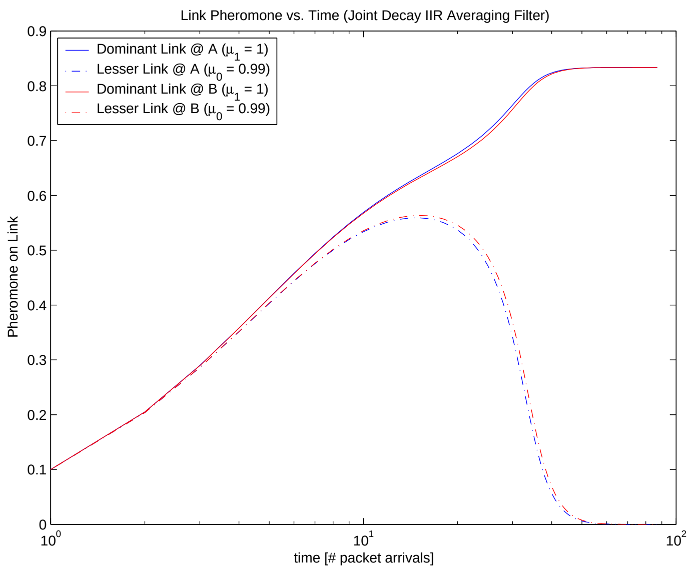

Figure 6.7: Link Pheromone vs. Packets Received using the $\bar{\gamma}$ Pheromone Filter

case is shown,

$$\tilde{P}_{0,B}^A = \frac{\lambda_B (\lambda + \tau_A)}{\lambda(\lambda_B + \tau_A)} \mu_0 \tag{6.19}$$

Note that,

$$\lim_{\lambda \to \infty} \tilde{P}_{\tilde{\gamma}PF} = \mu_0 \tag{6.20}$$

which not only confirms the normalization but also shows that the expected link pheromone has an upper limit of the mean of the incoming pheromone.

Probabilistic Bellman-Ford

In an effort to test a pheromone update method more similar to traditional approaches, the probabilistic Bellman-Ford (pBF) scheme is developed. As in the original, better paths are discovered by comparing the current best known solution to new information. If the utility of the new path is better, the routing table is updated. Thus, the pheromone update equation is,

$$\begin{aligned} \forall i \in \mathcal{N}^n, \quad & P^n_{i,s} \leftarrow P^n_{i,s} \cdot e^{-(t-t^n_s)\tau} \\ if\ P^n_{r,s} < \gamma, \quad & P^n_{r,s} \leftarrow \gamma \end{aligned} \tag{6.21}$$

All pheromone decays concurrently as in the previous methods.

The pBF pheromone update method cannot be analyzed in a similar way because it is nonlinear. For the purposes of the analysis presented here it suffices to note that the link pheromone is upper bounded by the utility of the link.

$$\tilde{P}^A_{0,B} \leq \mu_0 \tag{6.22}$$

6.3.3 Analysis

As shown by the simulation results of Figures 6.2 and 6.3, the γ pheromone filter has the best goodput performance but poor ability to track changes in the network. Probabilistic Bellman-Ford is able to do better in delivery efficiency, while the normalized γ pheromone filter scores best in this regard. These results can be explained based on the analysis in the previous section.

The γ pheromone filter maintains the largest equilibrium pheromone level on the best link. Since pheromone takes time to decay, this allows it to route on a particular link longer as the underlying link metric changes; large amounts of pheromone imply a large link forwarding probability. Due to this hysteresis effect on pheromone decay, suboptimal links are used longer which prevents the algorithm from adapting quickly. In essence, γPF tends to use a known good route and achieves a high goodput, but it is unwilling to change that route in the face of varying metrics.

The latter two update methods see less goodput but higher adaptivity compared to the former, both for similar reasons. They maintain less pheromone on the links and because of this are able to adapt to changes faster. Less pheromone requires less time to decay. Note that $\tilde{P}_{\bar{\gamma}PF} < \tilde{P}_{\gamma PF}$ due to an additional term of $\lambda\lambda_B$ in the denominator of the former. $\tilde{P}_{\bar{\gamma}PF}$ is also upper bounded by the mean of the incoming pheromone, as shown in Equation 6.20, while $\tilde{P}_{\gamma PF}$ has no such restriction. Differences in pheromone between links is relatively less than with γPF, thus the link probabilities are also less decisive during transition periods while the algorithm is choosing a new link. This leads to wandering packets which eventually timeout in the network and result in lower goodput.

Regular and Uniform Ants

In light of the results here regarding the qualitative function of system parameters and update methods, it is possible to generalize the SI routing algorithms into one of two categories. There are called Regular Ants and Uniform Ants [46]. Regular

Ants responsible for updating pheromone are also biased towards pheromone. Uniform Ants that update pheromone are unbiased by it and instead move uniformly over communications links.

Regular Ants Given the choice between two links, a Regular Ants algorithm such as ABC, AntNet, or Termite will tend towards using only the link with the highest utility. There will be a one-zero distribution between two links of unequal utility. If both links have the same utility, they will be used equally. Since agents laying pheromone are also biased towards pheromone, they become biased towards their own trails on return trips. It is a positive feedback loop that the agents cannot stop. If a network is expected to adapt to a changing environment, this feature of the algorithm is extremely detrimental. A link with zero probability will never be used again, even if its utility should increase in the future.

The most common solution for preventing a one-zero asymptotic distribution in a regular ants algorithm is to include a noise factor in the forwarding equation. With small probability, a link will be uniformly chosen instead of being biased by pheromone. The beneficial properties of the uniform algorithm can still be enjoyed while still using pheromone influenced data messages. These features can be found in the Termite algorithm in Figures 6.5 and 6.6. Only the best link is used if K is zero.

Uniform Ants A Uniform Ants algorithm does not suffer from the one-zero problem. Its asymptotic link distribution is proportional to the utility of each link. Since agents are not biased by pheromone, they travel the network blindly and update each path through the network equally. This approach has the advantage of using each link proportionally to its utility and thus being able to fairly balance the traffic load across the network. However, the pheromone updating packets cannot be used to carry data because the source cannot be sure of their final destination. In order to update the network, special uniform ant control packets must be used. The load on the network increases and data is prevented from being sent.

Termite Termite attempts to mitigate the disadvantages of regular ants by using normalized pheromone update. Even though packets are biased by pheromone, they only produce a pheromone value proportional to the true utility of the link. A disproportionate amount of pheromone cannot be placed on a link which will caused routing to become unbalanced to entirely unfair.

6.4 An Optimal Pheromone Decay Rate Heuristic

Over the course of this and the last chapter, the dynamics of pheromone according to various parameters has been explored. With some intuition established, it is possible to use the previous results and calculations to create some heuristics of the optimal pheromone decay rate. If the selected decay rate is larger than the optimal, $\tau > \tau^*$, then the network will forget its state too fast and throw away relavent information. If the decay rate is too low, $\tau < \tau^*$, then the network will retain too much information make suboptimal decisions.

The Maximum Pheromone and Filter Cutoff τ^* heuristics are presented. The first uses the expected pheromone results calculated in the previous section and calculates a decay rate to remove that pheromone in a timely fashion. The latter heuristic uses the pheromone filter interpretation directly in order to calculate a decay rate which will set the cutoff frequency of the filter at the desired point.

6.4.1 Maximum Pheromone Decay

A heuristic is developed for the pheromone decay rate based on the expected amount of pheromone resident on a link and the amount of time between "significant" events in the network that would require a change in routing behavior. The intuition is that pheromone should decay at a rate such that the expected amount could decay by the time a change in routing behavior is needed. Another perspective is that information from the last network event should be decayed by the time the next one happens.

An upper bound of the expected amount of pheromone on a link was derived earlier in the chapter. It is based on the average amount of arriving pheromone, μ, the rate of pheromone observation, λ, the pheromone decay rate, τ, and the packet arrival rate, λ_B. For simplicity, it is assumed that a communications session between source and destination is symmetric; $\lambda_A = \lambda_B = \frac{\lambda}{2}$.

The network event interarrival time, t_c, is the time between significant network events. It may also be viewed statistically as the route correlation time. It is the time during which the statistics of incoming pheromone accurately reflect the state of the route to destination; it is the path correlation time. An upper bound on this time could be the link lifetime, although specific statistics are difficult to determine, especially since routing is probabilistic. The link lifetime can be either measured or calculated [78]. Routing patterns must be reconsidered after the loss of a route. This idea agrees with the intuition that pheromone should decay faster the more volatile the network topology. Since links last a shorter amount of time, the pheromone must decay faster in order to account for the high route volatility.

It should be noted that this approach assumes that packets arrive faster than network events happen. If this is not the case, then the algorithm is better off considering the pheromone in each packet separately, without regard to previous information. Any memory a node would maintain would be useless by the time a new packet arrived in order to update the system. It is for this same reason that Termite performs so poorly under conditions of slow packet rate compared to the rate of change of the network. The rate of incoming information about the network is not high enough to properly adapt to it. This could be considered something like a Nyquist's rule for networks [69].

The general equation, $\tilde{P}e^{-t_c\tau} = K$, must be solved for the decay rate, where $\tilde{P}$ is the expected amount of pheromone on a link as derived earlier in this chapter. The second term is the decay factor based on decay time and decay rate. The decay factor must decay the expected pheromone to the pheromone threshold, K. Such low pheromone levels will not have a tangible effect on the link probability. From this, the pheromone decay rate can be derived as a general guideline to its optimal value.

Because different amounts of pheromone reside on a link depending on the specific pheromone update method being used, the heuristic reflects this as well. Unfortunately there is no closed form expression for the solution of these equations, however a numerical result is easily found.

γ **Pheromone Filter** Equation 6.23 shows the decay heuristic equation for the γ pheromone filter. It is of note that the decay rate is unbounded as the packet arrival rate increases. The γ pheromone filter does not incorporate any sort of pheromone upper bound, and as more packets arrive, the more pheromone is laid on a link.

$$\frac{\lambda + \tau^*_{\gamma PF}}{2\tau^*_{\gamma PF}} \cdot \mu \cdot e^{-t_c\tau^*_{\gamma PF}} = K \tag{6.23}$$

Normalized γ Pheromone Filter Equation 6.24 shows how the normalized γ pheromone filter estimates the pheromone decay rate. Unlike its sibling, this estimate is bounded (of course, it is normalized) and thus the estimate is upper bounded by $\tau^*_{\bar{\gamma}PF} = \frac{1}{t_c} \ln\left(\frac{\mu}{K}\right)$ as the packet arrival rate increases; The pheromone decay heuristic is a line with respect to the network event rate.

$$\frac{\lambda + \tau^*_{\bar{\gamma}PF}}{\lambda + 2\tau^*_{\bar{\gamma}PF}} \cdot \mu \cdot e^{-t_c \tau^*_{\bar{\gamma}PF}} = K \tag{6.24}$$

6.4.2 The Pheromone Filter Cutoff

The pheromone filter provides an interesting perspective on the determination of an optimal pheromone decay rate. As explained during the derivation, the cutoff frequency of the filter is a function of β, and thus tunable by adjusting τ. It is very tempting to use this connection to say that the optimal pheromone decay rate is that which properly tunes the filter to the desired cutoff frequency. Such an approach was described earlier in the chapter and is also shown in Figure 6.8a shows a. However, there are a number of caveats which make the implementation of such an idea much more difficult than first apparent.

The greatest difficulty is that the analysis assumes a constant mean packet arrival rate, λ. While mathematically convenient, this is absolutely not the case in any practical sense. The decay rate would have to be changed depending on the current statistics of the packet arrival rate, which means that extra infrastructure must be installed to estimate the parameter. The performance of this extra estimation stage will further influence the operation of the algorithm.

Not only must λ be estimated and τ matched, but there is also a question of the "sampling rate" of the system itself. If the designer wished the network to be able to adapt to events happening at a certain rate, then clearly there must be enough traffic in the system which can measure the effects of this feature. This is again the network sampling issue. Equation 6.11 shows that as the required cutoff frequency tends towards $\omega = \frac{2\pi}{\lambda}$, the frequency representation limits of this digital system, the value of τ tends towards infinity. That is, if the network must be able to adapt to faster phenomena, the pheromone should decay faster as well because relevant information about the network has a shorter lifetime. If the network is changing faster than packets are arriving, the decay rate should be infinity; there is no point to averaging because each successive sample of the network (pheromone arrival) has no correlation at all with previous samples. Alternatively, if the packet arrival rate is substantially greater than the rate of change of the network, then the decay rate can afford to be a little bit smaller. The system has a greater amount of information about the network over the relevant period of the time and thus can average over more packets.

There are of course drawbacks to this system. The most complete implementation of this idea is to install such a filter for each type of pheromone on each link of each node. The objective of the routing algorithm in general is to be as lightweight as possible. So many filters and λ estimators on top of per-packet probability computation likely constitutes too much overhead. It is not clear at this point in the development of ad-hoc networks what the packet rate for a typical application might be compared to the network volatility. Therefore, it is difficult to predict how well such a system might perform in general.

The previous chapter showed that a link independent implementation of such an averaging filter approach is doomed. Performance was by far substandard. The promise of independent per-link pheromone estimation and favorable load balancing properties were not realized. A more complex implementation is required.

6.4.3 Heuristic Comparison

Figure 6.8 shows a comparison between the maximum pheromone and filter cutoff decay heuristics. The specific parameters used here are, $\lambda = 8$, $\mu = \frac{1}{4.5}$, and $K = \frac{1}{32}$. They are meant to reflect some generic situation. The minimum network correlation time (maximum instance rate) shown here is $\frac{\lambda_A}{2} = \frac{\lambda_B}{2} = 2$ since that is the maximum frequency representable by the discrete time filter. Unfortunately the heuristics are not directly comparable. The maximum pheromone heuristics for γPF and $\bar{\gamma}$PF follow the same trend, however the filter cutoff heuristic is substantially smaller. The latter does not take into account the absolute amount of pheromone on the link, rather only the variation in that pheromone.

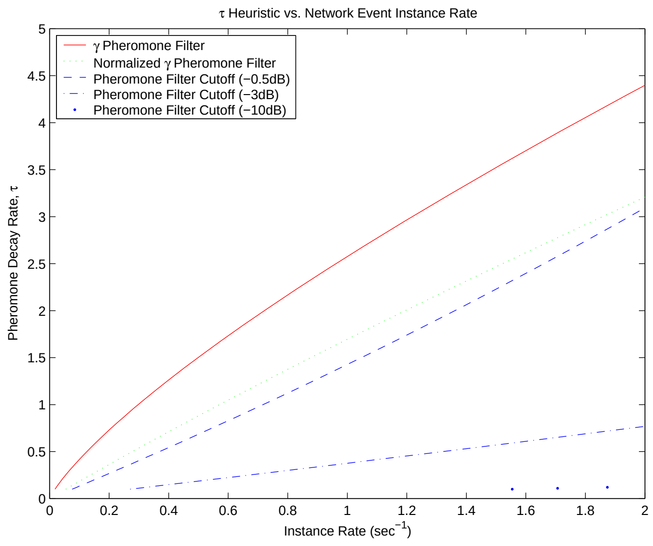

Figure 6.8: Pheromone Decay Rate Heurisitcs

Each of these heuristics is roughly linear, especially that for $\bar{\gamma}$PF. This should come as no surprise. The expected link lifetime (an upper bound on the route lifetime or correlation time) can be calculated according to [78]. Link lifetime is inversely proportional to node speed, and thus the link expiration or break rate is

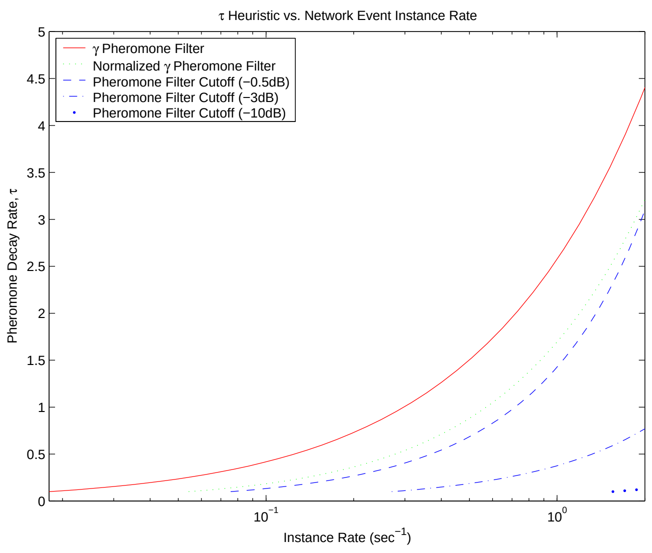

Figure 6.9: Pheromone Decay Rate Heurisitcs

directly proportional to node speed. This is corroborated by the link break rate results in Figure 5.1 which are linear with respect to node speed. If the network changes linearly with node speed, then routing information must also expire in the same way. Thus, the pheromone decay rate increases linearly with node speed. Such a conclusion vaguely agrees with the results of Figure 5.8 which shows the measured optimal pheromone decay rate of the simulations of the previous chapter. Unfortunately that work does not contain enough fidelity to draw any serious conclusions.

But this still leaves the question of how good the heuristic really is. Because the expected pheromone level and the path correlation time are upper bounds, the calculated pheromone decay rate is not a bound. Table 6.1 shows the calculated link expiry rate for the simulation parameters of the previous chapter. Figure 6.9 shows the heuristics on a semilog plot for more detail. The table indicates that links break on the order of 10^{-2} per second, and the heuristics recommend a lowest pheromone decay rate on the order of 10^{-1}. Figure 5.8 agrees with this generally with measured τ^* ranging from 0.5 to unity. Unfortunately no strong conclusion can be drawn due to the low resolution of the data.

Table 6.1: Calculated Link Expiry Rate

0 [m/s]	0 [link break/s]
1	0.0059
2	0.0118
5	0.0294
10	0.0589

6.5 Conclusion

An extension of an analytical model for a mobile wireless ad-hoc network is presented. This model was used to investigate the properties of the Termite swarm intelligent MANET routing algorithm. The mean pheromone on a single link and in a system of two links were determined. These results were compared for three different pheromone update methods, including the γ pheromone filter, the normalized γ pheromone filter, and the probabilistic Bellman-Ford. Relationships between parameters were explored and a scale invariant parameter was found. This parameter is shown to be critical in determining the amount of pheromone resident on a link, which in turn influences performance. This is true not only for goodput but also in adaptability. The analysis revealed a linear filtering perspective in which link utility is directly estimated with the pheromone rather than simply using it as a routing heuristic. The maximum pheromone and filter cutoff pheromone decay rate heuristics are derived from the model presented in this chapter. They are compared to simulation data from the previous chapter, although the results are inconclusive due to the poor quality of the data. The heuristics can serve as an order-of-magnitude bound to selecting the optimal pheromone decay rate.

CHAPTER 7

RETERMITE

7.1 Introduction

This chapter updates Termite with all of the knowledge gained over the course of its development in the previous three chapters. The end result shows how the MANET routing problem can be competetively solved with only minimal use of control traffic. A small amount of control information is imbedded in every data packet, which is usually sufficient for the network to maintain a current and accurate view of its state. The end result is a routing algorithm requiring only data traffic in the network under many circumstances. The version of the Termite used here with all of the previous enhancements is called *ReTermite* [80]. ReTermite is compared to the state-of-the-art and proposed MANET routing standard AODV, the Ad-hoc On-demand Distance Vector routing protocol.

7.2 ReTermite

This section will review the swarm intelligent ReTermite MANET routing algorithm in detail. It may be described simply as follows. Each node in the network has a specific pheromone scent. As packets move through the network on links between nodes, they are biased to move in the direction of the pheromone gradient of the destination node. The specifics of this operation are governed by the packet forwarding equation. Packets follow the pheromone gradient while laying pheromone for their source on the same links. The specific amount of pheromone deposited by a packet on a link, as well as how that pheromone behaves over time, is governed by the pheromone accounting process. Changes in the network environment, such as topological or path quality changes, are accounted for by allowing pheromone to decay over time. This requires paths to be continuously reinforced by new traffic; new information about the network is added to links. In this way, consistent pheromone trails are built through the network. Each node records the amount of pheromone that exists for each destination on each of its links. This creates a routing table similar to those found in traditional distance vector routing algorithms and is known as the pheromone table.

7.2.1 Pheromone Table

The pheromone table is ReTermite's routing table, and is that same as in the original Termite algorithm as shown in Figure 4.1. It maintains the amount of pheromone on each neighbor link for each known destination. Note that neighbors may also appear in the table as a destination. The pheromone table is generally oriented such that each column represents a destination, and each row a neighbor. When a neighbor is gained, an extra row is added to each column showing the pheromone on the link to the new neighbor, which is initialized to zero. If a neighbor is lost, its corresponding row is removed from the table. When a destination is gained, the current list of neighbors is replicated for the new destination but with all pheromone values reset to zero. If a destination is lost from the pheromone table, the column is simply removed.

A neighbor row is never removed unless the link is explicitly lost through communications failure. Even if the pheromone on that link decays, it is still retained since it still exists as a communications option. Alternatively, a destination column is removed if all of the pheromone on it decays. It is necessary to do this so that the algorithm knows when it should issue a route request for the destination. This procedure is initiated if the destination does not exist in the pheromone table.

For all nodes in the network, n, pheromone values in the pheromone table are referenced with, $P^n_{i,d}$, where $i \in \mathcal{N}^n$, and $d \in \mathcal{D}^n$. These sets represent the current set of neighbors and destinations that node n is aware of. $P^n_{i,d}$ is thus the amount of pheromone at node n for destination node d on the link to neighbor node i.

7.2.2 Packet Forwarding

In order to forward a packet towards its destination, the forwarding equation with source pheromone repel is used to determine the next hop neighbor. This formula maps the destination d pheromone on each outgoing link i at node n, $P^n_{i,d}$, to the probability that that link will be used to forward the packet to the destination, $p^n_{i,d}$. In order to account for the source pheromone, the source pheromone distribution, $p^n_{i,s}$, is also used. The specific next hop neighbor is chosen randomly according to the meta distribution, $\hat{p}^n_{i,d}$, which reflects the source pheromone repel. The forwarding equation is shown in Equation 7.1 below.

$$\begin{aligned}
p^n_{i,d} &= \frac{(P^n_{i,d} + K)^F}{\sum_{j \in \mathcal{N}^n} (P^n_{j,d} + K)^F} \\
p^n_{i,s} &= \frac{(P^n_{i,s} + K)^F}{\sum_{j \in \mathcal{N}^n} (P^n_{j,s} + K)^F} \\
\hat{p}^n_{i,d} &= \frac{p^n_{i,d} \left(p^n_{i,s}\right)^{-R}}{\sum_{j \in \mathcal{N}^n} p^n_{j,d} \left(p^n_{j,s}\right)^{-R}}
\end{aligned} \tag{7.1}$$

The constants F, K, and R are used to tune the routing behavior of ReTermite, and are the same as those found in Termite.

7.2.3 Pheromone Accounting

Upon the reception of any packet, pheromone must be updated according to the pheromone accounting procedure. This process, or even the eventual interpretation of its output, can take many different forms. Three pheromone update methods are considered in this paper. They represent the best approaches found in a comparison of eleven update strategies two chapters ago. The approaches are γ pheromone filtering (γPF), normalized γ pheromone filtering ($\bar{\gamma}$PF), and probabilistic Bellman-Ford (pBF).

Two critical mechanisms are used in order to facilitate pheromone accounting. These include true continuous pheromone decay and piggybacked routing information. Promiscuous mode packet reception is also found to be very helpful.

True Continuous Pheromone Decay

As described by the swarm intelligent framework, pheromone must decay in order to provide a negative feedback mechanism to the system; old information must be removed from the system. Past swarm intelligent routing algorithms have either decayed pheromone on regular intervals or decayed it upon packet arrival.

ReTermite features true continuous pheromone decay which properly simulates the continuous decay of pheromone. Pheromone is decayed when packets arrive and also when it is checked to create a probability distribution when forwarding a packet. In essence, pheromone is decaying all of the time, and the update algorithm implements this by noting all times at which the pheromone has been measured for any reason.

This approach is better able to account for reductions in certainty about network state. If no packet is received from a destination for some time, the routing probabilities now reflect the fact that less timely information is maintained about the whereabouts of the destination; the routing probabilities tend towards a uniform distribution over all neighbor nodes.

Piggybacked Routing Information

In order to easily disseminate routing information, each packet contains the cumulative metric measurement experienced as it has traveled through the network. The cumulative metric measurement is simply a number detailing the total metric utility that the packet has experienced enroute to the current node. The metric is updated at each node that the packet visits. On many platforms, this information results in the addition of only four bytes of overhead. While it is not new to piggyback routing information on data packets, it should be stressed that this information is piggybacked on *all* packets. Even data packets are used to actively update the network. Other algorithms such as AntNet, CAF, and ANSI use a stack to store more path history, but do so only for control packets.

Permiscuous Mode

Nodes are expected to exist in a broadcast medium. They may eavesdrop on the communications of neighbors and incorporate overheard routing data into their own routing table. This technique is used in many other ad-hoc routing algorithms, and is found to be particularly useful in Termite. In principle it is not obligatory, however it does afford a notable increase in performance.

Pheromone Update Methods

Each pheromone update method tested in this paper is described. Each case describes how the pheromone table is updated based on a packet arriving at node n, from source node s, previous hop r, and going to destination d. The previous hop is the node which just transmitted the packet. If n is not designated as the packet's next hop, it updates the pheromone table in the way described here and then drops the packet. The time at which the packet is received is t, and the last time at which the pheromone for node x was observed at node n is t_x^n. The last time which the pheromone for node x on the link to node y at node n was observed is $t_{y,x}^n$.

γ Pheromone Filter γPF deposits pheromone on the link that a packet is arriving on. The pheromone on the packet, γ, is equivalent to the utility of the path that the packet has taken. All pheromone for the source of the arriving packet decays exponentially.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ P_{i,s}^n &\leftarrow P_{i,s}^n \cdot e^{-(t-t_s^n)\tau} \\ P_{r,s}^n &\leftarrow P_{r,s}^n + \gamma \end{aligned} \tag{7.2}$$

Normalized γ Pheromone Filter $\bar{\gamma}$PF is a normalized version of γPF. It is essentially an infinite impulse response averaging filter with joint pheromone decay. Only a fraction of the received pheromone is added based on link observation time. $\bar{\gamma}$PF effectively limits the amount of pheromone on a link with its averaging properties. γPF allows much larger amounts since it is unnormalized. Additionally, the form of $\bar{\gamma}$PF make this variant of ReTermite similar to algorithms used by the reinforcement learning community.

$$\begin{aligned} \forall i \in \mathcal{N}^n,\ P^n_{i,s} &\leftarrow P^n_{i,s} \cdot e^{-(t-t^n_s)\tau} \\ P^n_{r,s} &\leftarrow P^n_{r,s} + \left[1 - e^{-(t-t^n_{r,s})\tau}\right]\gamma \end{aligned} \tag{7.3}$$

Probabilistic Bellman-Ford The probabilistic Bellman-Ford algorithm is designed as an asynchronous version of the Bellman-Ford algorithm where packets are routed probabilistically and update routing information at each node they visit. This is in contrast to the original algorithm which requires a local flood of control packets to update information. Unlike all other techniques used here, this one is nonlinear. If a packet is received with information of a better path over the receiving link than is already known, then the utility estimate (pheromone) is updated. Pheromone decays on all links in order to degrade the estimates over time.

$$\begin{aligned} \forall i \in \mathcal{N}^n, &\quad P^n_{i,s} \leftarrow P^n_{i,s} \cdot e^{-(t-t^n_s)\tau} \\ if\ P^n_{r,s} < \gamma, &\quad P^n_{r,s} \leftarrow \gamma \end{aligned} \tag{7.4}$$

7.2.4 Route Discovery

In case there does not exist any destination pheromone at a node for a packet to follow, a route discovery procedure must be initiated. ReTermite uses the traditional flooding approach and does not use any optimizations such as gossiping or expanding ring search. Only the next hop node designated in the packet may rebroadcast the RREQ and does so only if is it unable to answer the route request itself. This is done even if other neighbors have already transmitted a route reply (RREP) to the RREQ source. Any nodes overhearing a RREQ may reply to the request with a RREP if they have destination pheromone. A RREP is unicast back to the source normally by probabilistically following the source pheromone. A RREP is formatted such that its source is the destination of the RREQ. This creates a destination pheromone trail back to the RREQ source. It is necessary to maintain a list of previously seen RREQ packets according to the source address and a sequence number. Route requests also serve to spread source pheromone into the network.

The packet that triggers a route request is cached for a route request timeout period before it is dropped. Any additional packets received during this period for the same destination are also cached. If the hold time since the first packet was held is exceeded then all of the held packets for the sought destination are dropped. If a route reply is received while there are packets cached for the destination, they are processed normally according to the forwarding equation.

7.2.5 Route Repair

ReTermite has no concept of route repair in the traditional sense. Each next hop is computed online, and every node has an estimate of the utility of each link to deliver a packet to its destination. If a link should fail, the neighbor is simply removed from the routing table and the next-hop probabilities are recomputed for the remaining set. If all neighbor nodes are found to have disappeared (perhaps after many unsuccessful retransmissions), the packet is dropped. There is no such thing as a route error or route error packet (RRER).

7.2.6 Packet Structure

There are three types of packets in Termite, data (DATA), route request (RREQ), and route reply (RREQ). They can be all considered within one generic packet format with the following fields. This list does not assume a pheromone stack, as has been mentioned in previous chapters. Certain information such as the previous and next hop IP address can often be obtained from lower layers of the network stack, as seen in the AODV specification [41]. The total size of the header shown here is 24 bytes, excluding any user data. If necessary, an additional four bytes of flags could be added.

Packet Type

This field is one byte in size. It's value describes the purpose of the packet, data, route request, or route reply.

Source IP Address

This field is four bytes and describes the IP address of the data source.

Destination IP Address

This field is four bytes and describes the IP address of the data destination.

Previous Hop IP Address

This field is four bytes and describes the IP address of the previous hop.

Next Hop IP Address

This field is four bytes and describes the IP address of the next hop.

Pheromone

This field is four bytes and describes the amount of pheromone carried by the packet.

Time-To-Live (TTL)

This field is one byte and describes the remaining allowed hop-count for the packet. It is initialized to the maximum TTL and decremented at each visited node. The packet is dropped when this counter reaches zero.

Data Length

This field if two bytes and describes the length of the data field in this packet; the amount of data carried by this packet.

Data

This field contains all of the data carried by the packet.

7.3 Simulation

A number of different scenarios are simulated to compare the performance of the variations of Termite, along with the presented pheromone update methods. These results are also compared to the standard MANET Ad-hoc On-demand Distance Vector (AODV) routing protocol, as described in [41]. Simulations are designed to test the effect of node mobility on the global performance metrics.

7.3.1 Simulation Environment and Parameters

A common test scenario is used in which 100 mobile nodes are distributed uniformly over an area 2200 meters by 600 meters. Each node uses a simulated IEEE 802.11b MAC layer with 2 Mbps data rate and a 250 meter transmission range. The standard MAC layer has been modified to allow promiscuous reception of all in-range transmitted packets, and also to return unsuccessfully transmitted packets back to the routing layer for reassessment. Nodes move according to the random waypoint mobility model with zero pause time and a uniform speed. These parameters are the same as those in [81]. Speeds are varied over 1, 5, 10, and 20 meters per second. All runs are 600 seconds long and all data points are averaged over at least two runs for high speeds and five runs for low speeds. Ten nodes send 512 byte data packets with exponentially distributed interarrival times with a mean of 0.5 seconds to a unique communications partner, which replies to each received packet with an acknowledgement. Both AODV and ReTermite optimize for path length. AODV parameters are set according to the specifications found in [41]. The maximum TTL for any packet is 32. ReTermite parameters include, $K = \frac{1}{32}$, $F = 10.0$, $\tau = 2.0$, and $R = 0.5$. ReTermite holds packets for as long as AODV's ACTIVE_ROUTE_TIMEOUT parameter, which in this case is 1.28 seconds. All simulations are completed using Opnet [74].

7.3.2 Evaluation Metrics

A number of metrics are used to determine the utility of the proposed algorithms. These include data goodput, average path length, control packet overhead, control packet distribution, medium load, medium efficiency, medium inefficiency, link failure rate, and end-to-end delay.

Data Goodput

Data goodput is a classic evaluation metric for routing algorithms. It is the fraction of successfully delivered data packets. This metric should remain as high as possible under any circumstances.

Average Path Length

This metric measures the average number of hops needed for a packet to successfully arrive at its destination. This is a relative measure of protocol efficiency, especially since the network is being optimized to minimize hop count.

Control Packet Overhead

Any routing algorithm should expend as little control traffic as possible in order to successfully deliver data packets. Control packet overhead measures the fraction of control packets to the total number of transmitted packets in the system. Successively transmitted packets are counted individually.

Control Packet Distribution

This metric shows how many of each type of control packet were transmitted. This helps to identify the effectiveness of route request and discovery procedures.

Medium Load

This metric characterizes how inefficient the algorithm is in delivering packets. It is the ratio of the total number of packet transmissions, data or control, to the number of data packets successfully delivered. Successively transmitted packets are counted individually. This is not a general metric, but since the algorithms are optimizing for hop count, the fewer the transmissions the better. This metric should be as low as possible, however it will always be larger than the path length.

Medium Efficiency

Medium efficiency is the ratio of the number of transmissions of successfully arriving data packets to the number of total packet transmissions. Multiple transmissions of the same packet are counted individually. Since access to the communications medium often comes at a premium, it is important that it is only accessed in order to move packets that will arrive at their destination. Medium efficiency is a number between zero and one and values close to unity are desired.

Medium Inefficiency

This metric is related to the previous two and helps to fill in the full performance picture. Medium inefficiency is the fraction of transmitted packets to the number of data packets offered to the network for delivery. Lower numbers are better. Consideration of this metric should ensure that the routing algorithm is making an effort to deliver all packets, instead of just ones that are easy to deliver.

Link Failure Rate

The link failure rate measures the average number of links that are lost per node per second. It is a relative measure of how fast the network topology is changing, and thus how much time each node has to acquire a sensible local routing pattern before its local topology transforms.

End-To-End Delay

The average end-to-end delay of all successfully delivered data packets is measured. This metric gives another perspective on the overall performance of each algorithm. Delay should be minimized.

7.3.3 Results and Analysis

Data Goodput

As shown in Figure 7.1, the data goodput performance of ReTermite is higher than that of AODV. The latter is able to deliver at least 90% of its packets, and ReTermite outperforms it with a moderate 95%. The former sees a more graceful degradation of performance over the latter as node speed increases.

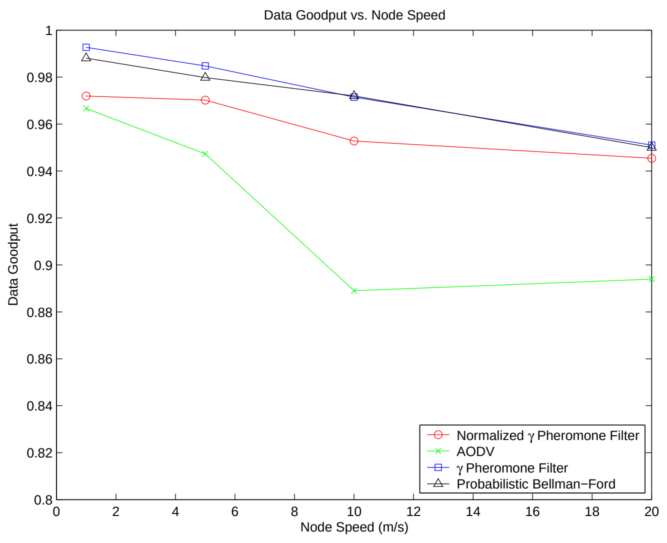

Figure 7.1: Data Goodput vs. Node Speed

Average Path Length

As might be expected, ReTermite has longer paths than AODV. This is illustrated in Figure 7.2. The former pays a cost for reduced control traffic by requiring data packets to take longer paths. The downward trend of AODV is due to the frequency route breaks and subsequent route rediscoveries. As node speed increases, links will break more often and force more route errors and route discovery operations. Since AODV will almost always find the shortest route by way of flooding, the algorithm is put into a mode of constantly finding the shortest route as the network changes. Thus, the average path length is continuously minimized as node speed increases.

ReTermite must rely on successive refinement in order to shorten its paths as the network topology changes. Since the packet rate is relatively low, there are not many opportunities to do so compared to the rate of change of the topology.

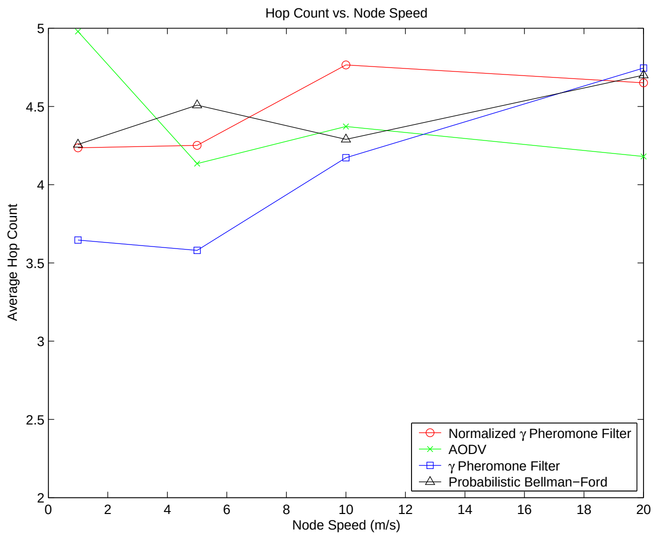

Figure 7.2: Average Path Length vs. Node Speed

Control Overhead

ReTermite shows favorable control overhead properties as compared to AODV in Figure 7.3. Not only does ReTermite have an order of magnitude less control overhead, but it also produces a nearly constant amount over a large range of node mobility. This is true despite the use of flooding for route discovery, and speaks to the effective use of route information caching on the part of ReTermite. AODV suffers so much overhead because it floods the network with a new route discovery every time a route breaks. This weakness is addressed with an expanding ring search, however the details have not been agreed upon in the standard [42].

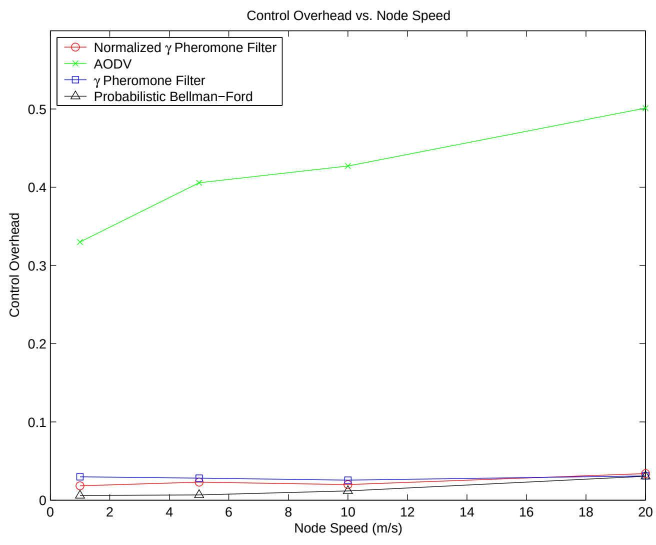

Figure 7.3: Control Overhead vs. Node Speed

Control Packet Distribution

The control packet distribution data from Figure 7.4 is truly telling. Compared to AODV, ReTermite uses little control traffic. As mentioned above, AODV must issue a route discovery flood whenever there is a route break. ReTermite is spared this because of its retransmission link repair policy. A full route discovery is almost never needed, which eliminates the majority of the control overhead. For AODV this proportion increases with node speed as links break more often. The most limiting factor is the number of route request packets; there are so many because such packets are flooded which ultimately generates a choking amount of overhead.

Alternatively, ReTermite produces more route reply packets than route requests. This attests to both the liberal route reply policy (any overhearing node with route information can generate a route reply) and also to the route caching of ReTermite. Since each node keeps route information about every other node, it is not difficult to find a pheromone trail.

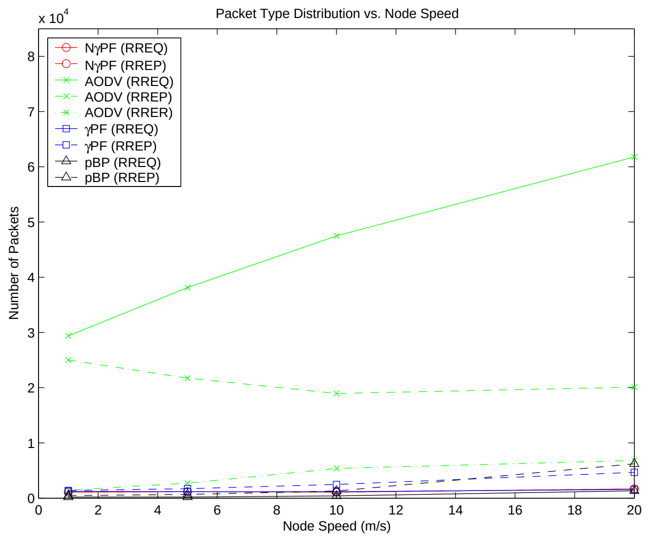

Figure 7.4: Packet Type Distribution vs. Node Speed

Medium Load

While the control overhead statistics are quite positive, it is ultimately necessary to compare the total amount of access to the medium needed to deliver a packet. It has already been shown that ReTermite has favorable control traffic but suffers in average path length as compared to AODV. Figure 7.5 shows that ReTermite is able to do better than AODV with regards to the total number of transmissions required to deliver a packet successfully. They perform equally at 20 m/s. Both algorithms show an increasing load on the medium as the network volatility increases. In the case of AODV, this is because of the increasingly large amount of control traffic generated. For ReTermite, this is because data packets must take longer paths to explore the network as the topology changes faster.

Medium Efficiency

The results of Figure 7.6 show that ReTermite is able to easily deliver packets at low speeds. Both AODV and ReTermite expend more effort to deliver a packet

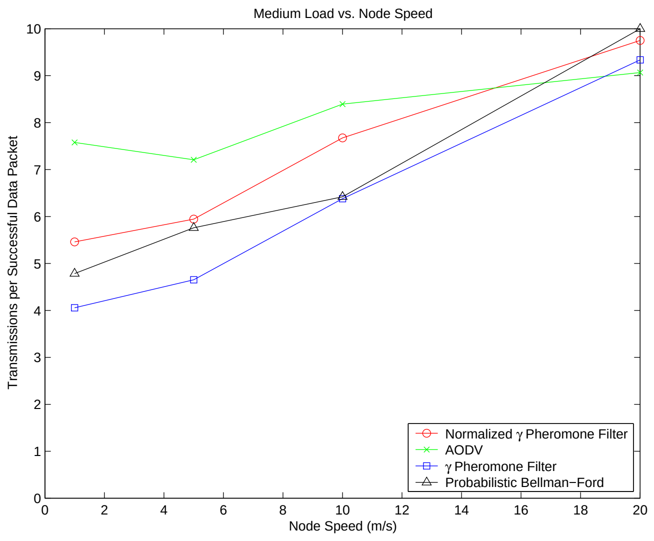

Figure 7.5: Medium Load vs. Node Speed

successfully, again with similar performance at 20 m/s. Medium efficiency decreases linearly with node speed in all cases. This metric reconfirms the results of the medium load.

Medium Inefficiency

This metric confirms the results of the previous two. Figure 7.7 shows that ReTermite makes a best effort to deliver all packets that are offered. It able to do so with fewer packet transmissions at low speed but suffers at higher speed by a margin of 13%. The reason for this is due to unsuccessful data packets that wander the network until they exceed their Time-To-Live (TTL). This behavior causes unnecessary transmissions which increases the metric. Naturally this happens more at high speeds when the network is topology is more volatile. When this the medium inefficiency is readjusted to include only successful data packets (the medium load in Figure 7.5), the outcome is somewhat more even.

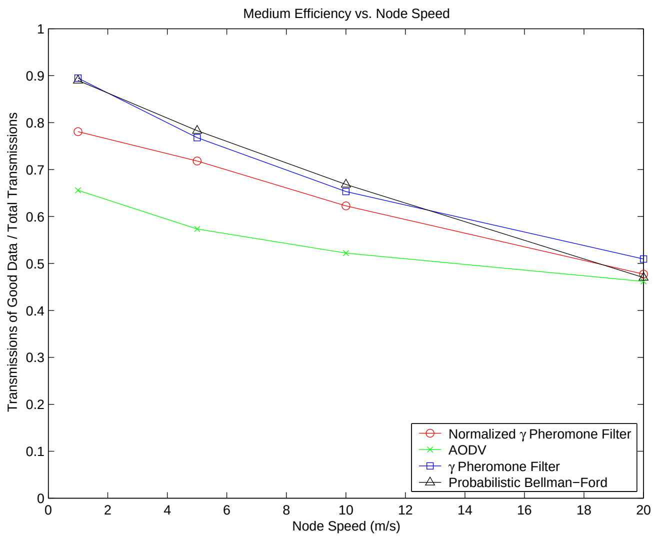

Figure 7.6: Medium Efficiency vs. Node Speed

Link Failure Rate

Figure 7.8 shows how the link failure rate changes with node speed. As expected from previous simulations, the trend is linear. The results may be somewhat skewed from standard link failure rates because this data is only measured when a communications attempt fails. That is, when a packet is sent on a link but the receiver is unavailable. Since the packet rate is relatively low, this measured link failure rate may also tend towards the low end. However, this metric does give an idea as to how quickly the network is changing and at what rate the nodes should reconfigure themselves.

End-To-End Delay

Figure 7.9 shows the average end-to-end (ETE) delay for each of the compared routing algorithms. The ETE delay of AODV stays constant regardless of node speed while the delay of the ReTermite variants grows from less than AODV at low speed to substantially larger at high speed. The AODV results reflect those

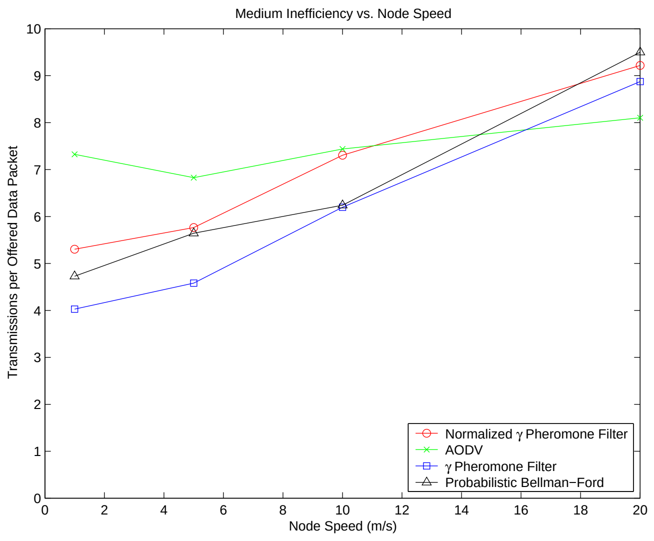

Figure 7.7: Medium Inefficiency vs. Node Speed

reported by [81] and is due to the fact that packets are only sent when a full route is known to exist. The ReTermite delay results are not positive considering that the delay of AODV is nearly an order of magnitude lower at high speeds. However, there are two extraneous issues at work in this situation. The first is that the metric only measures the delay of successful packets. AODV has a lower goodput than ReTermite, and so packets that might have otherwise timed out in AODV due to link or route discovery failure are delivered by ReTermite. This extra goodput comes at the expense of some additional delay incurred by perhaps a longer path length or congestion. It can be shown that when the slowest 5% of packets in ReTermite are not considered (the difference of goodputs between AODV and ReTermite at 20 m/s), then the delay can be reduced by 66%. This property indicates that ReTermite's high delay comes from statistical outliers. These are packets that require a great deal of effort to deliver, packets that AODV does not. The second factor is ReTermite's use of link recovery retransmission. When the network is changing quickly, packets may be retransmitted often due to

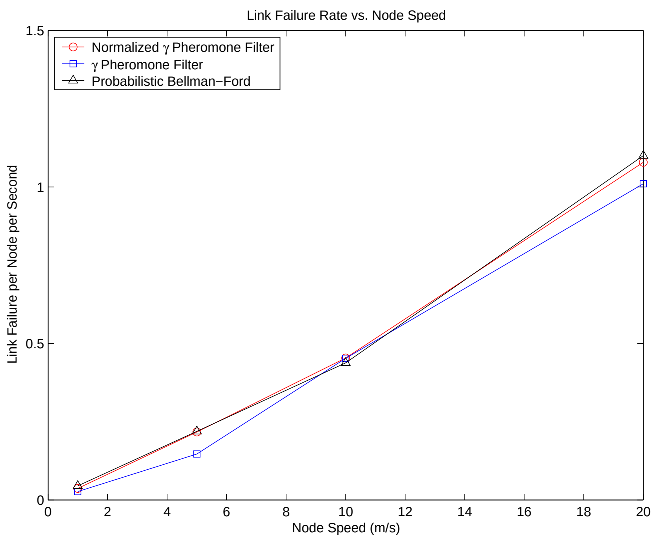

Figure 7.8: Link Failure Rate vs. Node Speed

neighbor loss. If several nodes in the same area are trying to retransmit, this can lead to localized packet storms. This is especially true when using 802.11 which automatically retries up to seven times before reporting a broken link.

7.3.4 Discussion

The results show that ReTermite is able to outperform AODV primarily due to the lack of control traffic and liberal route caching. AODV often finds itself repairing routes which requires route errors and route request floods. ReTermite avoids this complexity and effort through the use of piggybacked route information and promiscuous packet reception. However, ReTermite must find a way to explore the network as well in order to find better routes. It does this by letting data packets do the work. The medium load, efficiency, and inefficiency metrics show what the control overhead does not. The ReTermite approach works well at low speed but gives fewer performance gains at high speed. The number of packets transmissions per data packet is equal between the two algorithms; they both

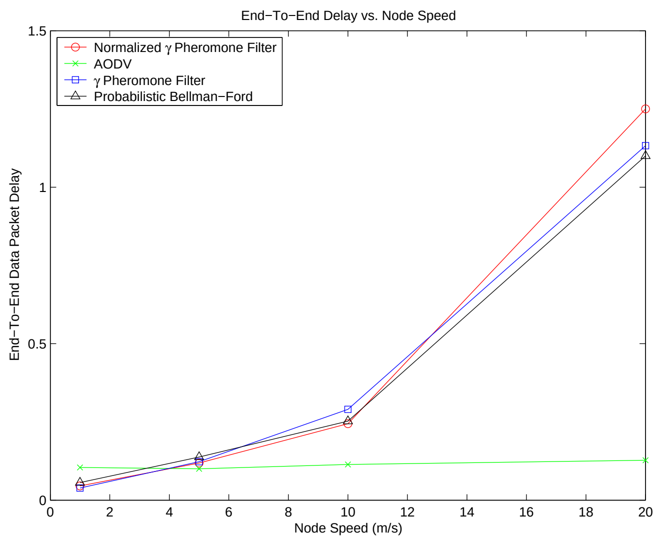

Figure 7.9: End-To-End Delay vs. Node Speed

do the same amount of work to deliver packets. The most noticeable difference between the algorithms at high speed is the end-to-end delay. ReTermite's packet storms hurt performance tremendously.

7.4 A Pheromone Decay Rate Heuristic

Previous discussion indicated that the pheromone decay rate, τ, can be thought of as the learning rate of the network. If the selected decay rate is larger than the optimal, $\tau > \tau^*$, then the network will forget its state too fast and throw away relavent information. If the decay rate is too low, $\tau < \tau^*$, then the network will retain too much information about its state and also make suboptimal decisions. It should be possible to determine the optimal decay rate and compare it to the heuristics developed previously.

7.4.1 τ^* Simulation

Figures 7.10 and 7.11 show how the performance of ReTermite can vary according to the global pheromone decay rate. The parameter was kept constant in the previous simulations at $\tau = 2.0$ in order to make each scenario directly comparable. The figures test τ over two orders of magnitude at node speeds of 1 m/s and 10 m/s. As is shown, the appropriate selection of this parameter is of critical importance. Using the normalized γ pheromone filter at 1 m/s, $\tau \approx 0.1$ is the best choice. $\tau \approx 1.0$ is best at 10 m/s. This is most easily seen from the achieved goodput and medium efficiency. The performance of ReTermite was quite good in the first set of simulations, and these results show that the performance could be even better if optimal parameters are chosen.

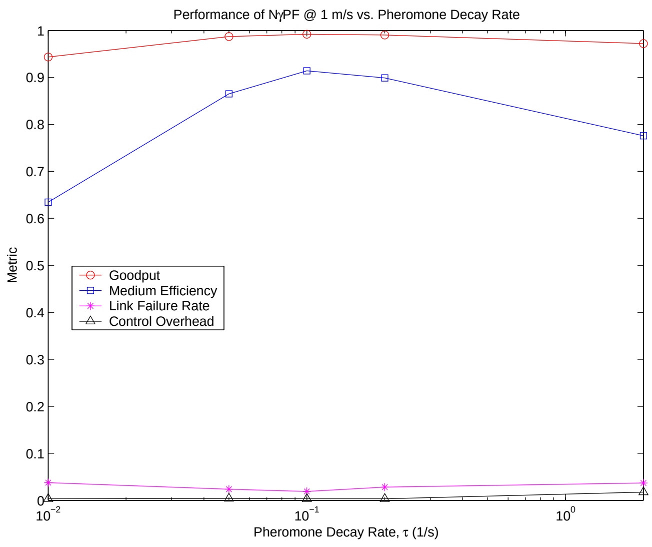

Figure 7.10: Performance of NγPF @ 1 m/s vs. Pheromone Decay Rate

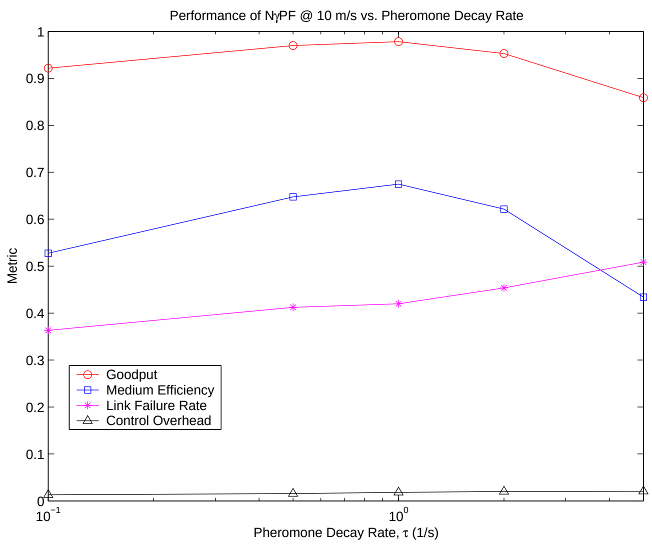

Figure 7.11: Performance of NγPF @ 10 m/s vs. Pheromone Decay Rate

7.4.2 τ^* Heuristic Analysis

The quality of the heuristic is determined by comparing its predictions to experimental data. The primary difficulty is to determine a good metric for the network correlation time or the network event rate (the inverse of the former). This work will use the link failure rate as a lower bound for the event rate (and thus an upper bound on the correlation time). Figure 7.12 takes the results reported in Figures 7.10 and 7.11 and compares them to the $\bar{\gamma}$PF τ^* heuristic.

Unfortunately the results are inconclusive. There is a clear difference between the heuristic and optimal values from simulation. The former predicts values roughly 15% lower that the latter. However, it is also true that the link failure rate overestimates the network correlation time. That is, the ideal network event rate is the path correlation time, which is lower than (and upper bounded by) the link lifetime [78]. Perhaps if the true path lifetimes could be determined (which would move the simulation points on the graph to the right) then the heuristic would

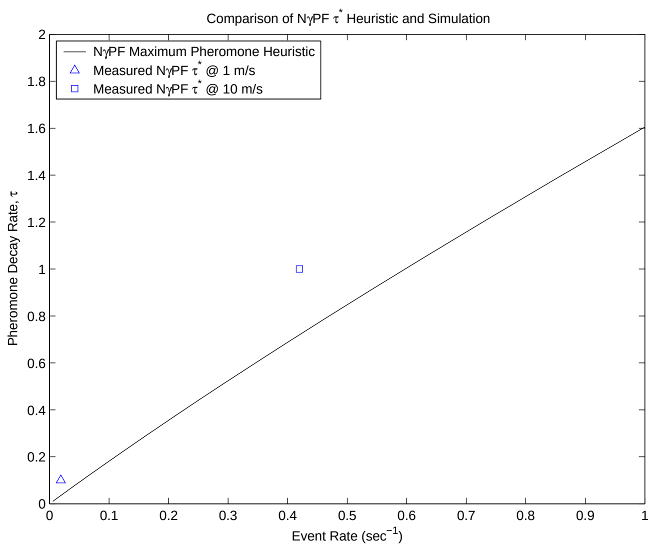

Figure 7.12: Comparison of NγPF τ^* Heuristic and Simulation

more closely match the simulation results. This value may be determined from traditional approaches by examining the rate that route errors are reported. However, since ReTermite uses probabilistic routing with automatic retransmission, it is difficult to measure when a route has changed significantly or truly broken.

While the heuristic is not perfect, it is reassuring to note that both the heuristic and optimal decay rate have a generally linear relation. An order of magnitude change in the node speed results in the same change in the optimal decay rate. Such a linear relationship is also seen in the τ^* heuristic for both γPF and $\bar{\gamma}$PF. It is most useful to compare the heuristic to node speed instead of correlation time. While this can be done for the experimental data, it is more difficult to do for the heuristic. It is impossible to compare node speed with link failure rate and then use it as an estimate of the event rate. The failure rate is itself dependent on the chosen decay rate. The path correlation time is independent of the pheromone decay rate; it is only dependant on the mobility and medium model. If this relationship could be found then a much more useful heuristic could be built.

7.5 Conclusion

The ReTermite swarm intelligent routing algorithm for mobile wireless ad-hoc networks is presented. Building upon the introduction of the Termite SI MANET routing protocol, it adds several features including source pheromone repel, the normalized γ pheromone filter, a route discovery flood, and a fast route repair mechanism. Many components are justified based on work presented in previous chapters. A comparison is made to the standard AODV routing protocol. The simulation tests are run over a wide range of node speeds in order to show how each algorithm is able to scale with respect to speed. ReTermite is shown to be superior in all of the tested metrics, especially at low speed. Using data to carry routing information turns out to be a very effective technique to reduce control overhead and to maintain routing information. However, the end-to-end delay suffers due to a fraction of packets which are difficult to deliver. This may be due to frequent route breaks, route discovery latencies, and other network effects. The determination of optimal parameters remains a difficulty. It is easy to show that optimal parameters exist, in this case the pheromone decay rate, however the available heuristics are not always effective or even tight. More work in this area is clearly requires. Ultimately, ReTemite is shown to deliver more packets with less overhead in more adverse conditions than AODV in a realistic medium access environment.

CHAPTER 8

CONCLUSION

8.1 Introduction

This chapter concludes the thesis. The original contributions to the state-of-the-art of the fields of swarm intelligence and routing in mobile wireless ad-hoc networks are first presented. This is follwed by a section on future work and further research opportunities which can continue the work presented here. The last section concludes the chapter and the thesis with some final remarks.

8.2 Original Contributions

A number of original contributions have been made to the state-of-the-art of the fields of swarm intelligence and ad-hoc routing over the course of this thesis. The swarm intelligent Termite routing algorithm has been developed. It was shown not only to work, but to work very well compared against the widely accepted standard in the field. An extensive testing of various pheromone update schemes revealed several insights into the nature of pheromone-based routing. Simultaneous pheromone decay is necessary in order to consistently maintain concurrent estimates of link utility. A continuous pheromone decay is also helpful in this regard. These tests confirmed some intuitions regarding the general influence of the pheromone decay rate parameter. They were the first of their type to be carried out in the field. The analytical model used to analyze the routing algorithm is not new, however it is extended beyond its original form in order to take into account stochastic packet arrival and pheromone utility models. The pheromone decay rate heuristics derived from the analysis are new to the field and they represent the first attempt to predetermine optimal operating values. The final effect of all of this development work is shown when a redesigned Termite is compared against AODV in a realistic simulation model. The performance of Termite is shown to be superior to AODV in all of the critical metrics, and shows promise for future development and further improvement in performance. The research into Termite is the most comprehensive of any other swarm intelligent MANET routing algorithms to date.

8.2.1 The Termite SI MANET Routing Algorithm

Termite fits very nicely into the swarm intelligence framework, and thus pushes the state-of-the-art in the same direction as those algorithms to have come before it. Termite offers many novel advancements over previous work including a pheromone/utility based metric, the use of random walk for route discovery, and local retransmission instead of local path repair.

One of these advancements is the use of a pheromone based metric which is also interpreted as path utility. ACO uses a pheromone based technique coupled with a local search parameter to determine next-hop probabilities. However the pheromone value does not have any real physical significance. ABC and AntNet calculate probabilities directly and do not use pheromone as an intermediate metric. The first routing algorithm to use only pheromone to route packets is ARA. Developed simultaneously, Termite uses this same approach, however with the use of the normalized γ pheromone filter, the pheromone value may be interpreted as representing the estimated path utility to a destination through a given link. This interpretation brings the concept of pheromone routing full circle back to the original concepts behind distance vector routing. This is exemplified by CAF, where pheromone is not even mentioned and costs are measured directly, similarly to how Termite does now (but without joint pheromone decay on all links).

Termite also introduces the option of using random walks for route discovery. This is not a new concept by any means, however it has some favorable characteristics especially when taken in conjunction with promiscuous mode receiving. A much larger number of nodes can be reached with a single route request message and many fewer broadcasts are needed compared to a full flood of the network. There are of course some issues, such as the route requests needed to cover a given area or the probability of finding a particular destination. But the tradeoffs can be favorable.

ARA introduces the use of local retransmission instead of local repair, which Termite then further develops. Traditional routing algorithms choose to either initiate local route requests when a link breaks, or to notify the source of a traffic stream to initiate a new route request, or both. Searching and notification can incur a large time delay and a heavy load on the network resources. When a link breaks, Termite removes that link from its pheromone table and then simply retransmits the packet on a remaining link based on the new pheromone distribution. The only delay is in recognizing that the link has broken, which any other algorithm must suffer as well. This technique does well, but ultimately the end-to-end delay of Termite is increased because more difficult-to-deliver packets arrive at their destination. Since no route errors are sent, the number of control packets is reduced. No additional route request or route error packets must be issued to find a new route to a destination.

8.2.2 Evaluation of Pheromone Accounting

The most common pheromone update method is the pheromone filter, or perhaps the classic pheromone filter. Oftentimes these techniques are used blindly for any application simply because they are used everywhere else. In an effort to verify the utility of the pheromone filter, the analysis of Termite includes the empirical testing of eleven different pheromone update methods. This is the first known direct comparison and serves to reveal several aspects of the Termite system. These include the need for joint pheromone decay (concurrent freshness maintenance), the need to update with variable amounts of pheromone, the difficulty of independent link utility estimation (such as the Box and Averaging filters), and the relative performance of routing with respect to situations of perfect information and no information. These different approaches have never been compared side-to-side and the analysis of Termite was able to show that the pheromone based approaches fare the best.

8.2.3 Pheromone Decay Rate Heuristics

Two novel pheromone decay rate heuristics have been introduced by this work. They are called Maximum Pheromone and Filter Cutoff. Earlier publications only described the decay rate as a system parameter and determined its optimal value either by simulation or simply by choosing a value that seemed to give good results. The work here is among the first to automatically identify good operating points. The heuristics were shown to be lower bounds, although they are not especially tight. There is also some difficulty in determining exactly what the network correlation time is; no good definition exists.

Maximum Pheromone Heuristic

The Maximum Pheromone heuristic determines the pheromone decay rate by first calculating the maximum expected pheromone level on a link given the local packet arrival rate and pheromone brought by each packet. This is coupled with the expected time during which the statistics of the incoming pheromone are the same, known as the network correlation time. The heuristic calculates the decay rate necessary to decay the expected pheromone on a link to an insignificant level, ie. equal to the pheromone threshold, within the network correlation time. The equation to determine the amount of pheromone on a link will depend on the type of pheromone filter being used. This work has developed expressions for both the γ pheromone filter and the normalized γ pheromone filter.

Filter Cutoff Heuristic

The Filter Cutoff heuristic interprets the pheromone filter updating method as a literal filter, and adjusts the filter cutoff to determine a pheromone decay rate. The filter's frequency response is determined by the packet arrival rate and the decay rate. By adjusting the decay rate to meet a particular definition of a cutoff frequency, a heuristic is defined to allow the pheromone or link utility estimation to change at a certain rate.

8.2.4 ReTermite

The development of Termite through its final incarnation as ReTermite also brings with it a number of original contributions to the field. Perhaps the most mundane of them is a simple comparison to AODV. The simulations were able to show that ReTermite has superior performance in many critical performance metrics across a wide range of node mobility. ReTermite includes a number of novel enhancements to a SI MANET routing algorithm. These include true continuous pheromone decay and source pheromone repel. These ideas were known previously in related forms, however ReTermite brings them all together and shows how they can be effectively applied to help solve the routing problem.

Comparison with AODV

A simulation-based comparison was made between ReTermite and AODV. The results show that ReTermite performs extremely well as compared to the control. It is able to deliver more packets faster and with less control traffic across a wide range of mobility. This gives very strong motivation to continue its development. Probabilistic routing algorithms should be reconsidered in a more positive light as more is known of how to control them.

Continuous Pheromone Decay

Termite has progressed the idea of continuous pheromone decay first introduced by ARA. Originally pheromone was decayed periodically, which was based on simple computer models of ant behavior. ARA then mentions the idea of continuous pheromone decay based on packet interarrival times, although does not give any implementation details. Termite is the first to give these details, but then also to include pheromone decayed on observation times; true continuous pheromone decay. In this way, if a packet arrives after a long silence, the pheromone levels used to calculate the forwarding distribution will reflect that wait.

Pheromone decay is a real time process in biology. There is good reason to accurately model this phenomenon in a computer implementation as well. Since pheromone in part measures the freshness or timeliness of information (since it is decayed based on time; a time based metric) it must be kept as up-to-date as possible. It is therefore decayed not only between packet arrivals, but also when it is measured in order to forward packets.

Source Pheromone Repel

Temite is the first SI MANET routing algorithm to use source pheromone repel (SPR). While MACO mentions it, the descriptions never give specific implementation details. Instead of being repelled by the pheromone of another colony, a packet is repelled by its own source pheromone. This provides a strong degree of directivity to the routing process. SPR also helps to mitigate looping since this implies a turn towards the source. ReTermite is simplified by removing any sort of packet identifiers or logs.

8.3 Future Work

No research is really ever done. There still exist a number of opportunities for future work. Clearly there are many areas for improvement with the Termite algorithm itself. This includes issues from packet forwarding to route discovery to pheromone updating and everything in between. More generally, there are also open issues regarding the development of better parameter heuristics for optimal performance, reformulation of the network routing problem, and new directions for artificial intelligence and machine learning algorithms. The following sections will describe open problems in each area in detail.

8.3.1 Termite

There are several assumptions upon which Termite is based that should be addressed. The most pressing of these is the requirement of symmetric, bidirectional communications links between nodes. The use of hop count as an optimization metric obscures this assumption, because as long as the link is bidirectional, it will be symmetric in that metric. However there are many more metrics that are not symmetric, such as throughput, delay, or energy consumption. Based on previous work, as well as the experience gained from designing and testing Termite as it stand today, the following recommendations are made for a more general SI MANET routing algorithm.

Packet Forwarding

The packet forwarding equation should be kept, and packets should continue to be forwarded probabilistically based on link pheromone concentrations. There is no evidence to show that an Equation 8.1 style packet forwarding scheme (as used originally in AS, and later in PERA and ANSI),

$$p_{i,d} = \frac{\tau_{i,d}^{\alpha}\eta_{i,d}^{\beta}}{\sum_j \tau_{j,d}^{\alpha}\eta_{j,d}^{\beta}} \tag{8.1}$$

can provide superior performance over something like Equation 8.2 (used in Termite and ARA).

$$p_{i,d} = \frac{(P_{i,d}+K)^F}{\sum_j (P_{j,d}+K)^F} \tag{8.2}$$

This is despite the fact that the former uses two metrics, both global pheromone and local visibility, and the latter only global pheromone. An answer to this question may come over time, however no direct comparison has been done to date.

Source Pheromone Repel SPR has been shown to provide a strong improvement in results when applied to the routing problem. While this technique cannot be considered in general, the structure of the routing problem allows its use; the destination is generally in a direction away from the source.

Route Discovery

Alternative route discovery methods should be considered. The random walk method currently being used by Termite works well, however a flooding or gossiping scheme should also be considered due to their higher destination discovery rate. Flooding schemes will find the destination if it exists in the network since they cover the entire network. Random walks may fail depending on their path. It may be worthwhile to consider directed random walks.

Pheromone Update

True continuous pheromone decay as well as the normalized γ pheromone filter should continue to be used. These techniques closely mirror older methods from reinforcement and statistical learning, giving a strong argument for their effectiveness and a strong basis for their analysis. They elegantly fit into the theory of the algorithm, and are easily implemented as well.

The CAF method for pheromone updates should be used while regarding the network sampling theory. The routing metric can only be updated upon the arrival of a routing agent. Thus the destination of a packet flow should take care to send routing agents to the source at a rate sufficient to update the metric fast enough based on the rate of change of the network environment. In fact, the pheromone decay rate and the send rate of route agents would be linked, since they are both dependant on the correlation time of the network. If this new algorithm is to be a purely on-demand protocol, then these route agents only need to be sent to nodes from which there is an active traffic flow. This could be coupled with information from the transport layer to determine which nodes should receive route updates. Due to the nature of the asymmetric environment, the use of route agents is unavoidably necessary. This will lead to the proliferation of control traffic in volatile networks.

Pheromone Updates with Data Traffic

A strong feature of Termite is the use of data packets to actively update the network. This substantially increases the sampling rate of the network which can lead to improved performance. Previous work in both the SI and MANET communities relies only on control packets to do this job. Termite should continue to do this according to the CAF route update. Each node will record the average cost of moving from a neighbor to itself. This average can be calculated with a simple one-tap averaging IIR filter with time dependant weights (as per continuous pheromone decay). Unlike the current implementation of Termite, local travel costs can be averaged over all packets arriving on a link, not just packets from a certain source.

If Termite should be developed assuming bidirectional symmetric links, then the algorithm should implement a limited pheromone stack on each data packet, similar to AntNet or CAF. This will allow the packets to disseminate more information while not adding too much control overhead.

Route Repair

The Termite route repair scheme should be kept. This includes local retransmission in case of link failure. Previous work attempts either a new route discovery, a route error message, or both in order to manage this situation. There is no evidence to support the use of the latter over the former. In fact, the former continues to make more sense in the context of minimizing end-to-end delay and control overhead.

8.3.2 Parameter Heuristics

While the question of optimally determining Termite's parameter values may never be fully answered due to its complexity, more effort should be invested in maturing the heuristics. A complete list of the controllable parameters include the pheromone sensitivity, the pheromone threshold, the pheromone repel constant, and the pheromone decay rate. Of primary concern is the determination of the latter, either on a global, per node, or per link per node level, which has a very intuitive effect on routing performance; the pheromone decay rate determines how long network information is retained. The global effect of the other parameters is not as clear, but should also be studied; the pheromone sensitivity controls the tradeoff between environment exploration and exploitation, the pheromone repel constant adjusts the liklihood of backtracking or looping, and the pheromone threshold defines a "significant" amount of pheromone. The attempts made in this work, while pioneering, are admittedly quite simple. The analytical model developed in Chapter Six is tractable, however it honestly has little relationship to a real ad-hoc network. Simulation tests are therefore recommended; at least the results will have some basis in (simulated) reality.

8.3.3 Networking with Linear Systems

After the pheromone filter perspective pheromone updating was developed, perhaps a system of linear systems approach could be taken in order to model the routing process. Such a perspective may also find some traction with ideas from controls systems, where the use of dynamic feedback loops is quite common. The idea that network routing can be formulated as a question of linear systems seems to be novel and may lead to some significant insight.

8.3.4 Artificial Intelligence

The strong relationship between artificial intelligence and swarm intelligence has already been demonstrated. The two fields can undoubtedly still learn a lot from each other. Problems usually solved with AI techniques such as reinforcement learning or neural networks might benefit from a swarm intelligent approach. This would include game playing, control problems, non-linear functions, and especially those variants in which the learning environment changes over time. On the other hand, swarm intelligence may benefit from the more established methodologies of artificial intelligence and machine learning. This is especially true with regards to statistics (or statistical learning).

8.4 Final Remarks

This thesis has presented a novel approach to routing in mobile wireless ad-hoc networks. Models based on social insect behavior are applied to create a probabilistic solution. The approach is able to deliver more packets with less overhead that traditional approaches. And that's all I have to say about that.

APPENDIX A

PHEROMONE UPDATE SIMULATION RESULTS

The following figures are oriented in the following way. The upper left shows the results for a node speed of 1 m/s, the upper right shows 5 m/s, the lower left shows 10 m/s, and the lower right shows 20 m/s.

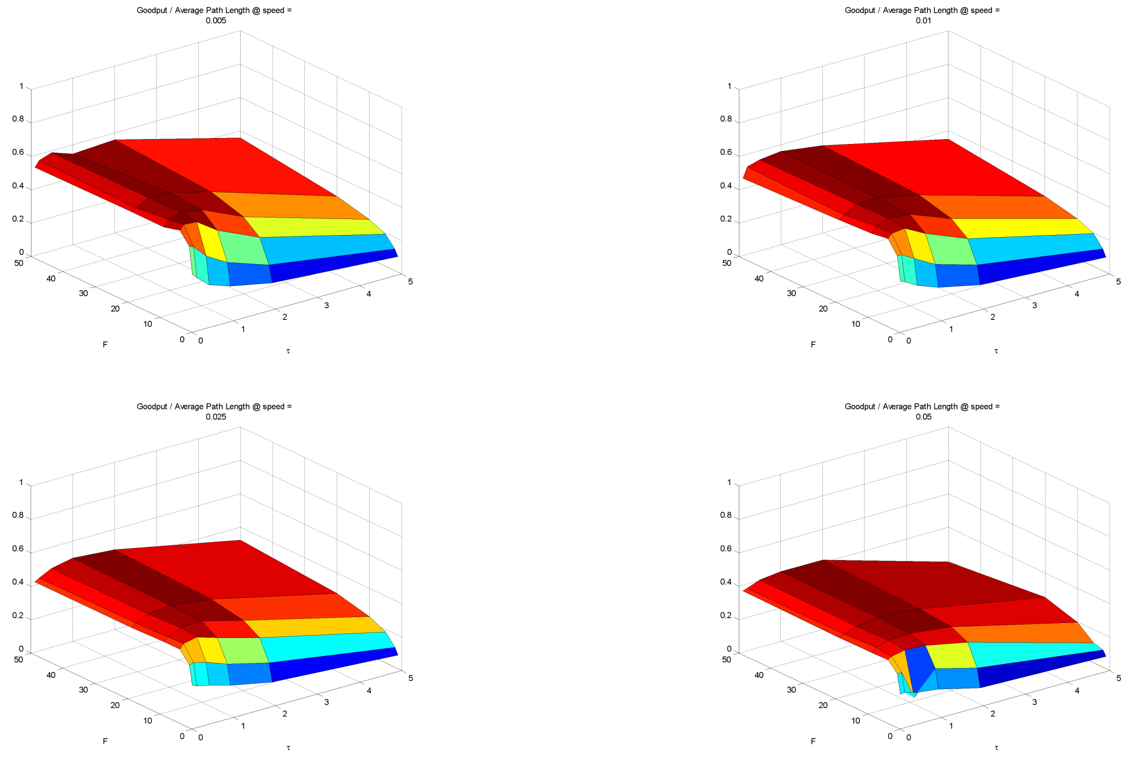

Figure A.1: Delivery Efficiency of γPF vs. F vs. τ (50 Nodes)

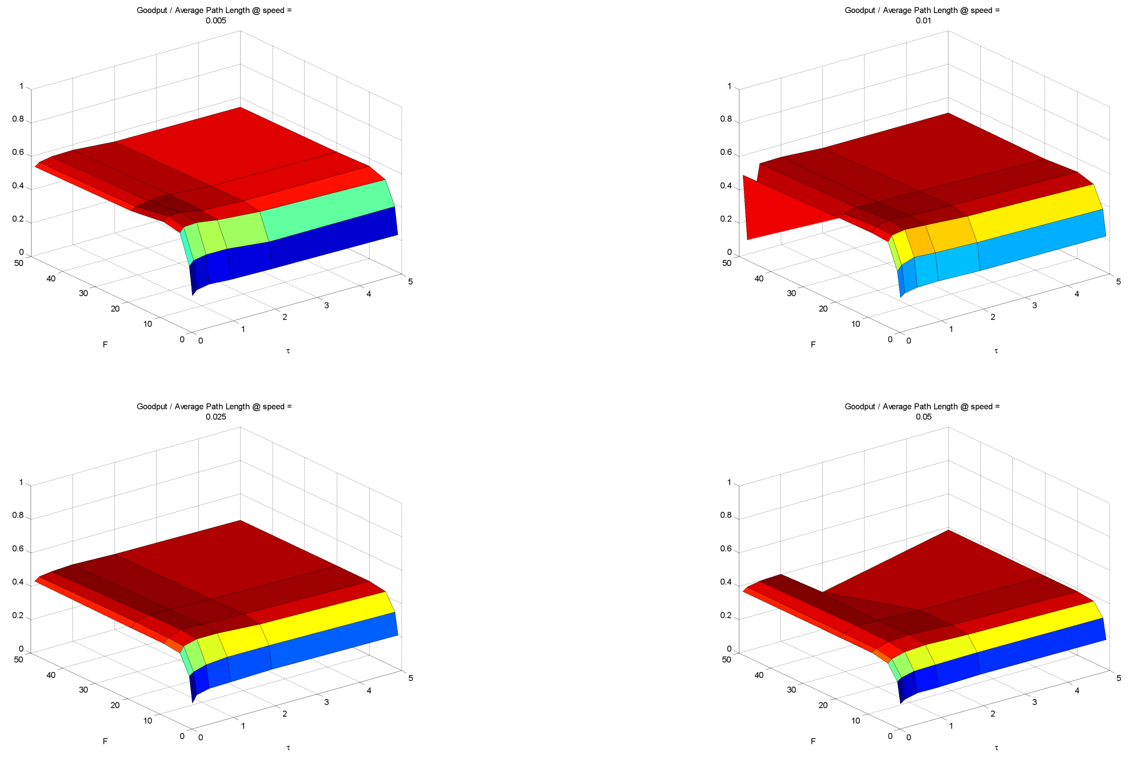

Figure A.2: Delivery Efficiency of $\bar{\gamma}$PF vs. F vs. τ (50 Nodes)

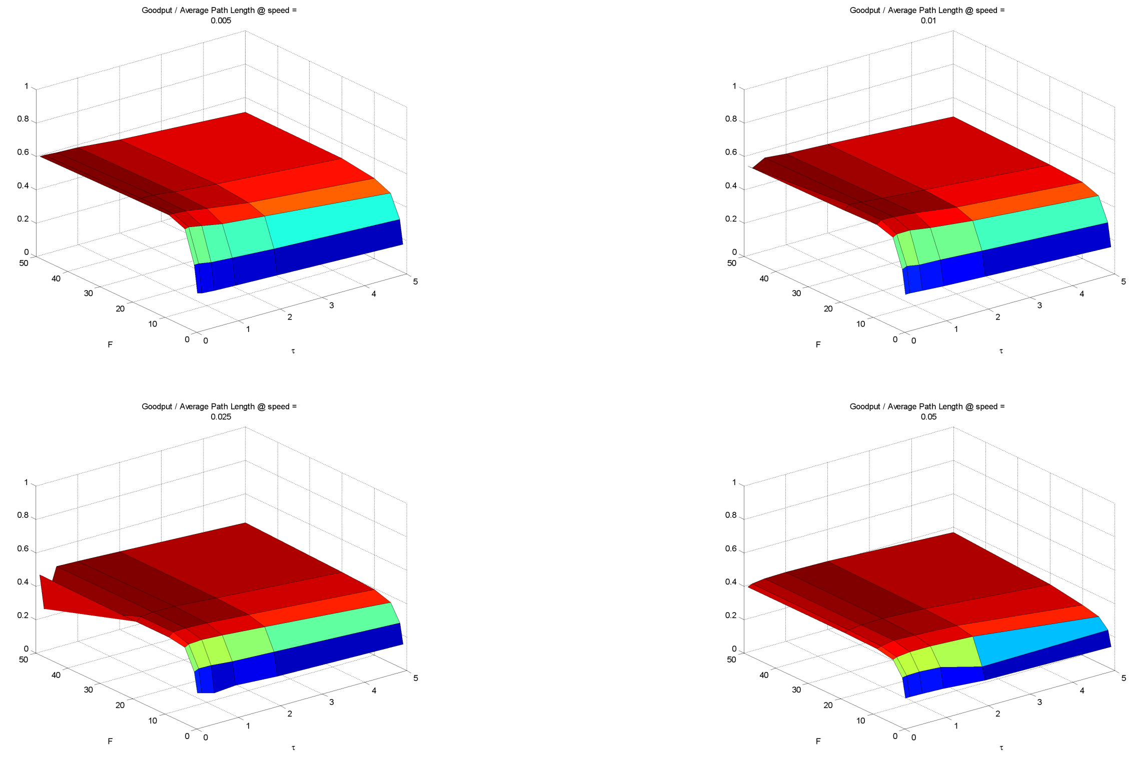

Figure A.3: Delivery Efficiency of pBF vs. F vs. τ (50 Nodes)

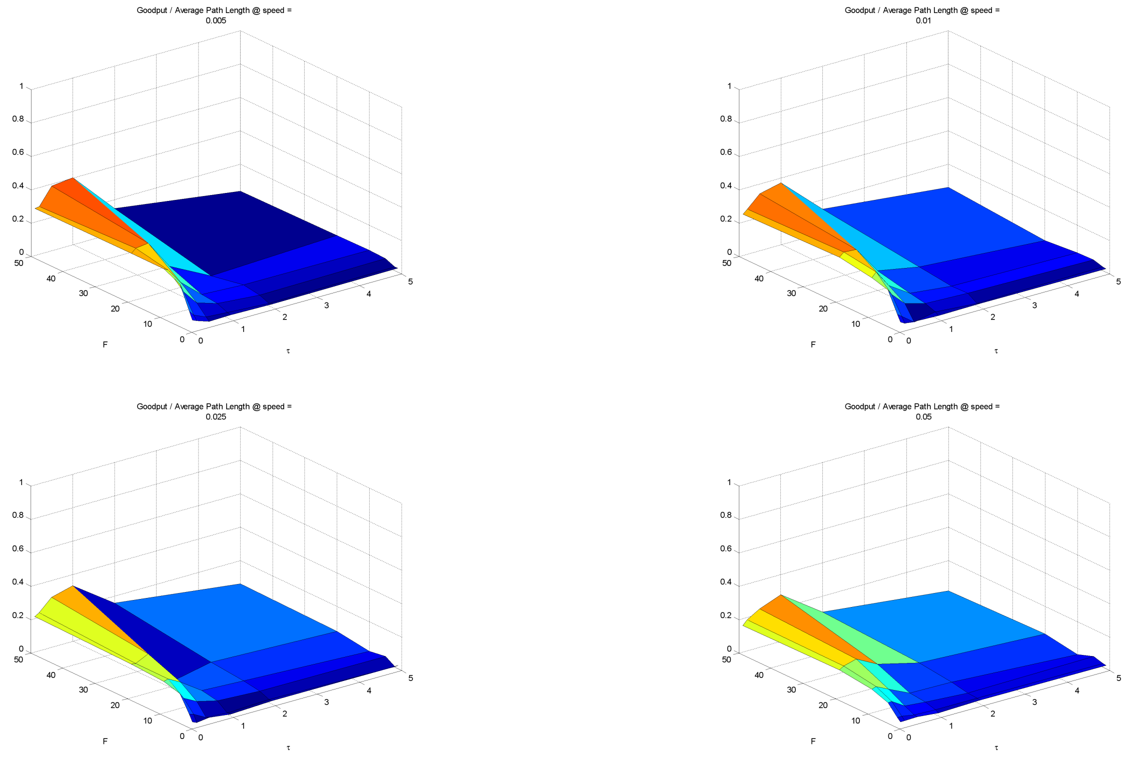

Figure A.4: Delivery Efficiency of Classic PF vs. F vs. τ (50 Nodes)

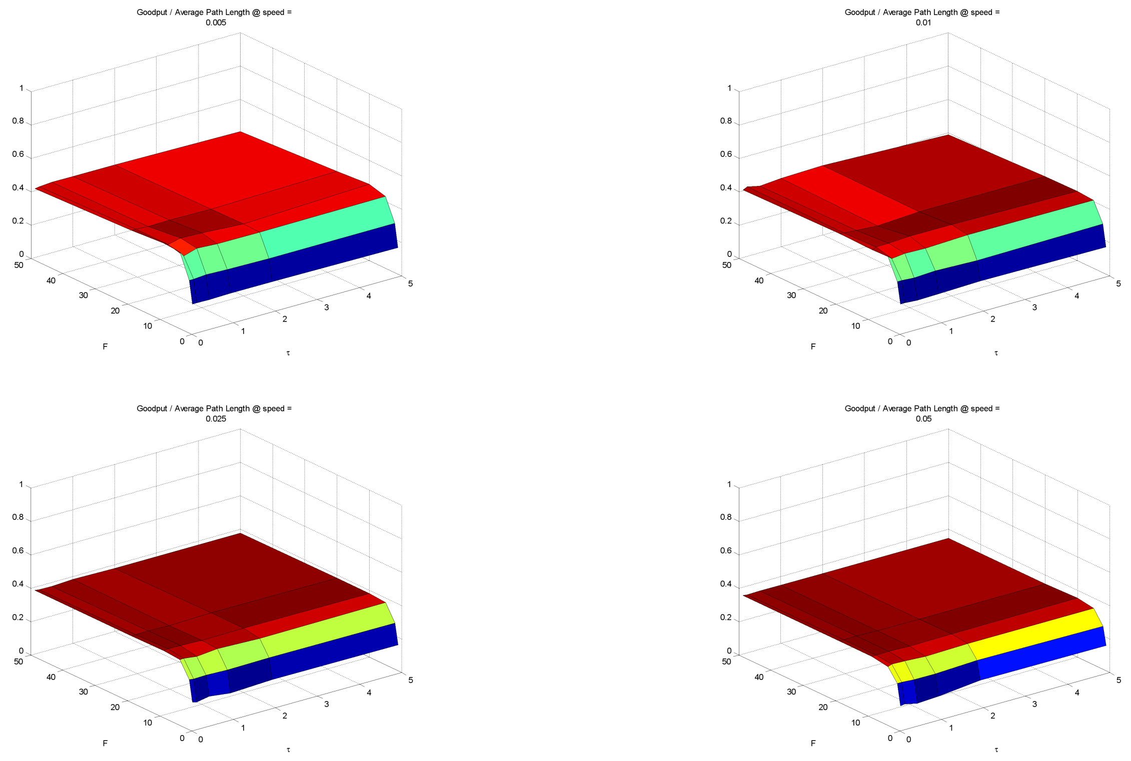

Figure A.5: Delivery Efficiency of Box Filter vs. F vs. τ (50 Nodes)

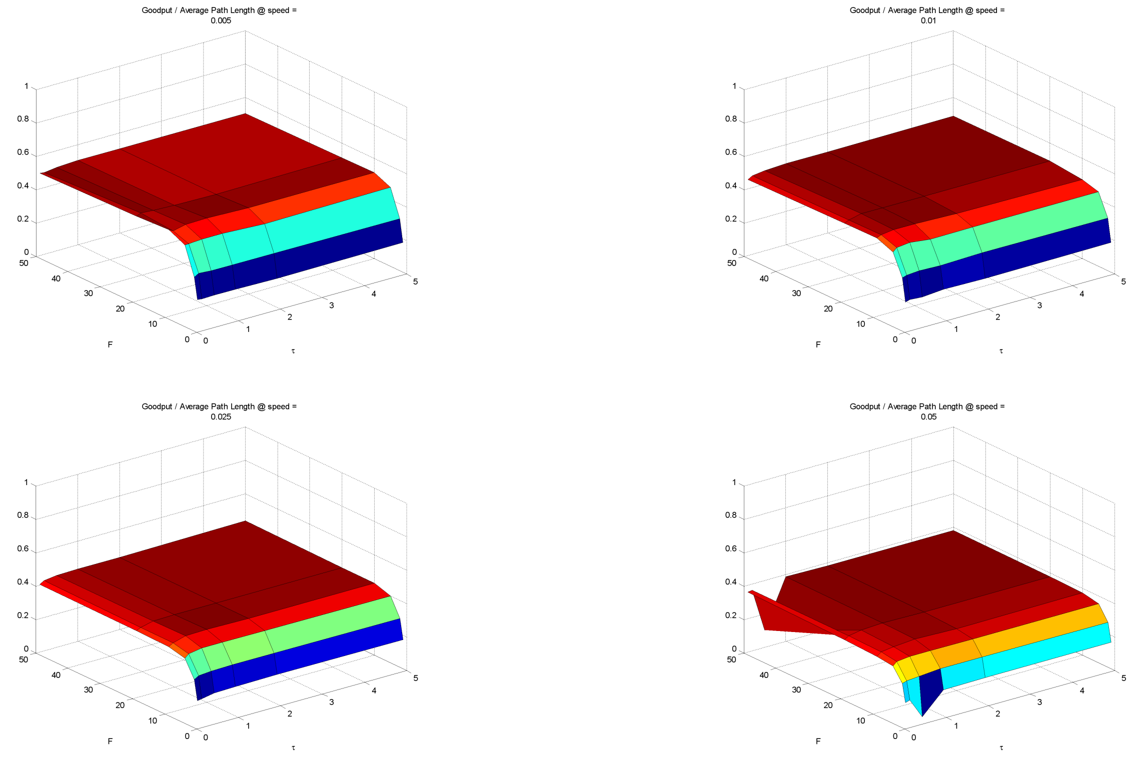

Figure A.6: Delivery Efficiency of IIR Averaging Filter vs. F vs. τ (50 Nodes)

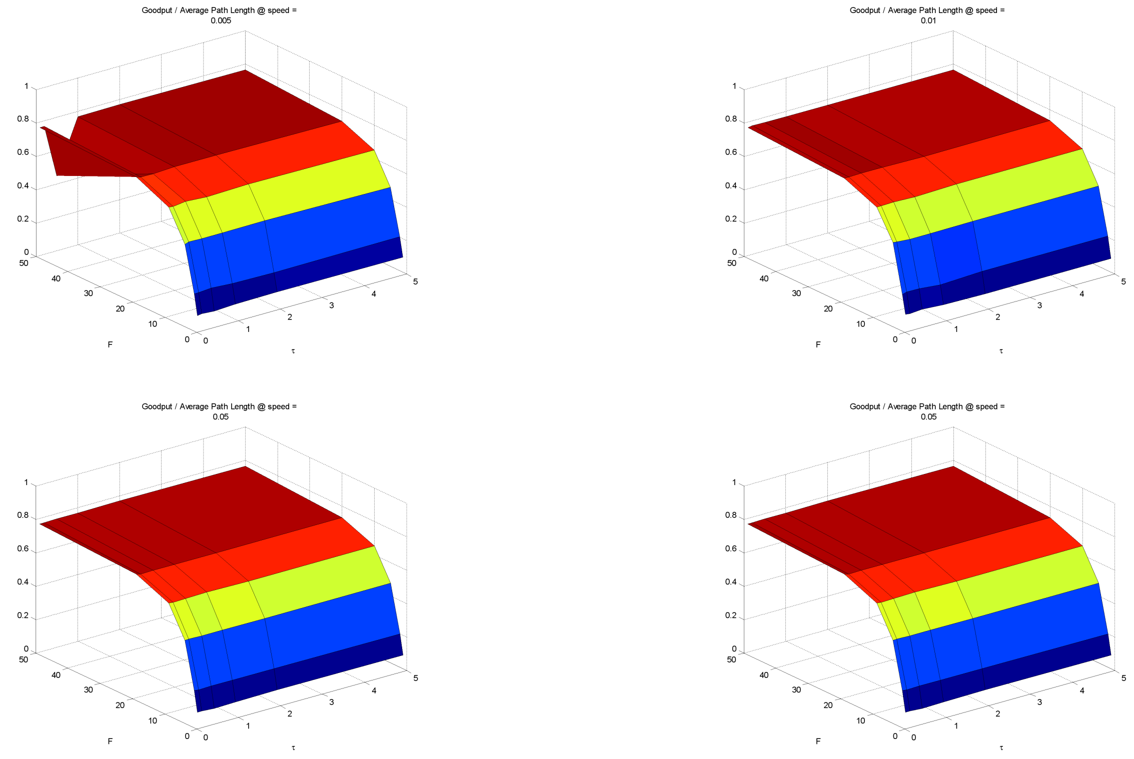

Figure A.7: Delivery Efficiency of Oracle vs. F vs. τ (50 Nodes)

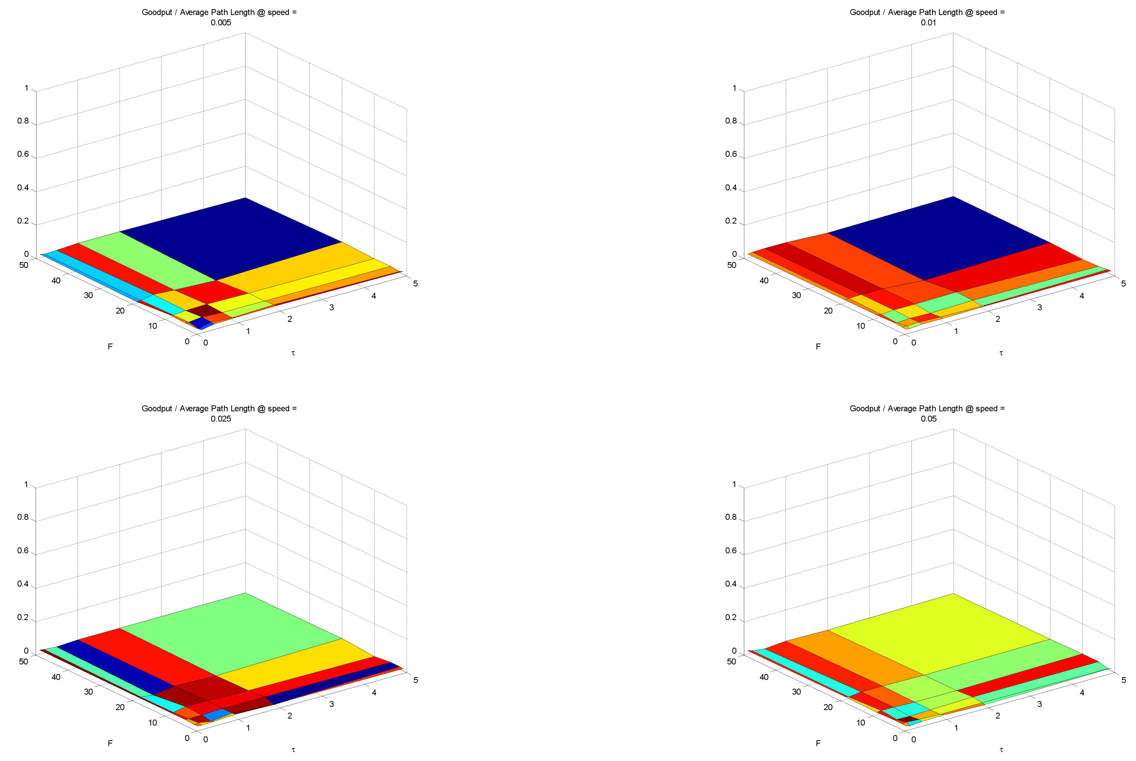

Figure A.8: Delivery Efficiency of Random Routing vs. F vs. τ (50 Nodes)

BIBLIOGRAPHY

[1] E. Bonabeau, M. Dorigo, G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.

[2] Levy, S., *Artificial Life: A Report from the Frontier Where Computers Meet Biology*, Vintage Books, 1993.

[3] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.

[4] P. Grassé, *La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositermes Natalensis et Cubitermes. La Théorie de la Stigmergie: Essai d'Interpretation du Comportement des Termites Constructeurs*, Insect Societies, 1959.

[5] http://www.stigmergicsystems.com/stig_v1/stigrefs/article2.html

[6] D. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*, Vintage Books, 1989.

[7] M. Resnick, *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*, Bradford Books, 1997.

[8] B. Holldobler, E. O. Wilson, *The Ants*, Belknap Press, 1990.

[9] http://iridia.ulb.ac.be/ mdorigo/ACO/RealAnts.html

[10] M. Dorigo, G. Di Caro, L. M. Gambardella, *Ant Algorithms for Discrete Optimization*, Artificial Life, Vol. 5, No. 2, 1999.

[11] M. Dorigo, V. Maniezzo, A. Colorni, *The Ant System: Optimization by a Colony of Cooperating Agents*, IEEE Transacations on Systems, Man, and Cybernetics, 1996.

[12] M. Dorigo, L. Gambardella, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, IEEE Transactions on Evolutionary Computation, 1997.

[13] M. Sipper, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, Springer-Verlag, 1997.

[14] S. Wolfram, *A New Kind of Science*, Wolfram Media, 2002.

[15] http://cell-auto.com/

[16] X. Tu and D. Terzopoulos, *Artificial Fishes: Physics, Locomotion, Perception, Behavior*, SIGGRAPH '94, 1994.

[17] http://www.red3d.com/cwr/boids/

[18] http://www.swarm-bots.org/

[19] http://leurre.ulb.ac.be/index2.html

[20] http://www.research.ibm.com/massive/tdl.html

[21] http://www.cs.washington.edu/research/jair/volume4/kaelbling96a-html/node59.html

[22] K. Chan, *A Simple Learning Algorithm for the Traveling Saleman Problem*, Physical Review, 1997.

[23] N. Meuleau, M. Dorigo, *Ant Colony Optimization and Stochastic Gradient Descent*, *Artificial Life 8*, 2002.

[24] http://upload.wikimedia.org/wikipedia/en/d/da/Tree_graph.png

[25] K. M. Sim, W. H. Sun, *Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 33, No. 5, September 2003.

[26] C. Hedrick, *Routing Information Protocol (RIP) version 1*, http://www.faqs.org/rfcs/rfc1058.html, 1988.

[27] G. Malkin, *Routing Information Protocol (RIP) version 2*, http://www.faqs.org/rfcs/rfc2453.html, 1998.

[28] R. Bellman, *On a Routing Problem*, Quarterly of Applied Mathematics, 1958.

[29] L. Ford Jr., D. Fulkerson, *Maximal Flow Through a Network*, Canadian Journal of Mathematics, 1956.

[30] J. Moy, *Open Shortest Path First version 2*, http://www.faqs.org/rfcs/rfc2328.html, 1998.

[31] E. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik, Vol. 1, 269-271, 1959.

[32] S. Corson, J. Macker, *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*, http://www.faqs.org/rfcs/rfc2501.html, 1999.

[33] http://grouper.ieee.org/groups/802/11/

[34] http://www.ietf.org/html.charters/manet-charter.html

[35] C. Perkins, P. Bhagwat, *Routing over Multihop Wireless Network of Mobile Computers*, SIGCOMM '94, 1994.

[36] P. Jacquet, P. Mühlenthaler, T. Clausen, A. Laouiti, A. Qayyum, L. Viennot, *Optimized Link State Routing Protocol for Ad-Hoc Network*, IEEE INMIC, 2001.

[37] http://hipercom.inria.fr/olsr/

[38] http://www.ietf.org/rfc/rfc3626.txt

[39] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, *A Performance Comparison of Multi-Hop Wireless Ad-Hoc Network Routing Protocols*, The Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98), 1998.

[40] P. Perkins, E. Royer, *Ad-hoc On-demand Distance Vector*, Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999.

[41] C. Perkins, E. Belding-Royer, S. Das, *Ad-hoc On-demand Distance Vector (AODV) Routing*, http://moment.cs.ucsb.edu/AODV/aodv.html, 2004.

[42] C. Perkins, E. Belding-Royer, S. Das, *Ad-hoc On-demand Distance Vector (AODV) Routing*, http://www.faqs.org/rfcs/rfc3561.html, 2003.

[43] D. Johnson, D. Maltz, *Dynamic Source Routing in Ad-hoc Wireless Networks*, SIGCOMM '96, 1996.

[44] D. Johnson, D. Maltz, Y. Hu, *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, http://www.monarch.cs.rice.edu/internet-drafts/draft-ietf-manet-dsr-10.txt, 2004.

[45] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz, *Ant-Based Load Balancing In Telecommunications Networks*, *Adaptive Behavior*, 1996.

[46] D. Subramanian, P. Druschel, J. Chen, *Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks*, Proceedings of the International Joint Conference on Artificial Intelligence, 1997.

[47] G. Di Caro, M. Dorigo, *Mobile Agents for Adaptive Routing*,Technical Report, IRIDIA/97-12, Université Libre de Bruxelles, Belgium, 1997.

[48] B. Barán, R. Sosa, *A New Approach for AntNet Routing*, Proceedings of the Ninth International Conference on Computer Communications and Networks, 2000.

[49] M. Heusse, D. Snyers, S. Guérin, P. Kuntz, *Adaptive Agent-Driven Routing and Local Balancing in Communication Networks*, ENST de Bretagne Technical Report RR-98001-IASC, 1997.

[50] G. Varela, M. Sinclair, *Ant Colony Optimization for Virtual-Wavelength-Path Routing and Wavelength Allocation*, Proceedings of the 1999 Congress on Evolutionary Computation, 1999.

[51] K. M. Sim, W. H. Sun, *Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, Vol. 33, No. 5, September 2003.

[52] M. Heissenbüttel, T. Braun, *Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks*, Kommunikation in Verteilten Systemen (KiVS), 2003.

[53] J. Baras, H. Mehta, *A Probabilistic Emergent Routing Algorithm for Mobile Ad-hoc Networks*, WiOpt '03: Modeling and Optimization in Mobile, Ad-Hoc, and Wireless Networks, 2003.

[54] M. Günes, U. Sorges, I. Bouazizi, *ARA - The Ant-Colony Based Routing Algorithm for MANETs*, Proceedings of the ICPP Workshop on Ad Hoc Networks, 2002.

[55] M. Günes, M. Kähmer, I. Bouazizi, *Ant Routing Algorithm (ARA) for Mobile Multi-Hop Ad-Hoc Networks - New Features and Results*, The Second Mediterranean Workshop on Ad-Hoc Networks, 2003.

[56] M. Roth, S. Wicker, *Termite: Ad-hoc Neworking with Stigmergy*, The Second Mediterranean Workshop on Ad-Hoc Networks, 2003.

[57] S. Rajagopalan, C. Shen, *A Routing Suite for Mobile Ad-hoc Networks using Swarm Intelligence*, unpublished, 2004.

[58] Z. Haas, *A New Routing Protocol for the Reconfigurable Wireless Networks*, ICUPC '97, 1997.

[59] V. Ramasubramanian, Z. Haas, E. Sirer, *SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad-hoc Networks*, The Fourth ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc), 2003.

[60] P. Samar, M. Pearlman, Z. Haas, *Hybrid Routing: The Pursuit of an Adaptable and Scalable Routing Framework for Ad Hoc Networks*, *The Handbook of Ad-Hoc Wireless Networks*, CRC Press, 2003

[61] K. Amin, J. Mayes, A. Mikler, *Agent-Based Distance Vector Routing*, Mobile Agents for Telecommunication Applications (MATA), 2001.

[62] S. Marwaha, C. K. Tham, D. Srinjvasan, *Mobile Agents based Routing Protocol for Mobile Ad-Hoc Networks*, IEEE Globecom 2002, 2002.

[63] A. Bieszczad, B. Pagurek, T. White, *Mobile Agents for Network Mangement*, Vol. 1 No. 1, IEEE Communications Surveys, 1998.

[64] A. Tanenbaum, *Computer Networks*, Prentice Hall, 2003.

[65] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, Y. Minsky, *Bimodal Multicast*, ACM Transactions on Computer Systems, 1999.

[66] L. Li, Z. Haas, J. Halpern, *Gossip-Based Ad-hoc Routing*, The 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2002.

[67] B. Krishnamachari, S. Wicker, R. Béjar, *Phase Transition Phenomena in Wireless Ad-Hoc Networks*, IEEE Globecom, 2001.

[68] J. Boyan, M. Littman, *Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach*, *Advances in Neural Information Processing Systems*, Morgan Kaufmann, 1993.

[69] A. V. Oppenheim, R. W. Schafer, J. R. Buck, *Discrete-Time Signal Processing*, 2nd Edition, Prentice Hall, 1999.

[70] C.-K. Toh, *A Novel Distributed Routing Protocol To Support Ad-Hoc Mobile Computing*, IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications, 1996.

[71] Y. Chen, E. G. Sirer, S. Wicker, *On Selection of Optimal Transmission Power for Ad-Hoc Networks*, The 36th Hawaii International Conference on System Sciences (HICSS 2003), 2002.

[72] V. Ramasubramanian, R. Chandra, D. Mossé, *Providing a Bidirectional Abstraction for Unidirectional Ad-Hoc Networks*, INFOCOM 2002, 2002.

[73] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, J. Jetcheva, *A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols*, Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking, 1998.

[74] http://www.opnet.com/

[75] J. Yoon, M. Liu, B. Noble, *Random Waypoint Considered Harmful*, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), 2003.

[76] M. Roth, S. Wicker, *Performance Evaluation of Pheromone Update in Swarm Intellignet MANETs*, under review, 2005.

[77] A. J. Hayter, *Probability and Statistics for Engineers and Scientists*, 2nd Edition, Duxbury Press, 2001.

[78] P. Samar, Z. Haas, *Updating Strategies for Proactive Routing Protocols in Ad-Hoc Wireless Networks*, , 2003.

[79] M. Roth, S. Wicker, *Asymptotic Pheromone Behavior in Swarm Intelligent MANETs: An Analytical Analysis of Routing Behavior*, Sixth IFIP IEEE International Conference on Mobile and Wireless Communications Networks (MWCN), 2004.

[80] M. Roth, S. Wicker, *ReTermite: Lessons in Swarm Intelligent Routing*, under review, 2005.

[81] C. Perkins, E. Royer, S. Das, M. Marina, *Performance Comparison of Two On-Demand Routing Protocols for Ad-Hoc Networks*, IEEE Personal Communications, February 2001.

[82] E. Nowicki, *Soldier Ant: Revolution Against The Hive Mind*, 2004.