Università della Svizzera italiana
Faculty of Informatics

Adaptive Routing in Ad Hoc Wireless Multi-hop Networks

Frederick Ducatelle

Dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy

Lugano, Switzerland, May 2007

Doctoral Committee:

Professor Luca M. Gambardella, research advisor (IDSIA)
Professor Antonio Carzaniga, academic advisor (USI)
Professor Amy L. Murphy, internal member (USI)
Professor Patrick Thiran, external member (EPFL)

Academic year 2006-2007

The research for this thesis was carried out at the *Istituto Dalle Molle di Studi sull'Intelligenza Artificiale* (IDSIA), under the supervision of Luca Maria Gambardella. This work was partially supported by the Future & Emerging Technologies unit of the European Commission through project "BISON: Biology-Inspired techniques for Self Organization in dynamic Networks" (IST-2001-38923) and by the Swiss Hasler Foundation through grant DICS-1830.

To Lili and Delphine

Abstract

Ad hoc wireless multi-hop networks (AHWMNs) are communication networks that consist entirely of wireless nodes, placed together in an ad hoc manner, i.e. with minimal prior planning. All nodes have routing capabilities, and forward data packets for other nodes in multi-hop fashion. Nodes can enter or leave the network at any time, and may be mobile, so that the network topology continuously experiences alterations during deployment. AHWMNs pose substantially different challenges to networking protocols than more traditional wired networks. These challenges arise from the dynamic and unplanned nature of these networks, from the inherent unreliability of wireless communication, from the limited resources available in terms of bandwidth, processing capacity, etc., and from the possibly large scale of these networks. Due to these different challenges, new algorithms are needed at all layers of the network protocol stack.

We investigate the issue of adaptive routing in AHWMNs, using ideas from artificial intelligence (AI). Our main source of inspiration is the field of Ant Colony Optimization (ACO). This is a branch of AI that takes its inspiration from the behavior of ants in nature. ACO has been applied to a wide range of different problems, often giving state-of-the-art results. The application of ACO to the problem of routing in AHWMNs is interesting because ACO algorithms tend to provide properties such as adaptivity and robustness, which are needed to deal with the challenges present in AHWMNs. On the other hand, the field of AHWMNs forms an interesting new application domain in which the ideas of ACO can be tested and improved. In particular, we investigate the combination of ACO mechanisms with other techniques from AI to get a powerful algorithm for the problem at hand.

We present the AntHocNet routing algorithm, which combines ideas from ACO routing with techniques from dynamic programming and other mechanisms taken from more traditional routing algorithms. The algorithm has a hybrid architecture, combining both reactive and proactive mechanisms. Through a series of simulation tests, we show that for a wide range of different environments and performance metrics, AntHocNet can outperform important reference algorithms in the research area. We provide an extensive investigation of the internal working of the algorithm, and we also carry out a detailed simulation study in a realistic urban environment. Finally, we discuss the implementation of ACO routing algorithms in a real world testbed.

Acknowledgements

I would like to thank in the first place my supervisor, Luca Maria Gambardella, who has given me the opportunity to work on this project and has assisted me in any way possible. Then, I want to thank my colleague, Gianni A. Di Caro, who has been like a second supervisor to me, and has been an invaluable help throughout this project. Other people who have been of scientific importance for my work on this thesis include (in order of appearance) John Levine, who has first introduced me to the field of ant colony optimization, Marco Dorigo, who initiated this interesting research field and also brought me in contact with IDSIA, Patrick Thiran, who brought us in contact with the field of ad hoc networks, and Martin Roth, who gave me the opportunity to work with him at Deutsche Telekom Laboratories in Berlin. Furthermore, I would like to thank all the people at IDSIA for all scientific input, in the form of presentations, discussions, etc..

Apart from the scientific input, of equal importance to me has been the personal support I have received throughout the years that I have spent on this thesis. I want to thank a number of people without whom this work would never have been completed. In the first place, I thank my wife, Liliana Carrillo, and my daughter, Delphine, who arrived just in time to give me the final support I needed to finish the work. Then I want to thank my family, my mother, my father and my sisters, Caroline and Barbara, who have been, from distance, a great support. Next, I thank Alex Graves and Matteo Gagliolo, partners in crime during all those years in Lugano. Friends back home, Pieter Vandenbossche and family, Kobe Lootens, Dieter Herregodts, Pieter Orbie, Frederik Demilde, Katelijne Carbonez, Tom Ghyselinck, and many others. And all other friends, wherever they are, Matthew Davies, Elaine Boyd, Jesper Salomon, Olga Chrysanthopoulou, Ola Svenson, Nikos Mutsanas, Shane Legg, Maciej Kurant, etc..

Contents

List of Figures

List of Tables

Glossary

ABC Ant-Based Control, 62
ABR Associativity-Based Routing, 43
ACK ACKnowledgement, 30
ACO Ant Colony Optimization, 49
ACS Ant Colony System, 61
ADRA Ant-based Distributed Routing Algorithm, 72
AHWMN Ad Hoc Wireless Multi-hop Network, 15
AI Artificial Intelligence, 59
AMQR Ant colony based Multi-path QoS-aware Routing, 72
ANSI Ad hoc Networking with Swarm Intelligence, 71
AODV Ad-hoc On-demand Distance Vector routing, 33
ARA Ant-colony-based Routing Algorithm, 71
ARP Address Resolution Protocol, 172
AS Ant System, 60
AS Autonomous System, 49
ASR Adaptive Swarm-based Routing, 68
ATM Asynchronous Transfer Mode, 68

BGP Border Gateway Protocol, 35, 49

CAF Co-operative Asymmetric Forward, 68
CBR Constant Bit Rate, 113
CE Cross-Entropy, 69
CGSR Clusterhead Gateway Switched Routing, 41
CTS Clear To Send, 30

DCF Distributed Coordination Function, 30
DSDV Destination-Sequenced Distance-Vector routing, 33
DSR Dynamic Source Routing, 33
DTL Deutsche Telekom Laboratories, 24
DTN Delay Tolerant Network, 28

ETT Expected Transmission Time, 44
ETX Expected Transmission Count, 44

FSR Fisheye State Routing, 42

GA Genetic Algorithm, 69
GM Gauss-Markov mobility model, 115
GPS Global Positioning System, 42
GSM Global System for Mobile Communications, 15

IETF Internet Engineering Task Force, 23

LAN Local Area Network, 44
LEACH Low Energy Adaptive Clustering Hierarchy, 42
LoS Line of Sight, 151
LQSR Link Quality Source Routing, 38
LSA Link State Advertisement, 55

MABR Mobile Ants Based Routing, 44
MAC Medium Access Control, 29
MANET Mobile Ad hoc NETwork, 22
MCL Mesh Connectivity Layer, 38
MIT Massachusetts Institute of Technology, 24
MMAS MAX-MIN Ant System, 61
MPLS Multiprotocol Label Switching, 174
MPR Multi-Point Relays, 36

NoC Networks-on-Chip, 69

OLSR Optimized Link State Routing, 33
OSPF Open Shortest Path First, 36, 49

PERA Probabilistic Emergent Routing Algorithm, 70

QoS Quality of Service, 55

RBA Routing By Ants, 68
RERR Route ERRor message, 38
RIP Routing Information Protocol, 35, 49
RL Reinforcement Learning, 73
RREP Route REPly message, 38
RREQ Route REQuest message, 38
RTS Request To Send, 30
RTT Round Trip Time, 32
RWP Random WayPoint, 27

SELA Stochastic Estimator Learning Automata, 68
SINR signal-to-interference-and-noise ratio, 101
SMS Short Messaging Service, 148

TCP Transmission Control Protocol, 32
TORA Temporally-Ordered Routing Algorithm, 40
TSP Traveling Salesman Problem, 60

UDP User Datagram Protocol, 32
UWB Ultra Wide Band, 15

VoIP Voice-over-IP, 148

WAN Wide Area Network, 44
WDM Wavelength-Division Multiplexing, 69
WiFi Wireless-Fidelity, 15
WiMax Worldwide Interoperability for Microwave Access, 15
WLAN Wireless Local Area Network, 15
WMN Wireless Mesh Network, 23

ZRP Zone Routing Protocol, 34

Chapter 1

Introduction

In this thesis, we investigate the development of adaptive routing algorithms for ad hoc wireless multi-hop networks using techniques from artificial intelligence, and in particular from swarm intelligence. The aim of the thesis is two-fold. From the point of view of networking, we want to take advantage of promising techniques from the area of artificial intelligence to develop a strong algorithm to solve the challenging task of routing in ad hoc wireless multi-hop networks. From the point of view of artificial intelligence, we want to show how the basic ideas from swarm intelligence can be adapted to work well for a realistic, very challenging dynamic problem.

In what follows, we first give a general introduction to the problem that is investigated in the thesis. Then we list the contributions of the thesis, and finally, we provide an outline of its content.

1.1 General introduction

One of the most important developments in recent years in the field of telecommunication networks is the increased use of wireless communication. A wide range of different wireless technologies and standards have been developed, including Wireless-Fidelity [3] (WiFi, IEEE 802.11), Bluetooth [33] (IEEE 802.15.1), Zigbee [13] (IEEE 802.15.4), Ultra Wide Band [4, 282] (UWB, IEEE 802.15.3), Worldwide Interoperability for Microwave Access [5] (WiMax, IEEE 802.16), etc.. These technologies are being made available on an ever increasing number of devices such as laptops, mobile phones, palmtops, etc., allowing them to connect to a variety of different networks. This explosive growth has made wireless communication networks one of the most important areas of research in computer science.

Within the field of wireless networks, one can make a distinction between infrastructured networks and infrastructureless networks [227]. In infrastructured networks, a fixed, wired backbone infrastructure is available, and all communication is directed over this backbone. This approach is followed in traditional

wireless network systems such as the Global System for Mobile Communications (GSM) [222] and Wireless Local Area Networks (WLANs) [110]. In infrastructureless networks, such a backbone does not exist, and wireless devices communicate directly with one another through point-to-point connections. An important aspect in infrastructureless wireless networks is the use of multi-hop data forwarding: since direct point-to-point connections are only possible between wireless nodes that are in immediate radio range of each other, communication between nodes that are remote from each other needs to be supported by other nodes in the network, which function as relay points, effectively substituting the missing wired backbone infrastructure. Infrastructureless wireless networks are also referred to as ad hoc networks, since they can be deployed on-the-fly, without the need for prior planning (as opposed to infrastructured networks, where considerable efforts and investments are needed before communication can take place). In this thesis, we will use the term ad hoc wireless multi-hop networks (AHWMNs).

Different types of AHWMNs exist. Examples are mobile ad hoc networks, wireless mesh networks and sensor networks. Mobile ad hoc networks (MANETs) [227] are networks that are made up of a set of homogeneous mobile devices. These devices communicate exclusively through wireless connections, normally using a single, omnidirectional antenna. All nodes in the network are equal, and there are no designated routers, meaning that all nodes can serve both as end points of data communication and as intermediate relay points or routers. Wireless mesh networks (WMNs) [16] differ from MANETs mainly because they are more heterogeneous. They consist of mesh client nodes, which are similar to MANET nodes, and mesh router nodes, which are usually less mobile, have more resources (e.g. more powerful processors, more battery power, etc..), and support a variety of different wireless technologies. The availability of mesh routers allows the creation of a structured organization and can greatly improve the applicability and the capacities of the network. Finally, sensor networks [17] are AHWMNs that consist of wireless sensor nodes. Each sensor node is a small unit containing one or more sensors, a small processing unit and a wireless radio. Problems specific to sensor networks stem from the fact that sensor nodes are small and have very limited capacities, that usually vast numbers of nodes are deployed, and that data traffic patterns show certain characteristic regularities.

In this thesis, we focus on the problem of routing in AHWMNs. Routing is the task of constructing and maintaining the paths that connect remote source and destination nodes of data. This task is particulary hard in AHWMNs due to issues that result from the particular characteristics of these networks. A first important issue is the fact that AHWMNs are dynamic networks. This is due to their ad hoc nature: connections between nodes in the network are set up in an unplanned manner, and are often changed while the network is in use. Especially when mobile nodes are used, such changes can take place continuously. An AHWMN routing algorithm should be adaptive in order to keep up with such dynamics. A second issue is the unreliability of wireless communication. Data and control packets can easily get lost during transmission, especially when mobile nodes are involved, and when multiple transmissions

take place simultaneously and interfere with each other. A routing algorithm should be robust with respect to such losses. A third issue is caused by the often limited capabilities of the AHWMN nodes. There are limitations in terms of network bandwidth, node processing power, memory, battery power, etc.. It is therefore important for a routing algorithm to work in an efficient way. Finally, a last important issue is the network size. With the ever growing numbers of portable wireless devices, many AHWMNs are expected to grow to very large sizes. Routing algorithms should be scalable to keep up with such evolutions.

To solve the challenging problem of routing in AHWMNs, we apply methods from the field of artificial intelligence. Specifically, we are interested in the use of techniques from swarm intelligence (SI) [34, 151] and ant colony optimization (ACO) [83, 87]. SI is the branch of artificial intelligence that is focused on the design of algorithms inspired by the collective behavior of social insects and other animal societies. ACO is a subset of SI that takes its inspiration from the foraging behavior of ants living in colonies. It has been observed from experiments that ants in a colony are able to find the shortest path between their nest and a food source, even though this task is well outside the capabilities of each individual ant. The key to this colony level shortest path behavior is the use of a volatile chemical substance called pheromone. Ants going between their nest and a food source leave a trail of pheromone behind, and also preferentially move in the direction of higher pheromone intensities. Shorter paths can be completed quicker and more frequently by the ants, and therefore get marked with a higher pheromone intensity. These paths then attract more ants, which in turn increases their pheromone level, until there is a convergence of the majority of the ants onto the shortest paths. The ants completing paths can be seen as repetitive samples of possible paths, while the laying and following of pheromone results in a collective learning process guided by implicit reinforcement of good solutions. ACO algorithms inspired by this shortest path behavior have been developed in recent years for many different problems. The main areas of application have been on the one hand the field of static combinatorial optimization problems (see e.g. [84, 107, 169]), and on the other hand the field of routing in telecommunication networks (see e.g. [70, 71, 235]).

ACO algorithms for routing in telecommunication networks differ substantially from more traditional routing algorithms. They gather routing information through the repetitive sampling of possible paths between source and destination nodes using artificial ant packets. Probabilistic distance vector tables, called pheromone tables, fulfill the role of pheromone in nature, with artificial ants being forwarded along them in a hop-by-hop way using stochastic forwarding decisions. Also data packets are forwarded stochastically using similar tables, resulting in automatic data load balancing. ACO routing algorithms boast some of the properties that we have outlined earlier as being important for AHWMN routing, such as adaptivity and robustness. This is mainly due to the continuous exploration of the network by stochastically forwarded ant probing packets. However, existing ACO routing algorithms have mainly been designed for wired networks. They are not able to deal with the high levels of change that are present in AHWMNs, nor do they offer the efficiency needed to

work in AHWMNs. For example, the same repetitive sampling of paths using ant agents that ensures continued adaptivity and robustness, can easily generate high levels of overhead that can clutter the network.

In this thesis, we investigate how ideas from ACO routing can be used efficiently to build an adaptive routing algorithm for AHWMNs. Our aim with this work is twofold. On the one hand, we want to develop a routing algorithm for AHWMNs that contains the advantages of adaptivity and robustness that are present in ACO routing. In this way, we want to obtain an algorithm that can deliver a better service than currently existing algorithms for these networks. On the other hand, we want to use AHWMNs as a testbed for the use of SI and ACO: it forms an interesting problem domain to see how principles from SI and ACO can be applied to a realistic and challenging dynamic optimization problem. This will lead to the development of new practices in the use of ACO and ACO routing.

One important aspect in the work presented here is the hybridization between different technologies. Since ACO routing as it exists now is not fit to work well in AHWMNs, we combine it with techniques from other fields. On the one hand, we include techniques that are present in existing AHWMN routing algorithms, such as e.g. flooding and local repair mechanisms. On the other hand, we also consider the integration with other methods from artificial intelligence. In particular, we use ideas from information bootstrapping [252] and dynamic programming [26], which in artificial intelligence are important for the field of reinforcement learning [252], and in networking have been used as the basis for the development of distance vector routing algorithms [27] such as RIP [123]. This novel combination of the techniques from ACO, which are primarily based on repeated sampling and are therefore related to Monte Carlo methods, with techniques from dynamic programming gives a fruitful interaction and allows to build a powerful algorithm.

1.2 Contributions of this thesis

The contributions of the work presented in this thesis can be considered from two different points of view. On the one hand, the thesis contains contributions for the field of computer networking, and on the other hand for the field of artificial intelligence.

From a networking point of view, we propose a new algorithm for routing in AHWMNs, based on ideas from artificial intelligence. The algorithm shows a novel way of combining reactive and proactive routing. It incorporates ideas from ACO routing, from dynamic programming, and from traditional AHWMN routing algorithms. In simulation tests over a wide range of different scenarios, we show that it can outperform existing state-of-the-art algorithms. We also carry out an analysis of the internal working of the algorithm, investigating the individual contribution of each of the mechanisms applied by the algorithm. Finally, we carry out tests using a detailed simulation of an urban scenario. We show how such a simulation can be carried out in an efficient way, and

investigate what the effect is of the urban environment on the performance of our routing algorithm compared to the current state-of-the-art.

From an artificial intelligence point of view, we show how mechanisms from ACO and SI can be adapted to work well for a realistic dynamic optimization problem. We apply ACO routing in a reactive way, and combine it with techniques from traditional AHWMN routing algorithms and techniques from information bootstrapping and dynamic programming. Especially the latter is relevant in the field of machine learning and more specifically reinforcement learning. In this research area, algorithms already exist that combine sampling techniques such as the ones used in ACO with elements from information bootstrapping. However, these combinations are done in a way that is completely different from what we present here. Our novel way of integrating elements from these important approaches to learning is dictated by the needs of the extremely challenging, dynamic environment formed by AHWMNs, and shows a way to form a powerful learning algorithm that can operate in such environments. Finally, we also describe a system for the implementation of ACO routing algorithms on a hardware implementation of an AHWMN.

1.3 Outline of the thesis

The work presented in this thesis is organized in the following chapters:

Chapter 2 - Ad hoc wireless multi-hop networks. This chapter gives an overview of the field of AHWMNs. The aim is to provide the technical background needed to understand the problem area we address in this thesis. While the focus of the chapter is mainly on routing, as this is the topic of this thesis, we also cover issues related to other aspects of the network protocol stack, in as far as they are relevant for our work.

Chapter 3 - Adaptive routing and learning. In this chapter, we give an introduction to adaptive routing. Here, the aim is to provide background knowledge about the techniques we use in the thesis. We first discuss adaptive techniques that are used in traditional routing algorithms. Then, we give a detailed description of ACO and ACO routing, which form the main source of inspiration for the work in this thesis. Finally, we also discuss the field of reinforcement learning, in order to provide a unifying framework, based on machine learning, in which we place different approaches to adaptive routing.

Chapter 4 - AntHocNet: an adaptive routing algorithm for ad hoc wireless multi-hop networks. This chapter is dedicated to the description of AntHocNet, the algorithm for adaptive routing in AHWMNs that was developed for this thesis. The algorithm is based on ideas from ACO routing, but also contains elements from more traditional routing algorithms and from other techniques from the field of machine learning.

Chapter 5 - An evaluation study of AntHocNet. In this chapter, we present results of an extensive set of simulation studies in which we evaluate the AntHocNet routing algorithm. The chapter is divided in two parts, with the first part dedicated to a comparison with current state-of-the-art AHWMN routing algorithms, and the second part focused on investigating the internal working of AntHocNet itself.

Chapter 6 - Simulation of an urban scenario. Here, we describe a different set of simulation tests, carried out in an urban environment. We first describe how we got a detailed simulation of such an environment in an efficient way, and then evaluate the behavior of AntHocNet in this environment.

Chapter 7 - Towards the implementation of adaptive routing in AHWMNs. In this final chapter, we discuss a system for the implementation of ACO routing algorithms in real AHWMN testbeds. The system is based on a set of kernel modules and a routing daemon running in user space.

Chapter 2

Ad hoc wireless multi-hop networks

The work presented in this thesis is focused on ad hoc wireless multi-hop networks (AHWMN). This chapter gives an overview of existing research on this kind of networks. Section 2.1 provides a definition for AHWMNs. Section 2.2 describes which different types exist. Section 2.3 lists a number of networking issues in AHWMNs, related to different levels of the network protocol stack. Finally, we dedicate the whole of section 2.4 to routing in AHWMNs, since that is the main focus of this thesis.

2.1 A definition of ad hoc wireless multi-hop networks

In recent years, a growing number of devices are getting equipped with networking capabilities. Many of these devices are mobile and communicate using a variety of wireless technologies, such as Bluetooth, WiFi, etc., which allow them to connect to existing telecommunication networks and to each other. If these devices also support routing, they can forward data for each other. One can then combine a number of such devices with minimal planning to form a network. Such a network would be an ad hoc wireless multi-hop network. Formally, we can say that an AHWMN is a network consisting of nodes that communicate solely through wireless connections, in which data can be forwarded over multiple hops, and which is at least partly deployed in ad hoc manner. Ad hoc deployment entails that little or no planning is needed, and that changes in the network (such as adding, moving or removing nodes) can be done with minimal extra work.

From the above definition, it is clear that there are some substantial differences between AHWMNs and traditional telecommunication networks. Probably the most important difference is that the topology of an AHWMN is not

the result of careful planning, but instead emerges from the placement of the nodes: there is a link between nodes if they are in range of each other's wireless radio signal. As a consequence, the topology is essentially dynamic: it can be extended by adding new wireless nodes, reduced by removing nodes, or changed in a continuous way if some of the nodes are mobile. Such dynamic behavior presents an important new challenge in networking technology, since in traditional networks, changes happen relatively infrequently. A second important difference is that AHWMNs rely exclusively on wireless links. This means that data transport is less reliable, and that there is less available bandwidth. Thirdly, AHWMNs are usually highly decentralized, lacking hierarchy or central control. This makes it more difficult to optimize the use of network resources (e.g., 2.3.3 explains how decentralized medium access control algorithms can severely reduce available bandwidth). Fourth, the nodes of an AHWMN often have limited resources. This can apply to memory, processing power, battery power supply, etc.. Finally, due to the increasing availability of wireless technology, AHWMNs are often expected to grow to very large sizes, making scalability an important issue.

As a result of these challenges, algorithms that were designed for traditional telecommunication networks often perform badly in AHWMNs. New algorithms are therefore needed at all layers of the network protocol stack. In this thesis, we concentrate on the routing layer.

2.2 Types of ad hoc wireless multi-hop networks

In network research, work is being done on a number of different, related types of wireless networks, which can all be classified as AHWMNs. In this section, an overview is given of these different kinds of networks, and of the similarities and differences between them. First, mobile ad hoc networks are described, since these were the first AHWMNs that received a lot of attention in the literature. Next, we present wireless mesh networks. These form a more general class of AHWMNs, of which mobile ad hoc networks could in fact be considered a subclass. After that, sensor networks are described, which is an application specific subclass of AHWMNs. We close this section with a short discussion on the similarities and differences between the different types of AWHMNs, and the consequences for this thesis.

2.2.1 Mobile ad hoc networks

Mobile ad hoc networks (MANETs) [227] are networks in which all nodes are mobile and communicate exclusively via wireless connections. Usually, the nodes are equipped with a single, omnidirectional wireless antenna. There is no fixed infrastructure in the network, and there is no hierarchy: all nodes are in principle equal, and can function both as end points of data communication, and as routers, forwarding data for each other in multi-hop fashion. One can think of a group of users carrying wifi enabled devices such as mobile phones, pda's,

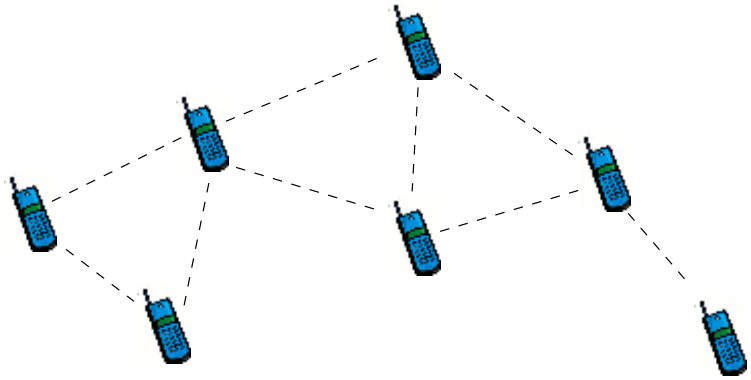


Figure 2.1: An example of a MANET built up of mobile phones. The dashed lines symbolize the wireless links.

laptops, etc., moving in a specific area and forming a dynamic wireless network among them. See for example the MANET made up of mobile phones depicted in figure 2.1, where the dashed lines symbolize the wireless links.

As a consequence of the above mentioned properties, MANETs are dynamic, flat, fully decentralized networks without central control or overview. This gives rise to a number of tough challenges for networking algorithms. MANET algorithms should be highly adaptive to the ever changing environment. They should be robust in order to deal with unreliable wireless transmissions. They should work in a fully distributed way. Finally, they should be efficient in their use of the limited network resources, such as bandwidth, battery power in the mobile nodes, etc..

The idea for MANETs stems from research into DARPA packet radio networks [142, 145]. Starting from the publication of the Destination-Sequenced Distance-Vector routing algorithm [211] in 1994, MANETs were the first type of AHWMNs to be investigated. The specific challenges that are encountered in these networks have called the attention of many researchers and have made this a very active research area. Also, the Internet Engineering Task Force (IETF) has set up a MANET working group to guideline standardizations. However, when it comes to implementation, the MANET challenges have proven to be very hard to deal with, so that there is now also a growing interest in AHWMNs with less mobility and more hierarchy and organization, such as the wireless mesh networks described in 2.2.2.

2.2.2 Wireless mesh networks

Wireless mesh networks (WMNs) [16] consist of two types of nodes: mesh clients and mesh routers. Mesh clients are equivalent to MANET nodes: they are mobile, and usually communicate through one wireless interface, which is normally

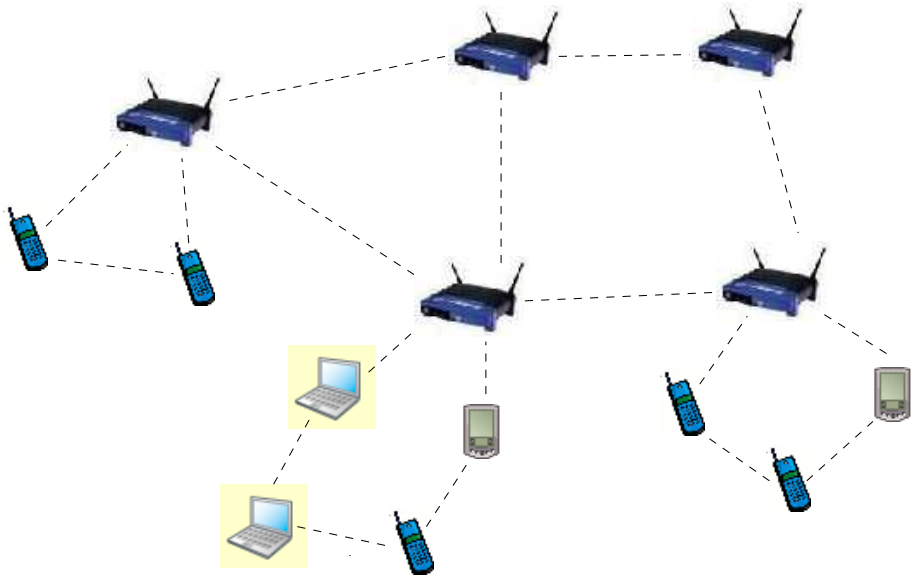


Figure 2.2: An example of a WMN, in which five static wireless nodes (in this example, these are WiFi access points) act as mesh routers and a number of heterogeneous mobile devices play the role of mesh clients.

an omnidirectional antenna. Like MANET nodes, they can serve both as end points of data traffic and as routers. Mesh routers, on the other hand, are less mobile or even static, and are usually equipped with various wireless devices, supporting different technologies. They are usually more powerful devices than the mesh clients, and often run on external power supply rather than on battery power. The aim of the mesh routers is to form a wireless backbone infrastructure for the WMN. An example of a WMN is given in figure 2.2: a small group of static wireless nodes function as mesh routers, while a larger number of heterogeneous mobile devices play the role of mesh clients.

The use of a more or less static backbone of mesh routers gives important advantages compared to MANETs. First, it gives some stability and organization to the network, which allows better exploitation of network resources. For example, data traffic can be routed primarily over the backbone nodes, which are normally more powerful and have higher bandwidth, alleviating the task of the mesh clients. Second, the fact that the mesh routers usually support a variety of different wireless communication technologies allows easy integration of heterogeneous devices and networks. Finally, the mesh routers partly solve the problem of battery power usage.

The mentioned advantages make WMNs easier to implement than MANETs. Consequently, an increasing number of mesh network implementation projects are being started. These include projects from academic research, such as the

Roofnet [30] experiment of the Massachusetts Institute of Technology (MIT), projects by businesses, such as the Magnets [148] project of Deutsche Telekom Laboratories (DTL), and even spontaneous efforts by independent enthusiasts, such as the olsr.freifunk.net experiment in Berlin [9]. These networks have large differences in the way they were set up, their structure, the devices that are used, etc.. For example, Roofnet and olsr.freifunk.net are both totally unplanned networks that only consist of randomly placed WiFi access points. Magnets on the other hand has a planned backbone of five high power routers that are connected via directed wireless antenna's, providing connectivity for a high number of randomly placed, less powerful devices around them. A detailed description of the Magnets project will be given in subsection 7.1.1.

2.2.3 Sensor networks

Sensor networks [17] are AHWMNs that consist of wireless sensor nodes. Those are small devices equipped with one or more sensors, some small processing capacity, and a radio transmitter. The aim of sensor networks is to deploy a large number of such sensor devices to measure a certain phenomenon. This can be geological activity, human body functioning, etc.. By forming a multi-hop network among them, the sensor nodes have a means to send the data they have measured to a "sink" node, where they can be processed. The fact that the network is ad-hoc allows to set it up with minimal planning. One can for example throw sensors from a boat into the sea so that they can form a network at the bottom, or drop them from a plane.

Sensor networks come with their own specific challenges. First of all, they are usually very large networks, possibly consisting of thousands of nodes or more, so that algorithms need to scale well. Next, the sensor nodes are normally designed to be very cheap, light devices. This means that they have very limited resources for storage, processing and transmission, so that highly efficient algorithms are needed. This problem is acerbated by the fact that sensor batteries can often not be replaced (e.g., when the sensors are thrown at the bottom of the ocean). Moreover, the use of cheap, low power radio technology also means that communication is highly unreliable and irregular. For example, changing radio ranges can give rise to unidirectional connections [290]. So algorithms have to be robust and should be able to deal with unidirectional links. Another issue in sensor networks is that their topology is usually very dynamic. Different from MANETs, this is not so much due to mobility of the sensor nodes (they are in fact often static), but rather to the easy failure of lightweight devices with limited power, and the fact that often new sensor devices are added. Finally, the communication patterns in sensor networks are quite specific: each sensor node acquires data at regular intervals, and needs to send this data to the sink node. It is important to take these patterns into account when designing network protocols, in order to obtain better usage of the limited available resources.

2.2.4 Algorithms for different types of AHWMNs

All three types of networks described above possess the typical properties of AHWMNs: they have an ad hoc, dynamic topology, they use unreliable connections, they are highly decentralized, and they have limited network resources. Also, they might all grow to large sizes, although this aspect is more pronounced in sensor networks than in MANETs and WMNs. Due to these similarities, networking algorithms for different types of AHWMNs should have similar properties, such as adaptivity to a changing environment, robustness with respect to packet corruption or loss, a highly distributed organization, efficiency and scalability. As a result, there is often a large overlap between algorithms used for the different types of AHWMNs. Especially for MANETs and WMNs this is the case. MANETs were the first AHWMNs that received a lot of research attention, and WMNs can in a way be seen as the follow up of MANETs. Therefore, WMNs use to a large extent the algorithms that have been proposed for MANETs. Sensor networks have been developed more or less in parallel with MANETs, and have more specific algorithms. The work in this thesis is in the first place focused on MANETs and WMNs.

2.3 Issues in ad hoc wireless multi-hop networking

This section builds on the general description provided in section 2.2 to investigate important issues for networking in AHWMNs. We start with aspects of network connectivity and node mobility, and then move up the network protocol stack discussing issues related to the physical layer, the data link layer, and the transport layer. Specific attention is given to how these issues have consequences for the task of routing. Routing itself is not discussed in this section: since it is the main focus of attention of this dissertation, it is treated in more detail in section 2.4.

2.3.1 Network topology and node mobility

As stated in 2.2, the topology of an AHWMN is normally not the result of careful planning, but arises from the placement of the nodes. In what follows, we first comment on how this affects network connectivity, and then on how it relates to node mobility.

Network connectivity

There is a link between two nodes of an AHWMN if they can receive each other's radio signals. Therefore, the topology of the network is directly defined by the relative placement of the nodes and the range of their radio transmitters. Since the placement of the nodes in an AHWMN is done in ad hoc manner, with minimal planning, it can be considered as a random process, from which the network topology emerges. An important factor in this process is the node density,

since it directly influences the connectivity in the resulting network topology. In [88, 89], the theory of percolation is used to investigate this relationship between node density and network connectivity analytically. The authors show that there is a quite clear cut off point in node density, called the critical density, under which the network falls apart into small, internally connected islands, and above which there is connectivity between the majority of the nodes in the network. For densities that are just above the critical density, this network wide connectivity is quite sparse, so that paths between most pairs of nodes depend on the availability of a few critical links. The failure of any of these critical links has a large impact on the routing possibilities in the network. This is in contrast with densely connected networks, where often many alternatives are available to route around a failed link. This means that in sparse networks, it is more difficult for a routing algorithm to provide adaptivity to changes in the network topology. So, we can conclude that the node density of an AHWMN directly affects the difficulty of the routing task.

Node density only has meaning when treated relative to the transmission range of the nodes' radio equipment: if this range is reduced while the number of nodes per unit of area stays the same, the connectivity of the network decreases. This means that variations of the radio range influence network connectivity as much as node density. Radio range variations can happen accidently, for example because of random variations in the environment or because of changes in the available power in each node (see also explanations in 2.3.2), or can be induced intentionally, for example in order to save battery power or to reduce radio interference between different transmitters [150, 195]. Some work treats the problem of defining a minimal power usage for each node under the explicit constraint that there needs to be at least one path between each pair of nodes in the network [114, 196]. This is particularly relevant in sensor networks, where, as mentioned in 2.2.3, batteries can sometimes not be replaced. Clearly, the application of such schemes can give rise to difficult topologies to maintain routing in.

Node mobility

Since the network topology is defined by the placement of the nodes, it changes when the nodes move. As a consequence, the difficulty of the routing task is also strongly influenced by the characteristics of the node mobility. These characteristics include the speed of the movements, and the specific patterns followed by the nodes. The latter defines for example how the nodes move relative to each other, and can give rise to temporary differences in node density. The impact of node mobility is especially important in MANETs, where all nodes are mobile.

Node mobility depends on the usage of the network. Unfortunately, most research on AHWMNs is done in academic context, without a clear understanding of their purpose. Mobility is therefore usually simulated with artificial models, of which the Random Waypoint model (RWP) [140] is by far the most popular. Under this model, each node picks a random destination, and a random speed,

and moves in a straight line to this destination with the chosen speed. Then the node pauses for a certain time, after which it picks a new destination and speed. Other models use different approaches, or model specific behaviors such as e.g. group movements. An overview of mobility models used in the literature is given in [46]. In recent years, there is a growing suspicion towards these artificial mobility models, because they do not reflect real movements very well, and because they may artificially give rise to certain node distributions. For example, under RWP, nodes tend to cluster more in the center of the AHWMN area, so that there is higher density there, giving better connectivity [29]. There is now a lot of interest in collecting traces of real movements of people (e.g., see [55]), but so far very few such information is available.

An interesting side remark can be made here with respect to the relationship between mobility, connectivity and network capacity. If applications can tolerate high delays, communication between remote nodes in the network can profit from node mobility by letting packets be temporarily stored in moving nodes, so that they can travel closer to their destination this way. This can increase the capacity of the network, since less wireless retransmissions need to be done [118]. It can also allow communication in networks where there is no direct connectivity between source and destination nodes. This is the area of delay tolerant networking (DTN) [44]. DTN was in the first place developed as a solution for interplanetary telecommunications, where some links can incur enormous delays, and some recipients can temporarily be out of range for communication, e.g. a space station that is circling around a remote planet. Recently, the term DTN has also been adopted to describe AHWMNs with intermittent connectivity, such as e.g. MANETs consisting of people carrying short-range bluetooth devices. Also the terms opportunistic networking or pocket switched networking are used [133, 246]. While these "terrestrial" DTNs could be seen as a new type of AHWMNs, they are usually still considered MANETs, operating in the extreme case where there is very limited connectivity. Nevertheless, they need specific networking algorithms, which can deal with these difficult circumstances. Such algorithms are outside the scope of this dissertation though.

2.3.2 The physical layer

The physical layer is concerned with issues regarding the physical transmission of data between two nodes. While a lot could be said about different radio transmission technologies that can be used, we will here limit the discussion to some issues that have direct implications for routing. First, we discuss about the occurrence of unidirectional links, and next, we comment on how choices at the physical layer are defining for network capacity.

Unidirectional links

Most networking algorithms for AHWMNs assume all links in the network to be bidirectional: if node i can hear node j, then node j can also hear node i.

In reality however, an AHWMN can also contain unidirectional links. These can occur for various reasons. One reason can be a difference in radio range between the nodes: if i has a higher range than j, it is possible that j can hear i while i cannot hear j. Such a difference in radio range can be chosen deliberately, or can be the result of a difference in available battery power in the nodes (see subsection 2.3.1). Another, related reason for the occurrence of unidirectional links is radio irregularity. It has been observed that the radio range of wireless nodes does not form a perfect circle around the node, but instead shows quite irregular patterns [290]. This is mainly due to differences in radio wave propagation in different directions. A third reason for unidirectional links can be interference by other transmitters. It is possible that the level of interference is different at i and j, so that one of the two can temporarily not receive data from the other. The negative impact on network performance due to the presence of unidirectional links has been documented in various works [187, 219].

Network capacity

Choices at the physical layer are defining for the network capacity. Most work on AHWMNs relies on WiFi technology (IEEE 802.11) [3], which can in theory provide a relatively high throughput of up to 54 Mbps. Other, newer technologies, such as UWB (IEEE 802.15.3) [4, 282] and WiMax (IEEE 802.16) [5], promise even much higher throughput. Despite these high bandwidth values, however, the actual available capacity in an AHWMN is much lower. This is due to interference between different transmitters. Different pairs of nodes in the network can only communicate simultaneously if they are situated far enough from each other, so that they do not disrupt each other's signal. This is also referred to with the term "spatial reuse": the wireless channel can be used for multiple simultaneous communications if there is spatial separation. In [120], the authors investigate how much capacity is actually available in an AHWMN if spatial reuse is optimally used. They conclude that the available capacity per node in bit-meters per second is inversely correlated with the square root of the total number of nodes in the network, which means that for large AHWMNs, the available capacity per node tends to zero. This result is not very encouraging for AHWMN research, but needs to be read with some caution. The investigation was done for networks using single-channel, omnidirectional antennas. If more than one channel is used, or other antenna systems, such as directional antennas [284] or multi-antenna systems [179], better capacity can be obtained. Nevertheless, when developing algorithms for AHWMNs, one needs to be aware that the total available bandwidth is much lower than what wireless technologies can in theory provide, so that efficiency is important.

2.3.3 The data link layer

The data link layer is concerned with the organization of transmission and retransmission of data between two nodes. Often, the data link layer is identified

with its most important component, the medium access control (MAC) sublayer. This component deals with the coordination of the access of different nodes to a shared communication medium. In AHWMNs, this comes down to avoiding interference while maximizing spatial reuse. A different goal, which is mainly important in sensor networks, is power saving [19]. In what follows, we first describe general issues related to MAC organization in AHWMNs, and then discuss the most popular AHWMN MAC protocol, namely the IEEE 802.11 DCF protocol.

Medium access control in AHWMNs

Since AHWMNs are usually highly decentralized, MAC protocols need to work in a distributed way. This makes it difficult to organize the sharing of the wireless medium in an efficient way via the creation of multiple channels through frequency or code division multiplexing, or via the assignment of time slots. Research is therefore mainly concentrated on single channel, contention based algorithms.

Contention based algorithms do not strictly separate simultaneous communications, but instead allow the possibility of a collision of transmissions [254]. If such collision happens, the transmission should be retried. Carrier sensing can limit the probability of collisions: the basic idea is to listen whether the channel is idle before starting to send. Important problems that need to be solved by single channel contention based algorithms are the hidden terminal and the exposed terminal problems. Consider figure 2.3, where four wireless nodes are placed in a row, so that the distance between each pair is roughly equal to their wireless radio range (assuming an idealized circular radio range for each node). The hidden terminal problem takes place when C wants to send to D while A is sending to B. Since C cannot hear the signals from A, it can erroneously assume that the channel is free and start sending, hereby disrupting the reception at B. The exposed terminal problem is more or less the opposite: when B is sending to A, C can hear the signal and therefore decide that the channel is busy, while a simultaneous transmission from B to A and from C to D is possible without causing interference at signal reception.

The most popular MAC algorithm in AHWMN research is the IEEE 802.11 Distributed Coordination Function (DCF) [135], which we describe below. A range of other MAC algorithms have been proposed. Many of these algorithms are variations on IEEE 802.11 DCF, or are at least similar in approach. See e.g. [159] for an overview.

IEEE 802.11 DCF

The IEEE 802.11 DCF MAC algorithm is a contention based single channel protocol. It uses both carrier sensing and extra control packets to limit the number of collisions. The basic working of IEEE 802.11 DCF is as follows. When a node wants to start a transmission, it listens to check whether the wireless medium is free. Next, it sends a request-to-send message (RTS) to the

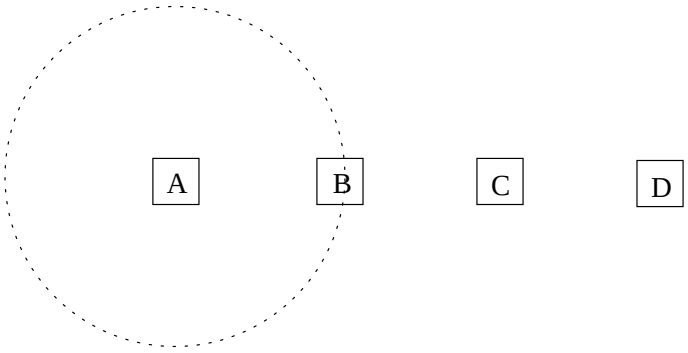


Figure 2.3: The hidden and exposed terminal problems (figure adapted from [254])

node it wants to transmit to, in order to request the start of a conversation. If this node is free to receive, it sends a clear-to-send message (CTS) back. The requesting node can then start the transmission of data, and if this is successful, the receiving node concludes the communication by sending out an acknowledgement message (ACK). Between each of the steps of this process, there are small waiting times to avoid collisions. If the process goes wrong at any time, it needs to be restarted completely, after a randomly chosen backoff interval. With each failed transmission attempt, the window from which this backoff interval is chosen, is doubled. The hidden terminal problem is solved by the RTS/CTS mechanism. The exposed terminal problem is not solved by the IEEE 802.11 DCF algorithm. A special remark needs to be made regarding broadcasting. In principle, all communication with omnidirectional wireless antennas is done in broadcast mode (i.e., all nodes on the medium receive the message). However, the RTS/CTS and ACK mechanisms are done with just one node and assure the implementation of unicasting. Therefore, if a node wants to broadcast a message to all its neighbors, it cannot use these mechanisms. As a consequence, broadcast transmissions are less protected and therefore less reliable than unicast transmissions.

A lot of research has been done to assess the efficiency of the IEEE 802.11 DCF algorithm. First of all, due to the sending of extra control packets around each data packet, and the small delays between each of these sendings, the theoretical maximum throughput using IEEE 802.11 DCF is a lot lower than the actual capacity of the radio transmitters [143] (down to 50% or less depending on the size of the transmitted data packets). More problematic is the use of the exponentially growing random backoff interval: if there are a lot of collisions, e.g. when there is a lot of data traffic in the network, it is possible that some communications are suspended for a longer time, or even get completely starved [28]. This problem can be acerbated in multi-hop communications, where subsequent hops in a path can actually interfere with each other [170, 280].

As a consequence of all this, the use of IEEE 802.11 DCF puts further severe

limitations on the already compromised capacity of AHWMNs (the capacity limitations based on spatial reuse calculations described in 2.3.2 assumed a perfect MAC layer protocol). Some improvements have been proposed in other MAC protocols (see [159]), but the basic problems stay the same. The main cause of inefficiency in MAC layer organization is the decentralized nature of AHWMNs, which commands a distributed approach. Improvements are possible if there is some structure in the network, such as between backbone nodes in a WMN (see 2.2.2), or if node mobility is low enough so that there is time to set up a different MAC organization, as is sometimes done in sensor networks [19,161]. In other cases, however, we need to be aware that the MAC layer limits the capacity of the network, and increases the need for efficiency at the routing layer.

2.3.4 The transport layer

The transport layer is situated above the routing layer. It is concerned with the organization of the transport of data between the source and destination nodes of a communication session over the path provided by the routing algorithm [254]. The two main transport protocols in the internet are the User Datagram Protocol (UDP) [217] and the Transmission Control Protocol (TCP) [218]. TCP offers a secure transport service: a triple handshake mechanism is used to establish a connection at the start of the transport session, the arrival of each data packet is acknowledged by the receiver, warnings are sent from the destination to the source if packets arrive out of order, lost packets are retransmitted, and a congestion control mechanism is used to adapt the speed of the data flow in case there is packet loss. UDP, on the other hand, offers a service without any guarantees: it just sends packets on their way from source to destination, without using any of the mentioned mechanisms. UDP can be preferred over TCP for short data transports when the underlying network is very reliable. In what follows, we comment on the use of TCP and UDP in AHWMNs.

TCP versus UDP in AHWMNs

AHWMNs are inherently unreliable networks, due to their use of wireless links and their dynamic nature. So it makes sense to use a secure service like TCP, rather than UDP. However, there are problems with the use of TCP in AHWMNs, stemming from various causes.

A first problem for TCP in AHWMNs is that its various mechanisms cause a lot of extra overhead [111]. For example, for each data packet going from source to destination, an acknowledgement is sent back from destination to source. This goes in conflict with the typically low capacity of AHWMNs, as described in 2.3.2 and 2.3.3. This problem can be acerbated due to interactions with the MAC layer algorithm, as TCP uses exponentially growing random backoff timers for retransmission, just like IEEE 802.11 DCF (see 2.3.3). A second problem is caused by TCP's congestion management mechanism [22,250]. TCP was designed with wired networks in mind, where data transmissions are quite

reliable. Therefore, when TCP detects a problem with data delivery (out of order delivery or packet loss), it assumes that the cause is congestion of the network, and consequently reduces the data flow. In AHWMNs, however, packet loss is often caused by the use of unreliable wireless links or mobility induced path loss. TCP will therefore lower its throughput unnecessarily and reduce the throughput of the network. A third problem has to do with TCP timers. To define whether a packet was lost, TCP uses timers, which are based on an estimate of the time it takes for a packet to go to a destination and for an acknowledgement to come back to the source. This is the so-called round trip time (RTT). In an AHWMN, which is an inherently dynamic environment, the RTT can vary greatly, making its estimation difficult. This causes TCP to detect false packet losses, which are again interpreted as caused by congestion, with a reduction of the data throughput as a consequence. A final problem has to do with the frequent path failures that occur in AHWMNs [130]. In these situations, there is again packet loss, but the destination will not be able to warn the source about it. Therefore, the source only detects the loss when its timer for the acknowledgement runs out, causing extra delay.

The above problems make the use of TCP in AHWMNs problematic. Therefore, many variations of TCP have been proposed, in an attempt to deal with these difficulties. See [268] for an overview. Nevertheless, many AHWMN research studies avoid these problems altogether by using the UDP transport protocol. For the routing layer, the important message is that correct data delivery is an important evaluation criterium, as the transport layer above it has difficulties providing reliability.

2.4 Routing in ad hoc wireless multi-hop networks

Routing is the task of directing data flows from source nodes to destination nodes while maximizing network performance. This is particularly hard in AHWMNs. Due to the ad hoc and dynamic nature of the network, the topology can change constantly, and paths between sources and destinations that were initially efficient can quickly become inefficient or even infeasible. This means that routing information should be updated more regularly than in traditional wired telecommunication networks. However, this can be a problem in AHWMNs, with their limited bandwidth and node resources, and their possibly unreliable communication channels. New routing algorithms are therefore needed, which can give adaptivity in an efficient and robust way.

A lot of research has been done on routing in AHWMNs in the past few years. In this section, we give an overview of the research in this area. The aim is not to present an exhaustive list of existing protocols (for this, we refer to existing surveys, such as [14, 227]), but rather to describe the most important ones and to give insight into interesting ideas and techniques that have been developed in this area. We start the section with an explanation of the distinction between

proactive and reactive routing protocols, which is the most used criterium to classify AHWMN routing algorithms. Next, we describe in detail a number of representative routing algorithms. Finally, we give an overview of important techniques used in other AHWMN routing algorithms.

2.4.1 Proactive versus reactive routing algorithms

Traditionally, AHWMN routing protocols are classified as proactive or reactive protocols. Under proactive routing protocols, all nodes of the network try to maintain consistent routing information about all other nodes at all times. This means that routing information needs to be updated after each change in the network. Data are routed by a simple lookup in the tables which store the routing information, and proactive routing algorithms are therefore also called table-driven [227]. Examples of proactive algorithms are Destination-Sequenced Distance-Vector Routing (DSDV) [211] and Optimized Link State Routing (OLSR) [61] (see subsection 2.4.2 for descriptions of both algorithms). Under reactive routing protocols, nodes only gather the routing information that is strictly needed. This is the case when a new data session is started, or when a currently used path fails. Gathering routing information usually involves a route discovery or a route repair phase. Reactive algorithms are also called on-demand algorithms [227]. Examples are Dynamic Source Routing (DSR) [140] and Ad-Hoc On-Demand Distance Vector Routing (AODV) [213] (see subsection 2.4.2).

Proactive algorithms have the advantage that routing information is always readily available when data needs to be sent. Also, all changes in the network are taken into account, so that new routing opportunities can be exploited, and backup paths can be provided when primary paths fail. On the downside, these algorithms can become quite inefficient or even break down completely when a lot of changes need to be tracked. This is the case when the topology is highly dynamic, or when the network is large. Reactive algorithms only focus on the routing information that is strictly necessary. This way, they can greatly reduce the overhead they create, so that they are in general more efficient. The disadvantage is that they are never prepared for disruptive events, and that some data packets can therefore incur large delays, e.g. during the route setup phase at the start of a communication session. In MANETs, where all nodes are mobile, so that there are a lot of topology changes, the general preference is for reactive algorithms [42]. However, in WMNs, which are less dynamic, and in sensor networks, where nodes are often not mobile, proactive algorithms can also be a good choice.

So far we have used the terms proactive and reactive routing in a rather strict sense. We can define reactive behavior in a more general way as the gathering of routing information in reaction to an event, and proactive behavior as the gathering of routing information at all other times. Algorithms which combine both reactive and proactive elements are called hybrid algorithms. They try to combine the advantages of both approaches. The classic example is the Zone Routing Protocol (ZRP) [121] (see subsection 2.4.2). In practice, a lot

of AHWMN routing algorithms can be labeled hybrid, since they contain some combination of reactive and proactive behavior.

2.4.2 Important routing algorithms for AHWMNs

In what follows we describe in detail some of the most representative routing algorithms for AHWMNs. For routing in MANETs and WMNs, which largely use the same protocols, we describe two proactive algorithms, DSDV and OLSR, two reactive algorithms, DSR and AODV, and one hybrid algorithm, ZRP. For sensor networks, we describe directed diffusion, which is a reactive protocol.

Destination-Sequenced Distance-Vector Routing

DSDV, published in 1994 [211], was the first routing algorithm for AHWMNs. It is a direct adaptation of distance vector routing, which is at the basis of internet routing protocols such as Routing Information Protocol (RIP) [123] and Border Gateway Protocol (BGP) [178] (see also subsection 3.1.1).

In distance vector routing algorithms, every node i keeps for every destination x a set of distances $\{d_{ij}^x\}$, indicating the distance to reach x from i when neighbor j is used as the next hop in the path. The distance metric can be the number of hops or the end-to-end delay. Data that need to be routed for destination x are sent to the neighbor with the lowest value for the distance. In order to keep the distance vector tables up to date, each node periodically broadcasts an update of its shortest routes to its neighbors. A neighbor k receiving from i the update d_i^x, which is the shortest distance from i to x, can calculate its own distance to x over i, d_{ki}^x, by combining d_i^x with the cost of the link from k to i (which it monitors locally). Even if nodes initially start with completely random estimates for each of the path costs $\{d_{ij}^x\}$, the system of continuously updating the cost estimates eventually converges, so that all nodes have correct estimates for the costs of all paths. Distance vector routing is also known as distributed Bellman-Ford routing [27], and stems from the adaptation of dynamic programming [26] to the problem of routing. More on distance vector routing will follow in subsection 3.1.1.

One of the main problems with distance-vector methods is that loops can be formed. This is mainly due to the fact that nodes make routing decisions in a distributed way, and possibly do so using stale distance information. Moreover, since nodes base their routing information on estimates provided by other nodes, wrong information can quickly propagate through the network, and even though eventually all routing information should converge to correct values, this might take quite long (see e.g. the problem of counting to infinity [254]). As routing information gets out of date really fast in AHWMNs, this problem can get quite severe. DSDV solves the problem using sequence numbers assigned by the destination node. When faced with multiple route updates, nodes always prefer the fresher information (the newest sequence number). Only when two routes are equally fresh the shorter route is chosen. Another feature in DSDV is the use of incremental updates. In order to avoid that nodes have to broadcast

complete routing tables at every topology change (which can be very often), it is possible to report only the changes since the last full table update.

Despite the careful adaptations done to the original distance-vector method, DSDV suffers from the earlier mentioned problem usually faced by proactive algorithms: tracking all changes in a dynamic network is inefficient. Simulation studies which compare different routing algorithms confirm this: in [42] it is found that DSDV's performance deteriorates quickly with increasing network mobility.

Optimized Link State Routing

Like DSDV, OLSR [61] is a proactive routing protocol. And like DSDV, OLSR was inspired by an important class of routing algorithms for wired networks, namely link state routing, to which the internet routing protocol Open Shortest Path First (OSPF) [197] belongs (see also subsection 3.1.2).

The typical mode of operation of link state routing algorithms is that each node locally monitors the network situation, and periodically floods its local view over the network. By combining all received local views, each node can get a complete picture of the network, and calculate shortest paths to all destinations based on it. This is different from distance-vector algorithms, where nodes periodically send out their complete set of routes (so not just their local view), and where these updates are only sent to direct neighbors, instead of over the whole network. The main advantage of link state routing is that it converges faster to correct routing information. However, it is not more robust than distance vector routing (things can go quite wrong when routing updates get lost), and is not necessarily more scalable, because, even though routing update messages are smaller, they need to be flooded over the whole network, and nodes need to locally build a complete picture of the whole network. See subsection 3.1.2 and [254] for more discussion on link state routing and the difference with distance vector routing.

OLSR offers an adaptation of link state routing which is optimized for AH-WMNs. Monitoring of the local network situation is done with beacon messages, which are periodically sent out by all nodes. When a node i receives a beacon message from a node j, it can conclude that j is its neighbor. When sending its own beacon messages, i includes a list of all its neighbor nodes. E.g., in the example of figure 2.4, node i includes nodes a, b, c, d, e, f, g and j in the beacon message. j can then see that i has received its beacon message (since it sees itself present in i's neighbor list), and conclude that there is a bidirectional link between i and j. Moreover, by combining i's neighbor list with those of its other neighbors, j can get a complete picture of its 2-hop neighborhood. Then follows the most characteristic part of the OLSR algorithm, in which j picks a number of so-called multi-point relays (MPR). This is a subset of its neighbor nodes that is sufficient to reach all nodes in the 2-hop neighborhood. In figure 2.4, j chooses i, k, l and m as its MPR nodes. When j subsequently floods its local view over the whole network, as is needed in link state routing (see above), only j's MPR participate in forwarding the flooded message. This way, the

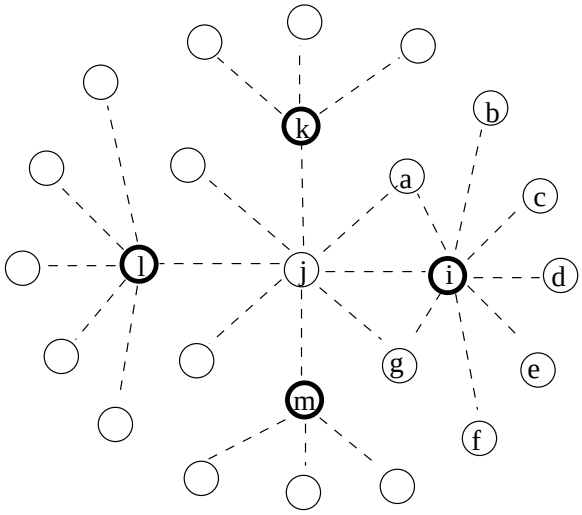


Figure 2.4: The working of OLSR. The dashed lines symbolize wireless links (not all links are drawn in order not to overload the picture). Node j chooses i, k, l and m as MPR nodes, since they are sufficient to reach all its two-hop neighbors. Figure adapted from [61].

flood needs less packet transmissions. Moreover, the local view sent out by j is nothing more than a list of all nodes that have selected j as MPR, j's so-called MPR selector nodes. This means that the flooded messages can be quite small. It also means that the global network views constructed based on these local views contain only connections between nodes and their MPR selector nodes, so that the network is smaller but still fully connected. Finally, if MPR nodes are chosen from among neighbors with which there is a bidirectional connection, the global network view will contain only bidirectional connections. This can be important in case there is irregular radio wave propagation which can cause links to be unidirectional (see also 2.3.2).

Being a proactive routing protocol, OLSR inherits the earlier mentioned efficiency problems in highly dynamic AHWMNs. Also, being a link state routing protocol, OLSR is not very robust with respect to loss of control packets, and not very scalable. Nevertheless, the mechanisms adopted for improving efficiency are interesting, and the fact that bidirectional links can be identified can be an important asset. Consequently, OLSR has received considerable attention since its publication in 2001, especially when it comes to making implementations for real deployment: it was the routing algorithm of choice for the already mentioned olsr.freifunk.net experiment in Berlin [9], and has been implemented for use under Windows and Linux by the Navy Research Labs [8]. In one study with a real MANET testbed [36], OLSR has been shown to give comparable performance to the AODV reactive routing algorithm when the network is small

and has limited mobility. OLSR has been published as a standard by the IETF MANET working group [60].

Dynamic Source Routing

DSR [140] was in 1996 the first reactive routing algorithm to be published. As in other reactive algorithms, nodes only actively look for routing information when it is strictly needed, i.e. when data needs to be sent to a destination for which no valid route exists. Important features of the algorithm are that it uses source routing, and that it makes extensive use of caching to increase the available routing information.

Under DSR, nodes that need to send data first check whether they have any routing information available for the requested destination. If this is not the case, a route discovery process is started, in which a route request message (RREQ) is flooded over the network. Once the RREQ reaches the destination, a route reply message (RREP) is sent back to the source, either following the same path of the RREQ back, or following a different path from the destination to the source. At the return of the RREP, the route discovery process is complete, and the source can start sending data. If later, during the communication, one of the links on the path fails, a route error message (RERR) is sent from the node where the error was detected back to the source, which can then start a new route discovery process.

An important aspect of DSR is that it uses source routing. This means that data are not routed hop by hop, as is common practice in most routing algorithms, but that instead the source node decides about the complete path to the destination, and adds it to each data packet. While this means extra overhead per data packet, source routing has as an advantage that the source has complete control over the path, so that e.g. loops can be avoided. Another advantage is that all nodes that receive a data packet on its way to its destination can extract information about the topology of the network. This last property is exploited extensively in DSR, which allows nodes to cache all routes they can extract from passing data packets (if promiscuous mode is possible, this includes also data packets that were not destined for the current node, but could somehow be overheard). This allows to build a database of existing routes, which can be used to avoid the need for a route discovery process, to provide backup paths in case of failure, etc.. However, caching can also be dangerous. Overheard routes often stay in cache unused for a while before they are actually needed, at which point they might have expired already. As a consequence, data packets can be sent onto erroneous routes, leading to extra delay and overhead. Moreover, these erroneous routes can be overheard by other nodes, so that wrong information can actually spread over the network. This phenomenon is called "cache pollution" [183, 186].

DSR is an important reference algorithm in the field. A number of studies compare it to AODV, the other important reactive algorithm, which is described below, however with few conclusive results [42, 65]. Recently, DSR formed the inspiration for Link Quality Source Routing (LQSR) [92], which is the routing

algorithm used in the Mesh Connectivity Layer (MCL) architecture developed by Microsoft Research labs, and for the Srcr routing algorithm, which was developed for use in MIT's Roofnet WMN testbed [30].

Ad-Hoc On-Demand Distance Vector Routing

AODV [213] was published as the on-demand version of DSDV. Like DSDV, it uses hop by hop data routing based on distance vector tables. Also, it uses the destination based sequence numbers of DSDV to distinguish old from new routing information. Being a reactive algorithm however, AODV is in its general working more similar to DSR.

Under AODV, nodes that have data to send to a destination that they have no information about, start a route discovery process. This process is similar to the one of DSR: a RREQ is flooded over the network, and when it reaches the destination, a RREP is sent back to the source, which can then start sending data over the newly established path. The difference with DSR lies in the way routes are identified: the RREQ does not store the full path that it has traveled, but instead leaves in the routing table of each intermediate node a pointer towards the source. The RREP follows these pointers back to the source, and changes them into pointers towards the destination. Data packets are routed from source to destination following these pointers, and do not have to carry full paths. When there is a failure anywhere along the path, a RERR is used to warn the source, which can start a new route discovery process. A possible improvement, which was studied together with some other optimizations of AODV in [165], allows for intermediate nodes to try to locally rebuild failed routes, i.e. to find a route around a failed link. This is called local repair.

The use of hop by hop rather than source routing has as most important advantage that there is no extra overhead for each data packet. A disadvantage is that nodes cannot extract routes from passing data packets, so that caching like in DSR is impossible. However, as was explained before, caching can also have a negative impact.

AODV is by far the most cited and studied AHWMN routing protocol. It forms the most important benchmark algorithm for the evaluation of other routing protocols, and has been standardized by the IETF [212].

Zone Routing Protocol

ZRP [121] was the first hybrid routing algorithm. It is less used than other routing algorithms presented in this section, but is described here because it provides a simple example of how reactive and proactive routing can be combined.

Under ZRP, proactive routing is used within a certain zone around each node (so each node keeps up-to-date routes to nodes in a k hop radius, where k can be equal to 2 or 3 e.g.), and reactive routing is used for destinations further away. Data for destinations within the zone can therefore be delivered straight away. For destinations further away, a route discovery process is needed. The

use of the zones allows to design a route discovery process that is more efficient than those of AODV and DSR: the source node sends the RREQ to the nodes at the outer edges of its zone, and those nodes forward it to their zone's outer edges and so on, so that the RREQ travels in jumps. The discovered paths consist of a list of zone edge nodes, and do not contain information about the nodes between them. This means that the paths can be smaller to store, and that they can in principle be more robust (since they are not affected by the movement of nodes between the zone edge nodes).

Directed diffusion

Directed diffusion [137] is a reactive routing protocol for sensor networks. It addresses issues of scalability, robustness and efficiency, and was designed specifically to support the typical communication patterns of sensor networks, where the sensor nodes periodically need to forward their sensed data to one or more sink nodes. Directed diffusion has similarities to Temporally-Ordered Routing Algorithm (TORA) [209], which is a reactive routing algorithm that was developed for MANETs.

In directed diffusion, the setup of data paths is initiated from the sink node when data are needed. To this end, an interest message is formulated, which describes which data are needed, for which period they are needed, and with which frequency. This interest is subsequently propagated through the network to all sensor nodes. Each node that receives it sets up a gradient, which is a pointer to the node it received the interest from, and starts looking for the requested data with the requested frequency. When a node obtains data that fit the interest message, it forwards them along the corresponding gradient. Data coming from different parts of the sensor network can be aggregated when they meet in an intermediate node while traveling to the sink. This way, forwarding can be done more efficiently, which is very important given the strict power limitations in sensor networks. Once the sink node starts receiving data, it starts to regularly resend its interest, in order to reinforce and repair the existing paths. This way, higher robustness is provided.

2.4.3 Other techniques for AHWMN routing

The algorithms described in 2.4.2 represent the most cited ones in the area of AHWMN routing. In what follows, we give a broader overview of this research area. In particular, we describe a number of interesting techniques that have been developed, explain how they are used in different algorithms, and comment on advantages and disadvantages of each of them. In particular, we talk about the use of multipath routing, about creating a hierarchical structure in the network, about the use of geographical location information, about different routing metrics, about techniques for large scale AHWMNs, about multicasting, geocasting and broadcasting, and about data-centric routing, which is important in sensor networks.

Single path versus multipath routing

Most routing algorithms use single path routing: at all times, only one path is maintained between source and destination. The alternative is to maintain multiple paths simultaneously. Advantages of multipath routing include higher throughput, because data can be spread over the different paths, and higher robustness, because backup paths are available in case of a failure and because multiple copies of the data can be sent in parallel over different paths. Both of these are important features in the face of the low capacity and reliability that are typical for AHWMNs. Disadvantages of multipath routing are mainly due to the fact that more than one path needs to be set up and maintained. This leads to more complex algorithms, and more overhead. A confirmation of the advantages and disadvantages of multipath routing in AHWMNs, both via analysis and simulation, is given in [214]. On the other hand, a similar analytical study in [109] shows how a bad spreading of the multiple paths can negatively affect the throughput advantage.

A large number of multipath routing algorithms have been proposed in AHWMN research. See [198] for an overview. These algorithms differ in the way multiple paths are set up, maintained and used. During path setup, a number of different paths are selected. Some algorithms allow braided multiple paths [108], whereas others look for link [186] or node [283] disjoint paths, or even paths which are outside each other's interference range [278]. This is important in AHWMNs, where the augmented throughput offered by the use of multiple paths can only be realized if these paths do not interfere with each other. Once the paths are set up, they need to be maintained. Most algorithms manage the paths in a reactive way: they remove paths when a link break occurs, and only take action when no valid path to the destination is left. A few algorithms use probing to obtain up-to-date information about the paths and to detect failures [108, 269]. Finally, the way the multiple paths are used differs strongly among algorithms. In many of them, only one of the paths is used for data transport, while the others are only used in case of a failure in the primary path [163, 201]. Some algorithms also explore the idea of spreading data over the multiple paths to increase the throughput [108, 164, 269]. An interesting way of spreading data to increase robustness is presented in [261]: diversity coding allows to encode N blocks of data into $M + N$ blocks in such a way that any subset of size N of these $M + N$ blocks allows to reconstruct the original data. By using this encoding and spreading data over multiple paths, robustness of data delivery can be greatly improved.

Network structure

Many AHWMN routing algorithms, including the algorithms DSDV, OLSR, AODV and DSR described in 2.4.2, consider all nodes in the network as equal and perform routing in an essentially flat space. These are called uniform protocols. A different approach is to try to make the routing task easier by giving some structure to the routing space. This is the strategy used by partition-

ing protocols. A classic example is Clusterhead Gateway Switched Routing (CGSR) [59]. This algorithm organizes the AHWMN in clusters, each with a clusterhead and a number of gateways, and organizes all communication between nodes of different clusters via the clusterheads and the gateways.

The advantage of partitioning algorithms is that they simplify routing, so that in principle better efficiency and scalability can be achieved. In practice, however, there are also some important disadvantages. First, if mobility is high, the maintenance of the network partitioning can cause a lot of overhead. Second, attributing roles such as clusterhead or gateway to certain nodes means that the network is more dependent on them, and that they have to perform more work than other nodes (this can be a problem if there is limited battery power e.g.). Finally, although partitioning leads to faster route discovery in large networks, the discovered routes are not always the shortest: nodes could be close to each other in the network, but still have to communicate through their respective clusterheads. In sensor networks, where nodes are often static and scalability is a very important factor, partitioning algorithms are relatively more popular than in MANETs and WMNs. An example is the Low Energy Adaptive Clustering Hierarchy protocol (LEACH) [125].

A middle way between uniform and partitioning protocols is given by neighbor selection protocols [102]. In these protocols, each node gives some structure to the network from its own point of view. An example is the previously mentioned ZRP (see 2.4.2), in which each node uses a proactive protocol to maintain routing information about nodes within a certain number of hops, and a reactive protocol for nodes further away. Another example is Fisheye State Routing (FSR) [210], in which updating of routing information is done frequently for nearby nodes, and less frequently for nodes further away. This results in a blurred vision of the network: for faraway nodes only a vague routing direction is known, but as the data packet approaches its destination, more precise routing information is available. The advantage of neighbor selection protocols is that there is some structure in the network, which improves scalability and efficiency, without there being an overall structure that needs to be maintained.

Using location information

Some AHWMN routing algorithms use geographic location information for routing (e.g. [146]). In such location based algorithms, source nodes add the geographic coordinates of the destination to each data packet, and forwarding decisions are based on which next hop brings the data packet closest to its destination. This approach exploits the fact that the network topology is defined by the placement of the nodes (see 2.3.1), so that geographical proximity is closely related to proximity on the network topology graph. A good overview of existing location based algorithms can be found in [113].

Advantages of location based routing are that no explicit path setup is necessary, that paths are flexible (here, a path is not a list of nodes to be visited, but instead results from the location based decisions at each intermediate node), and that very little routing information needs to be maintained. This makes the

system more efficient, scalable and robust. There are, however, also a number of problems that need to be solved in this approach to AHWMN routing. First of all, each node has be able to obtain its own location coordinates. This is normally done through the use of the Global Positioning System (GPS) [129]. However, use of GPS can be expensive, and is not always possible (e.g. indoors). In [47], the authors present a rather complicated method for GPS free location detection. A second problem is that source nodes need a way to figure out the location coordinates of the destination node, and maintain them in the face of mobility. This is usually done using some form of location server. However, the use of a single location server is in conflict with the distributed and ad hoc nature of AHWMNs, and creates a single point of failure. Some solutions to this problem are described in [31, 171]. Finally, location based algorithms can experience difficulties when there is a big gap in the graph that stops greedy forwarding from finding the destination. Approaches to get around wholes in the network graph are presented in [31, 157].

Routing metrics

Routing algorithms need a metric to evaluate different routes and choose one (or several) among them. The most straightforward choices are the end-to-end delay and the hop count. The end-to-end delay can however be quite unstable in AHWMNs. One reason is that there is often only one wireless device, with one queue that is shared for all wireless links, so that the traffic for one neighbor also suffers from high delays for the traffic for other neighbors. Another reason is that differences in wireless connection quality can lead to high delay variations, e.g. due to the exponential backoff interval used for retransmissions at the MAC layer as explained in 2.3.3. Compared to delay, the hop count is a very simple and stable metric. It has therefore been used by a lot of AHWMN routing algorithms. Nevertheless, there is a growing awareness that hop count is not necessarily a good metric [67]. This is because paths with a low number of hops usually consist of long hops, which can be of low quality and break easily as a consequence of node movement, and because short paths tend to go through the center of the AHWMN area, where congestion and wireless channel contention is higher.

A number of other metrics have been proposed to evaluate paths. In Associativity-Based Routing (ABR) [257], nodes periodically broadcast beacon messages, and they count how many of these messages they have received from each of their neighbors. These are called "associativity ticks". Links with a high number of associativity ticks are considered more stable, and are therefore preferred.

Power aware routing algorithms (e.g. [247]) evaluate paths based on their power usage. Apart from saving on limited power resources, lowering power usage also reduces interference and increases the possibility for spatial reuse. Different power based metrics are possible. One can e.g. choose the path which uses minimal power, or the path going over nodes with most power left (so that the remaining lifetime of the path is maximized). A number of power based routing metrics are discussed in [240]. A general survey on power aware

networking (not just routing) is given in [141]. Power aware networking is especially important in sensor networks, where power resources are usually more limited and can often not be replaced (see also 2.2.3).

An interesting newly proposed metric is the Expected Transmission Count (ETX) [66]. This metric estimates the expected number of transmissions that will be needed to successfully conclude the transfer of a data packet over a link. This includes possible retransmissions of the data packet due to failures, and the transmission of the acknowledgment packet in backward direction. The estimate is obtained by measuring the transmission success of regularly sent probe messages. An important reason for the development of ETX was the observation that MAC layer mechanisms such as the RTS/CTS and ACK messages of IEEE 802.11 DCF allow to send data packets over lossy links, making it difficult to assess the real link quality. The ETX measure was developed based on experiences from real AHWMN deployment, and formed the basis for the Expected Transmission Time (ETT) metric used in the Roofnet testbed [30]. In [93], the ETX metric was compared to hop count and end-to-end delay. It was found to give significantly better performance in static networks (although producing longer paths), but was outperformed by hop count in dynamic networks because the probe based transmission estimation was too slow to adapt.

Large AHWMNs

In the field of MANET and WMN routing, most of the work is done on the scale of a local area network (LAN), with simulation and implementation tests carried out with networks of up to maximally 100 nodes. An exception is the scalability study of AODV described in [162], which evaluates the performance of AODV in a network of up to 10000 nodes. This paper proposes a number of extensions to AODV to improve its performance in such large networks: expanding ring search, query localization and local repair. With expanding ring search, the flooding of the RREQ message for initial path setup is limited in size in order to improve the efficiency of the process: the flooded message is forwarded for a maximum number of hops, and only if this limited flood does not result in a path to the destination, it is gradually increased in size. Also query localization and local repair are meant to make the path setup more efficient, but only in the case of the failure of an existing path. With query localization (originally proposed in [53]), the RREQ flood is only allowed to propagate in the neighborhood of a previously known path to the destination. With local repair, path failures are resolved with a limited flooding of a repair message around the area where the failure was detected. Despite these adaptations, AODV's performance was found to degrade quickly for large networks, due to the long path lengths. For example, throughput was down to 50% at 1000 nodes.

A few algorithms were proposed specifically for MANETs and WMNs of wide area network (WAN) scale. An example is the Terminodes routing algorithm [31]. It uses geographic location information for scalable routing, and maintains a small world overlay network to support efficient route detection [32]. The algorithm performed well in large scale simulation tests that went up to

600 nodes. Also the Mobile Ants Based Routing (MABR) algorithm [126, 127] aims at supporting routing in large AHWMNs. Like Terminodes, MABR uses geographical information to increase scalability. Also, it divides the whole area of the AHWMN into zones to make the system more manageable. In a large simulation study with 10000 nodes, it compared favorably to Terminodes routing [127]. The MABR algorithm is partly based on the Ant Colony Optimization meta-heuristic described in chapter 3.

In the field of sensor network routing, scalability is very important, since the expected size of such networks is typically very large. To provide such scalability, sensor network routing protocols often use location based routing [281], and/or network partitioning techniques [125]. Sometimes, sensor networks are considered so large that it is impossible to assign a unique identifier to each node. In that case, a possible solution is to apply data centric routing, where data is routed based on their content, rather than based on a destination node identifier [18]. Data-centric routing is described in more detail later in this section.

Unicasting versus multicasting, geocasting and broadcasting

So far, we have only discussed unicast routing protocols, in which a sender sends to one particular receiver. Some protocols also support communication between a sender and a group of receivers. Based on characteristics of the group of receivers, one can distinguish between multicasting, geocasting and broadcasting.

In multicast protocols, the group of receivers is a subset of the nodes of the network, where each member needs to be explicitly identified. Many of the often cited applications of AHWMNs, like disaster recovery and battlefield communication, typically need support for this form of communication. Multicasting could be provided via the use of parallel unicast sessions between the source and each of the destinations. Such an approach would however be quite inefficient. A commonly used alternative is to built a routing structure between the members of the multicast receiver group, so that messages from the source can efficiently be forwarded between them. This routing structure can in principle have any shape. E.g., the approach described in [279] proposes to use a tree structure. In [166], several multicast routing protocols for AHWMNs are described, and their performances are compared in a simulation study. One conclusion of the paper is that multicast protocols using a mesh routing structure are to be preferred over those using a tree structure, since they provide multiple paths between each pair of receivers.

Geocasting can be seen as a special case of multicasting, whereby the group of receivers is defined as all the nodes that are currently in a certain geographical area. The term was first proposed in [203]. Typical applications of geocasting could be local advertising or service provisioning, finding out who is in your neighborhood, or geographic message delivery (e.g., sending an emergency message to all nodes in a certain area). In sensor networks, where the relation between the sensed data and their location is often important, geocasting is of-

ten needed. In [154], three different geocast routing protocols, all adapted from existing multicast routing protocols, are described and compared.

In broadcasting, the group of receivers contains all the nodes in the network. It is important to understand the difference between broadcasting at the MAC level and broadcasting at the network level. At the MAC level, a broadcast reaches all nodes that are in radio range of the sender. Some details about MAC level broadcasting in AHWMNs are given in 2.3.3. At the network level, a broadcast reaches all nodes in the AHWMN, some of which possibly after multi-hop relaying. Network level broadcasting is also often referred to as flooding. Many existing AHWMN unicast routing algorithms rely on flooding to spread or gather routing information. Examples are AODV's and DSR's flooding of RREQ messages to set up paths, and OLSR's flooding of local network views via MPR nodes (see 2.4.2). Due to the need for multi-hop forwarding of flooded messages, flooding can get quite inefficient. In fact, in reactive algorithms such as AODV and DSR, flooding is the most important source of control overhead. Some mechanisms have been proposed to make flooding more efficient, such as OLSR's use of MPR nodes. An overview and comparison of efficient flooding protocols is given in [285].

Address-centric versus data-centric routing

Address-centric routing is the normal way of operation in computer networks: data are routed through the network based on the address of their destination node. Data-centric routing provides a radically different approach: destination addresses are not used, and data are instead forwarded based on their contents. Under data-centric routing, routing information does not indicate the next hop corresponding to a specific destination address, but corresponding to certain data properties. An important example is the directed diffusion algorithm described in 2.4.2. In terms of operation, data-centric protocols can be classified as query based or negotiation based. In query based algorithms, such as directed diffusion [137] and rumor routing [41], the destination node defines the properties of the data it is interested in, and the sensor nodes forward the requested data. In negotiation based algorithms, such as sensor protocols for information via negotiation (SPIN) [158], sensor nodes advertise which data they have available, and destination nodes can register to receive this data. Closely related to data-centric routing is the more general framework of content-based networking [52].

Data-centric routing was in the first place developed for sensor networks, to support the typical communication patterns of such networks, where data measured by sensors spread over a wide area need to be drawn to one or more sink nodes. In this context, data-centric routing has some important advantages. These are in the first place related to the fact that data forwarding is made more efficient: sensor nodes only send the data that are requested by the sink node, and similar data coming from different sensors can be aggregated at intermediate nodes to travel more efficiently till the sink. The efficiency advantages due to aggregation have been confirmed via analytical modeling in [156]. Other

advantages of data-centric routing come from the fact that no destination node address is used in the routing. This can help reduce the need for network wide topology information, as in directed diffusion [137], and eliminates the need for a global space of unique addresses. These are important elements to improve scalability and efficiency in very large sensor networks.

2.5 Conclusion

In this chapter, we have described the context in which the work for this thesis is situated. We have explained what AHWMNs are, which different types exist, what issues need to be dealt with in AHWMN networking, and what the current state-of-the-art is in the area of AHWMN routing. Throughout this chapter, we have explained how the properties of AHWMNs define in a direct or indirect way the characteristics of the routing task in such networks. Specifically, we have formulated a number of different challenges that will need to be addressed by the algorithms developed in this thesis. The first of these is adaptivity. An AHWMN routing algorithm needs to provide adaptivity to deal with the dynamic nature of these networks. We have shown how the placement of the nodes and their movements affect the topology and its changes, and how the node density and radio transmission range influence this. The second challenge is robustness. Wireless transmissions are often unreliable, and mobility induced path failures can cause extra packet loss. AHWMN routing algorithms should be able to work correctly and provide a reliable service in such a challenging environment. This is especially important since, as we described, the traditional approach of providing service level guarantees at the transport layer encounters difficulties in AHWMNs. The third challenge is efficiency: the needed adaptivity and robustness have to be obtained in an efficient way, wasting as little as possible of the limited network resources. These resources refer in the first place to network capacity. Network capacity is mainly limited due to the need to share the wireless channel among the nodes of the network, and the difficulties that are encountered at the MAC layer when trying to organize this sharing effectively. Other limited resources are the battery power of the nodes, their processing power, etc.. Finally, the last challenge is scalability. In the flat AHWMN environment with limited capacity, AHWMN routing algorithms should provide a service that can scale to large numbers of nodes.

We have in this chapter also described a number of techniques that have been developed by different researchers in the field of AHWMN routing to deal with the above challenges. We have first described multipath routing, and have explained how it can improve throughput, adaptivity and reliability. Then, we have explained how the use of structure and organization can help scalability. Next, we have described how location information can be used in routing, and how it can increase scalability and efficiency. We have also given an overview of different path evaluation metrics that can be used, and have discussed their respective advantages and disadvantages. Then, we have discussed some routing algorithms for large scale AHWMNs, and have explained the techniques that

they use. Next, we have discussed techniques used for multicasting, geocasting and broadcasting. Finally, we have described the possibility to use data-centric routing to improve scalability and efficiency in sensor networks. Many of the techniques described in this chapter are used in the algorithms developed in this thesis, or in the algorithms we compare with.

The focus in the rest of this thesis will be on routing in MANETs and WMNs. Sensor networks have slightly different properties, as has been explained throughout this chapter, and command therefore a different approach. Nevertheless, since they are also types of AHWMNs, there are a lot of similarities with MANETs and WMNs, so that the techniques developed in this thesis can to some extent also serve in them. On a related note, we expect that also research in other types of networks could benefit from the ideas developed in this thesis. We think hereby of more traditional, wired networks, and of application layer overlay networks, such as peer-to-peer networks [243] or resilient overlay networks [20]. Also in these kinds of networks, properties such as adaptivity, robustness, efficiency, etc. can be very useful. In wired networks, for example, algorithms need to be adaptive to deal with link failures, or to change their behavior depending on the network load. In overlay networks, where nodes are connected through virtual links, the topology is at least as dynamic as in AHWMNs, so that also there adaptivity is important. On the other hand, robustness and efficiency could be less important, both in wired and in overlay networks.

Chapter 3

Adaptive routing and learning

In this chapter, we discuss existing approaches to adaptive routing. The aim is to make the reader acquainted with some important core techniques behind these algorithms, and their advantages and disadvantages, so that the choices made for the work presented in this thesis can be better understood. Where possible, we will comment on how the adaptive techniques discussed here score with respect to the important challenges in AHWMN routing that were outlined in the previous chapter, namely adaptivity, robustness, efficiency and scalability. We will also point out how and where these techniques are used in the AHWMN routing algorithms described before.

In what follows, we first discuss adaptivity in internet routing algorithms. Due to the large size of the internet, it is impossible for human operators to keep track of all changes in its network topology. Therefore, internet routing algorithms contain built-in mechanisms for adaptivity. Some of these mechanisms will be used further on in this thesis. Next, we discuss Ant Colony Optimization (ACO) routing algorithms. This is a relatively new class of routing algorithms, which was inspired by the mechanisms used by ant colonies to find the shortest path between their nest and a food source, and by the ACO metaheuristic which was derived from this. ACO routing algorithms form the main source of inspiration for the work in this thesis. In the final section, we provide a deeper discussion on those of the described techniques that are most relevant for this thesis, and on their relationship with the field of machine learning.

3.1 Adaptive routing in the internet

The internet today can be considered as a collection of subnetworks, commonly referred to as autonomous systems (ASs) [254]. For routing inside an AS, intra-domain routing protocols such as the Routing Information Protocol (RIP) [123] and Open Shortest Path First (OSPF) [197] are used. For routing between ASs,

which is called inter-domain routing, the standard is the Border Gateway Protocol (BGP) [178]. Underlying these different algorithms are two basic approaches to routing: distance vector routing and link state routing. While RIP is a direct implementation of the basic ideas behind distance vector routing, OSPF is the most important instantiation of link state routing. BGP is based on distance vector routing, but it follows a significantly different approach from RIP, in order to allow the implementation of internet policies and to overcome some of the shortcomings of RIP. Recently, a new approach to routing, ant colony optimization (ACO) routing, has been proposed as an alternative to both distance vector and link state routing. ACO routing is a class of highly adaptive algorithms, which was developed based on artificial intelligence techniques, and in particular on the ACO metaheuristic for optimization.

In what follows, we describe the working of distance vector routing and link state routing, using the mentioned internet routing algorithms as examples, and discuss advantages and disadvantages of both approaches. ACO routing is described separately later in section 3.2; since it forms the most important source of inspiration for the work in this thesis, a full separate section is devoted to it.

3.1.1 Distance vector routing

Distance vector routing is also known as Bellman-Ford routing [27] or Ford-Fulkerson routing [103], after the authors of the earliest work on this approach. Below, we first describe RIP as an example of the basic working of distance vector routing. Then, we discuss advantages and disadvantages of this approach. Next, we describe BGP, which implements distance vector routing differently from RIP. Finally, we comment on the use of distance vector routing in AH-WMNs. A short description of distance vector routing has also been provided in the description of DSDV in subsection 2.4.2.

RIP: basic distance vector routing

Distance vector routing was originally derived from the principles of dynamic programming, which is a general solution method for optimization problems proposed by Bellman in 1957 [26]. The basic idea behind dynamic programming is to split an optimization problem into subproblems, and use the solutions to these subproblems to construct an optimal solution for the main problem. The subproblems themselves are recursively split into smaller subproblems, until an elementary case is reached that is easy to solve. In routing, the optimization problem is to find the shortest path from a source node s to a destination node d. This problem is split into the subproblems of finding the shortest path from each of the neighbors of s to the destination d. Each of these subproblems is recursively split up further, until they are reduced to the simple case of finding the one hop path between one of d's neighbors and d. As an example, consider the network of figure 3.1. The shortest path from node 1 to node 7 is found by calculating the shortest path from node 2 to node 7 and that from node 3 to

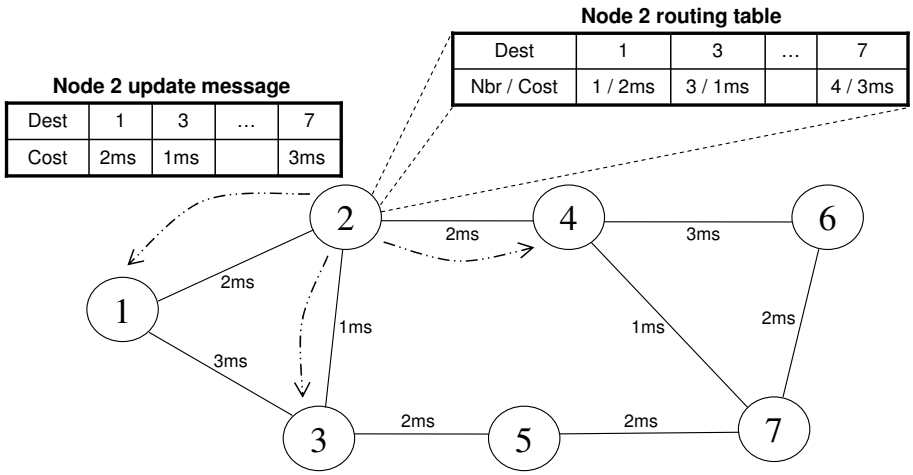


Figure 3.1: An example of the working of RIP. The figure shows the routing table maintained in node 2, and one of the update messages that are periodically sent out by this node. The trajectories of the update messages are given in dashed lines. The cost of each link is given in milliseconds.

node 7, adding the cost of the one hop link from node 1 to respectively node 2 and node 3, and comparing the results. In turn, the shortest path from node 2 to node 7 is found by comparing the shortest paths from the nodes 1, 3 and 4 to node 7, and the shortest path from node 3 to node 7 by comparing the shortest paths from the nodes 1, 2 and 5 to node 7. Finally, finding the shortest path from the nodes 4 and 5 to node 7 is trivial, since they are one hop paths.

Distance vector routing is an asynchronous, distributed implementation of this approach, and RIP is a more or less direct implementation of the basic distance vector algorithm. Under RIP, each node keeps routing information in a table that has one entry for each destination. In this entry, it stores the estimated cost of the best path to this destination, and the outgoing link over which this best path goes. In the example of figure 3.1, the routing table of node 2 is shown. Periodically, each node broadcasts to all its neighbors a route update message containing a list with the cost estimates for all destinations in its routing table. A node n which receives from its neighbor m a message that m can reach destination d with cost c_m^d, can calculate the cost c_n^{md} of going from n over m to destination d. It does this by adding together the locally observed cost c_n^m to go to its neighbor m and the reported cost c_m^d. So, in the example of figure 3.1, node 1 calculates that the cost of going over node 2 to reach node 7 is 5ms: it is the cost of taking the link to node 2 (which is observed to be 2ms), and the cost of going from node 2 to node 7 (which is reported by node 2 to be 3ms). Node n calculates in this way the cost of the path over each of its outgoing links, and chooses the best one to put in its routing table and report

in its own periodic update messages.

Starting from any initial values in the routing tables of all nodes (initialization can for example be random), this distributed and asynchronous process will eventually converge to a situation where correct routing information is available in all nodes. This means that when there are changes in the network that make the current routing information incorrect, the RIP updating algorithm will assure that the routing information is adapted and eventually reflects the new situation.

Advantages and disadvantages of distance vector routing

We investigate the behavior of basic distance vector routing as implemented in RIP in terms of adaptivity, robustness, efficiency and scalability.

As pointed out above, RIP is in principle adaptive, in the sense that its updating process is guaranteed to converge and produce routing information that reflects the network situation correctly. However, there is no bound on how long this convergence takes. In general, it is known that RIP reacts fast to good news, such as e.g. the appearance of a new link or node, but slowly to bad news, such as the failure of a link or node [254]. An extreme example is the counting-to-infinity problem. Consider the simple network of figure 3.2, where the link between the nodes 2 and 3 fails. Node 2 realizes that its 2ms path to node 4 over node 3 is no longer feasible. However, it receives from node 1 an update stating that node 1 can reach node 4 in 3ms. Node 2 is unaware that this path includes itself, and uses the information to adapt its own routing table: it registers a path to node 4, going over node 1, with a cost of 4ms. When it subsequently sends this new information out to node 1, node 1 adapts its own path to node 4 to have a cost of 5ms, and includes this new information in the next message it sends to node 2. This updating process goes on indefinitely, with both nodes slowly adapting their cost estimate of the path to node 4 to the correct value of infinity. A number of adaptations for RIP have been proposed in order to provide better adaptivity and deal with the counting-to-infinity problem. See [182] for an overview.

The robustness of RIP is related to its adaptivity. In principle, the algorithm's capacity to converge to correct routing information allows it to recover from disruptive events, loss of control packets, or the presence of erroneous information (e.g. out-of-date routing information). However, as pointed out before, this recovery process can be quite slow. This means that stale routing information can remain in the network for extended periods of time. The presence of such wrong information can lead to problems such as the formation of loops [58]. This is because nodes base their routing information on estimates provided by other nodes, and because routing decisions are taken independently in each node. An example is provided in the earlier mentioned counting-to-infinity problem: in the situation of figure 3.2, node 2 and node 3 form a routing loop between them for packets going to destination node 4 as long as they have not reached the correct cost value of infinity.

In terms of efficiency, RIP scores very well. The ability to reuse routing

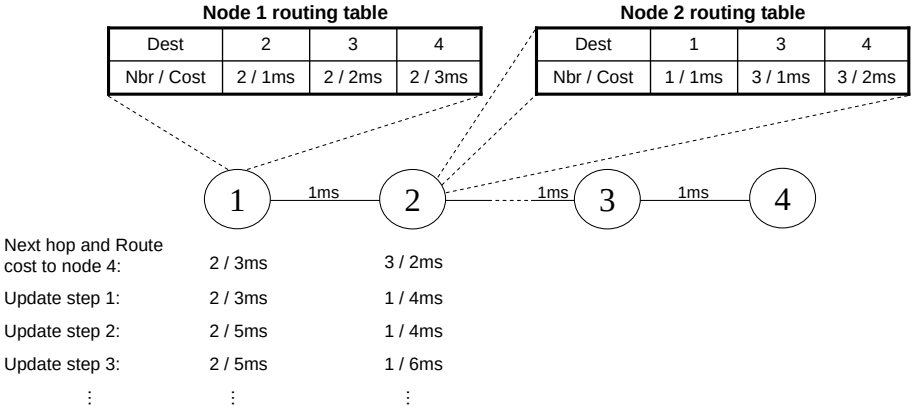


Figure 3.2: The counting to infinity problem. After the failure of the link between node 2 and node 3, node 1 and node 2 adapt their estimate of the cost of the route to node 4 based on the updates they receive from each other. Figure adapted from [254].

estimates provided by neighboring nodes allows to extract maximal information from minimal communication. Scalability, on the other hand, is often considered a big problem for distance vector algorithms. This is because nodes should include their full routing table in each periodic update message they send out.

BGP: Advanced distance vector routing

BGP is the routing algorithm used for routing between ASs in the internet. It implements distance vector routing, but in a different way than RIP. The main reason for the differences between both algorithms is the need to apply policies. At the level of inter-domain routing, politics and economics play an important role in forwarding decision. E.g., it is possible that an AS only wants to forward packets if it gets paid for it, and therefore refuses packets originating from ASs which it does not have a contract with. Or an AS of one country might want to avoid routes over ASs situated in a hostile neighbor country. BGP has been designed specifically to implement such policies.

The main difference between BGP and RIP is that under BGP, routing table entries do not just contain the cost and the outgoing link of the best path to a certain destination, but also the sequence of nodes that make up the entire path to that destination. Similarly, when a node sends an update message about its routing information to a neighbor, it includes full paths. By combining the paths for different destinations reported by all neighbors, each node can construct a view of the network, on which it bases its own routing decisions. Thanks to the availability of full path information, it is easy to implement policies such as the ones mentioned above. BGP is sometimes also called a path vector routing algorithm, since its routing table is a vector of paths, rather than distances.

In terms of adaptivity, BGP scores better than RIP. The availability of full path information allows to solve the counting to infinity problem, and to avoid loop formation [254]. Nevertheless, observations of internet behavior have shown that BGP can still be slow to converge to correct routing information after a failure, in the order of tens of minutes [160]. Also robustness remains a problem, since the loss of control packets or the presence of erroneous information still leads to instabilities (see e.g. [270] for a study of the behavior of BGP in a stressful situation). Finally, in terms of efficiency and scalability, BGP scores worse than RIP, since the algorithm is more complex, and more routing information needs to be exchanged between nodes. The growing size of routing tables in the internet has become an important problem, and different strategies have been developed to deal with it [43, 134].

Distance vector routing in AHWMNs

The first routing algorithm that was developed for AHWMNs, DSDV, is a distance vector algorithm. It follows quite closely the basic approach described for RIP. The main reasons for adopting the distance vector approach for AHWMN routing are its relative simplicity and efficiency, which are important properties in resource constrained environments such as an AHWMNs. To deal with problems that stem from slow adaptivity and low robustness, such as loop formation, DSDV contains an elegant solution based on sequence numbers. All routing table entries are labeled with a sequence number that is assigned by the destination. A node i only accepts new routing information for a destination d if the new information has a higher sequence number than what is already available in i, or when it has the same sequence number but a better routing metric. For an example of how this solves the problem of loop formation, consider again the counting-to-infinity problem depicted in figure 3.2. After the failure of the link between node 2 and node 3, updates about the route to destination node 4 sent out by node 1 or node 2 always have the same sequence number. This is because assigning new sequence numbers is the responsibility of the destination, node 4, and all connectivity to this node is lost. Moreover, the updates sent back and forth between the two nodes have increasing routing metrics, and are therefore not accepted. Details about the DSDV algorithm are given in subsection 2.4.2 and in [211].

Despite the use of the destination based sequence number mechanism, DSDV was found to be insufficiently adaptive and robust to work in AHWMNs, especially as network size or dynamism increased (see e.g. [42, 180]). Therefore, a few years later, AODV was proposed as a follow up to DSDV. AODV is labeled a distance vector algorithm, since its routing tables hold the cost and next hop of the best route for each destination, and are therefore distance vectors just like the tables used in DSDV and RIP. However, AODV's approach to gathering and updating routing information has nothing in common with the distributed algorithm described in this subsection. One could therefore say that AODV is a distance vector algorithm but not a Bellman-Ford algorithm. However, since these two terms have always been associated with each other, this can

create some confusion. For details about AODV we refer to subsection 2.4.2 and to [213].

3.1.2 Link state routing

Link state routing was in the first place developed to overcome problems of slow convergence and scalability that are present in distance vector protocols. Its basic approach to spreading and calculating routing information is fundamentally different. In what follows, we first describe OSPF, which is the main representant of link state routing, then discuss advantages and disadvantages of this approach, and finally comment on the use of link state routing in AH-WMNs. A short description of link state routing was also provided earlier, in the discussion of the OLSR protocol in 2.4.2.

OSPF

The basic working of link state routing algorithms is depicted in figure 3.3. Each node locally monitors the cost to go to each of its neighbors. This forms the node's local state. Whenever a node detects a change in its local state, it floods it to all other nodes in the network in a link state advertisement (LSA) message. LSA messages contain sequence numbers in order to distinguish old from new routing information. By combining the LSAs received from all nodes in the network, each node can get a complete view of the current network situation. This is stored in the node's topological database. Using this database, each node runs the Dijkstra shortest path algorithm [208] to construct a shortest path tree with itself as root, and uses the result to fill its routing table. In terms of the mechanism used to spread routing information, the difference with distance vector routing could be summarized as follows: while in distance vector routing, nodes send their global network information out locally to their neighbors, in link state routing, they send their local network view out globally to all other nodes.

The OSPF algorithm implements the above scheme. However, it has a lot of added features to make it better suited to the specific properties of the internet. Some of these are aimed at improving scalability. OSPF allows to split the network into so-called areas, so that inside each area, nodes only maintain routing information about that area. For routing between areas, a backbone is set up. Note that these areas are subnetworks inside the earlier mentioned ASs (which are themselves subnetworks of the internet). Another scalability related feature of OSPF is the possibility to represent a group of directly connected routers by one node in the OSPF network. To this end, one of the routers, called the designated router, acts on behalf of the whole group. This is for example relevant in LANs where all nodes are connected through an Ethernet bus; it would be inefficient to let each of these nodes run OSPF separately. Other features of OSPF, which are not related to scalability, include the fact that all message exchanges are authenticated, in order to improve security, and the possibility

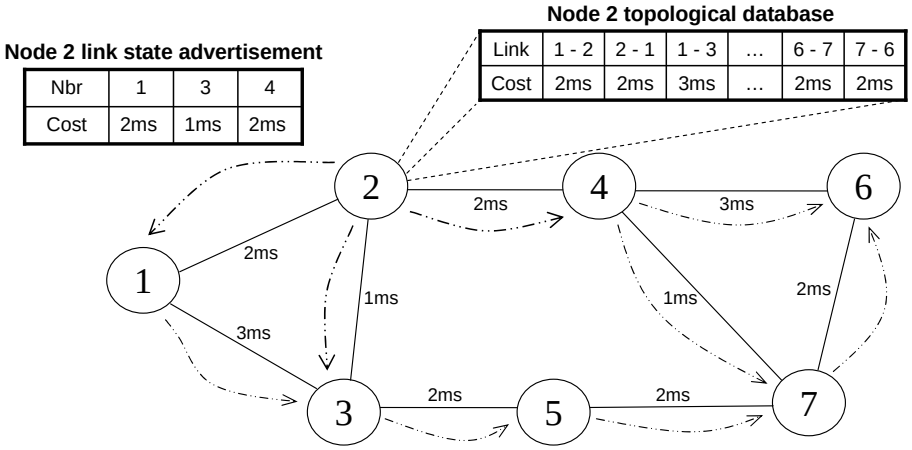


Figure 3.3: An example of the working of OSPF. The figure shows the topological database maintained in node 2 (this should be the same for each node in the network), and a link state advertisements sent out by node 2. The link state advertisements are flooded over the network, as is indicated by the dashed lines.

to route packets distinctly based on their IP Type of Service field, so that in principle differentiated Quality-of-Service (QoS) routing [153] can be supported.

Advantages and disadvantages of link state routing

Link state routing offers good adaptivity: any detected change is flooded through the network in LSAs, and as soon as the nodes in the network have received all updates, they can recalculate the shortest paths to all destinations so that their routing tables reflect the new network situation. This is faster than the slowly converging distributed updating process used in distance vector routing.

Robustness, on the other hand, can be a problem for link state routing algorithms. If LSAs get lost, or stale LSAs are used for updates, nodes end up with erroneous topological databases, and calculate wrong paths. In OSPF, a number of measures have been taken to alleviate these problems. In order to avoid the loss of LSAs, each LSA transmission needs to be acknowledged, and duplicate transmissions are done in case of failure. Also, each node periodically sends out refresh LSAs, which repeat previously transmitted information. In order to avoid the use of stale routing information, each LSA is given a sequence number, so that receiving nodes can figure out the relative age of different LSAs coming from the same source node. Furthermore, LSAs are given a timestamp, so that they can be discarded when they get too old.

In terms of efficiency, link state routing should in principle score quite well. However, the previously mentioned measures to improve robustness compromise this: extra overhead is created by LSA acknowledgements, duplicate LSAs and

refresh LSAs. Moreover, in the presence of links of unstable quality, a large number of new LSAs can be created in reaction to repeated changes in link availability [236].

Finally, also scalability might be an issue for link state routing protocols. This is slightly ironic, since the need for scalability improvement was one of the reasons for developing link state routing as a replacement for distance vector routing. However, the fact that each node needs to build a map of the entire network, and that each LSA needs to be flooded to all other nodes, can be problematic in large networks. These problems are partly alleviated in OSPF by the earlier described mechanisms of using areas and designated routers.

Link state routing in AHWMNs

The link state approach has been applied to routing in AHWMNs in the OLSR routing protocol. OLSR follows the basic ideas of link state routing in terms of the spreading of routing information and the calculation of shortest paths. It does not include any of the specific features of OSPF such as the use of areas and designated routers to improve scalability. Instead, OLSR obtains improved adaptivity and scalability through the use of multi-point relays. Detailed descriptions of this mechanism and of OLSR in general are given in subsection 2.4.2 and in [61]. Despite the limitations in terms of robustness and scalability that come with the link state approach, OLSR gives acceptable performance in AHWMNs, at least when the network is not too large or dynamic [36].

3.2 Ant Colony Optimization routing algorithms

In this section we describe ACO routing. This is a class of adaptive routing algorithms that form an alternative to the more traditional approaches to routing described above. ACO routing was originally inspired by mechanisms found in biology: it is based on principles that are present in the foraging behavior of ants in nature, and on the ACO framework for optimization that was derived from these principles. ACO routing algorithms work in a highly distributed way, and have properties such as adaptivity, robustness and scalability. This makes them particularly interesting to deal with the challenges in AHWMN routing that were described in chapter 2. ACO routing forms an important source of inspiration for the work described further on in this thesis.

In what follows, we first describe the mechanisms behind the food gathering process of ants in nature that was the original inspiration behind ACO and ACO routing. Then, in subsection 3.2.2, we present the ACO metaheuristic for the solution of optimization problems that was derived from this process and formed the basis for the development of ACO routing. Next, in subsection 3.2.3, we introduce ACO routing through the description of an example, the AntNet routing algorithm. Subsequently, in subsection 3.2.4, we provide a deeper analysis of the given example in order to identify which are the main principles and properties of ACO routing. Finally, in subsections 3.2.5 and 3.2.6, we give an

overview of existing ACO routing algorithms, respectively for wired networks and for AHWMNs.

3.2.1 Ants in nature

The main source of inspiration behind ACO and ACO routing is a behavior that is displayed by certain species of ants in nature during foraging. It has been observed that ants from e.g. the family of Argentine ants Linepithema Humile are able to find the shortest path between their nest and a food source [115]. This is remarkable because each individual ant is a rather simple creature, with very limited vision and computing power, and finding the shortest among several available paths is certainly beyond its capabilities. The only way that this difficult task can be realized is through the cooperation between the individuals in the colony.

The key behind the colony level shortest path behavior is the use of pheromone. This is a volatile chemical substance that is secreted by the ants in order to influence the behavior of other ants and of itself. Pheromone is not only used by ants to find shortest paths, but is in general an important tool that is used by many different species of ants (and also by a lot of other social animals) for a wide variety of tasks that involve coordinated behavior [131].

In the case of the path finding task we describe here, the ants use pheromone to recruit subsequent ants to the paths they have followed. Ants moving between their nest and a food source leave a trail of pheromone behind, and they also preferably go in the direction of high intensities of pheromone. We use the example situation depicted in figure 3.4 to explain how this simple behavior leads to the discovery of shortest paths. In our example, there are two possible paths between the ant nest and the food source, one of which is considerably shorter than the other. The first ants leaving the nest have no information available. They therefore choose their movements randomly. This leads to approximately 50% of the ants choosing the short path and 50% choosing the long path. All moving ants leave a trail of pheromone behind. The ants going over the short path reach the destination earlier than those going over the long path. Moreover, they can return faster. This leads temporarily to a higher pheromone concentration on the shortest path. Subsequent ants leaving the nest are attracted by this higher intensity, and go therefore preferably also over the shortest path. As this process continues, the majority of the ants eventually concentrate on the shortest path. It needs to be pointed out however, that the behavior of the ants is never deterministic, so that there will always remain a minority of ants that explore the longer path.

The use of pheromone is an example of a form of indirect communication that is often referred to as stigmergy [116, 256]. We speak of stigmergy when concurrent agents communicate through local adaptations of the environment: agents make local changes to the environment, and in turn locally sense the environment for this kind of changes. The environment's properties that are changed and sensed by the agents are called stigmergic variables. In the case of the above described shortest path mechanism, the stigmergic variable is the

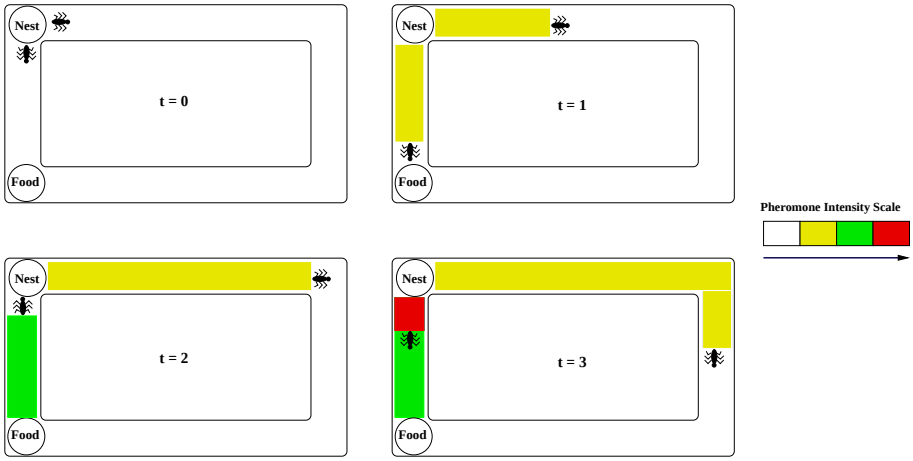


Figure 3.4: The shortest path mechanism used by ants. The different colors indicate increasing levels of pheromone intensity. From left to right and then from top to bottom, we see the situation in successive time steps. Figure taken from [70].

pheromone. Ants change the pheromone intensity locally by dropping their own pheromone, and they sense the environment in their immediate neighborhood for variations in pheromone intensity, to which they adapt their behavior. In processes such as the ant colony shortest path behavior, stigmergy is a key element to provide self-organization in a system consisting of highly independent distributed agents.

The shortest path finding process of the ants has a number of interesting properties. First of all, it is highly distributed and self-organized. There is no central control mechanism; instead, the organization of the behavior emerges from the simple rules of stigmergic communication that are followed by the individual ants. Second, it is highly robust. This is related to the property of self-organization: the system has no single point of failure, but instead consists of a high number of individually unimportant agents, so that even significant agent losses do not have a large impact on the performance. Third, the process is adaptive. Since none of the ant behavior is deterministic, and some individuals keep exploring also longer paths, the system can adapt to changes in the environment. In [115], the authors describe how ants are able to start using a new shorter path when it is presented to them at a later point in the experiment. Finally, the process is scalable: the process can be scaled to arbitrarily large colonies.

3.2.2 The Ant Colony Optimization metaheuristic

The distributed shortest path finding process of foraging ants described above has been an important source of inspiration for artificial intelligence (AI) re-

searchers. In particular, it was the basis for the development of the ACO metaheuristic [83,87]. This is a general framework for the development of algorithms to solve optimization problems. The main idea behind ACO is the use of a colony of artificial ants and a matrix of artificial pheromone. ACO algorithms work in an iterative way. In each iteration, all artificial ants build a solution to the problem at hand in parallel, using the artificial pheromone matrix. Then, the pheromone matrix is updated based on the solutions that were found. This way, the pheromone matrix reflects information about good solutions that have been found so far, and allows ants in subsequent generations to use this information when building new solutions.

The first applications of ACO were for the traveling salesman problem (TSP). An instance of the TSP is defined by a fully connected weighted graph $G = (V, E)$, where the set of vertices V corresponds to a number of cities, and the set of edges E represents the connections between the cities. With each of the edges (i, j), a distance $d(i, j)$ is associated. The distances can be symmetric (in which case $d(i, j) = d(j, i)$ for all pairs of cities i and j), or asymmetric. The aim is to find a closed tour that visits all cities exactly once while minimizing the total traveled distance. This combinatorial optimization problem is NP-hard. The TSP can very easily be seen as a shortest path finding problem, which makes it an obvious first choice for the implementation of ACO.

The first ACO algorithm that was developed for the TSP is Ant System (AS), which was originally proposed by Dorigo in his PhD thesis in 1992 [82] and first published in English in 1996 [85]. In AS, an artificial pheromone value $\tau(i, j)$ is associated with each edge (i, j). The algorithm maintains a colony of artificial ants, which build solutions using this artificial pheromone, and afterwards update the pheromone based on the quality of the solution they obtain. The algorithm works in an iterative way. At the start of each iteration, each ant is placed in a randomly chosen initial city. Starting from there, it moves from city to city, building a solution to the TSP. When choosing the next city to move to, an ant considers all cities that it has not visited yet. It picks one of these using the random-proportional rule given in equation 3.1. This rule calculates the probability $p_k(i, j)$ that ant k in city i chooses city j to move to next.

$$p_k(i,j) = \begin{cases} \frac{[\tau(i,j)]^\alpha [\eta(i,j)]^\beta}{\sum_{l \in N_i^k} [\tau(i,l)]^\alpha [\eta(i,l)]^\beta} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

In this equation, N_i^k represents the set of cities that ant k has not yet visited before reaching city i. $\eta(i, j)$ is equal to $1/d(i, j)$, the inverse of the distance between i and j. It serves as a heuristic value that helps guiding the construction of solutions. Using the rule of equation 3.1, the probability of choosing city j after city i increases when the pheromone between i and j is higher and when the distance between i and j is lower. The parameters α and β define the relative weight given to respectively the pheromone and the distance heuristic in the decision process: with $\beta = 0$, the decision is purely based on the pheromone, meaning the experience gathered in previous iterations, while with

$\alpha = 0$, the decision is purely based on the heuristic, so that AS comes down to a randomized local search.

At the end of each iteration, the solutions constructed by the ants are evaluated, and the pheromone values are updated. Pheromone updating includes pheromone evaporation and pheromone deposition. Pheromone evaporation refers to the decrease of all pheromone values. It is done using equation 3.2. In this equation, $0 \leq \rho \leq 1$ is the pheromone evaporation rate. Pheromone evaporation allows to forget old solutions. Pheromone deposition refers to the increase of pheromone on edges that have been used in the solutions constructed by the ants. It is done using equation 3.3. In this equation, m is the total number of ants, and $\Delta\tau(i,j)^k$ is the inverse of the cost of the solution constructed by ant k if edge (i,j) is part of this solution, and is 0 otherwise. Pheromone deposition serves to reinforce good solutions.

$$\tau(i,j) = (1-\rho)\tau(i,j), \quad \forall (i,j) \in E \tag{3.2}$$

$$\tau(i,j) = \tau(i,j) + \sum_{k=1}^{m} \Delta\tau(i,j)^k, \quad \forall (i,j) \in E \tag{3.3}$$

The algorithm is run until a given number of iterations is reached, or until no solution improvement has been obtained for a number of iterations.

AS was found to work well for small instances of the TSP, but failed to compete with state-of-the-art methods on large instances. However, its publication raised a general interest in the approach, and inspired the development of a number of similar algorithms for the TSP that were more powerful and did manage to provide state-of-the-art performance. These algorithms include Elitist AS [85], Ant-Q [105], Ant Colony System (ACS) [84] and MAX-MIN AS (MMAS) [245]. All of these algorithms are based on the same basic principles as AS: a number of artificial ants each build their own solution to the TSP. They do this in a constructive way, starting from a random initial city and adding new cities until a full solution is reached. Each decision to add a new city is made stochastically, using probabilities that depend partly on a pheromone value and partly on a heuristic value. Pheromone is updated according to the quality of the solutions provided by the ants. The main difference between these new algorithms and the original AS lies in the balance between the exploitation of information that has been learnt so far and the exploration of new possibilities. In general, AS's successors are more aggressive to exploit. For example, in Elitist AS, pheromone updating is only done for the ants that found the best solutions in the current iteration, and for the best solution found so far over all past iterations. In ACS, exploitation is also increased by using the selection rule for cities (equation 3.1) deterministically in a certain percentage of cases. Also, AS's successors provide mechanisms to balance exploration and exploitation, so that the algorithms can be better tweaked for the problem at hand. Apart from these differences, some of AS's successors, namely ACS and MMAS, have been combined with local search: to each of the solutions found by the ants, a local search procedure is applied to bring it to a local optimum. Then, pheromone

is updated based on the improved solution. This hybrid approach was found to be very powerful.

After the applications to the TSP, ACO has in recent years also been adapted for the solution of a wide range of other problems. These include the quadratic assignment problem [107, 184], the vehicle routing problem [106], the graph coloring problem [63], the shortest common super-sequence problem [193], the multiple knapsack problem [168], the bin packing problem [99, 169], the 2D HP protein folding problem [239], etc.. Most of these problems are very different from the TSP and have a structure that is much less easy to reduce to a shortest path finding problem. ACO algorithms for these problems use therefore often quite different approaches, which is mainly reflected in the way pheromone is stored, updated and used. The core ideas always remain the same however: to produce multiple parallel solutions in each iteration, using a high degree of stochasticity, and to store information about previously found good solutions in an artificial pheromone matrix, which is used to produce new solutions in subsequent iterations. Overviews of applications of ACO can be found in [34, 86, 87].

ACO has also been applied to networking problems. Some of these are off-line combinatorial problems, such as the problem of routing and wavelength-allocation in an optical fibre network [202] or the problem of finding disjoint paths in a telecommunication network [267]. ACO algorithms for these problems follow a similar pattern as the other ACO algorithms for combinatorial problems described earlier. A very different problem is that of adaptive routing in telecommunication networks. This is an online dynamic problem: the properties of the problem change continuously and the optimization algorithm has to adapt its solution online. Applications of ACO for routing, such as Ant-Based Control (ABC) [235] and AntNet [71], are therefore very different from those for static off-line problems. The rest of this section is dedicated to the description of these algorithms.

3.2.3 AntNet: an ACO algorithm for routing in telecommunication networks

A number of different ACO algorithms for adaptive routing in networks have been proposed. The first of these were ABC [235] for connection oriented telecommunication networks, and AntNet [70, 71] for packet switched data networks. Both of these algorithms were developed for wired networks. Here, we describe AntNet as a prototype example of ACO routing.

In AntNet, every node in the network keeps two data structures: a routing table and a local traffic statistics table. This is illustrated in figure 3.5. The routing tables contain the artificial pheromone, and are therefore also called pheromone tables. Pheromone tables contain for every destination a vector with one entry per outgoing link. The entry T_{ij}^d of node i's pheromone table T_i contains the pheromone value τ_{ij}^d, which is a floating point number indicating the relative goodness of taking outgoing link j on the way to destination d.

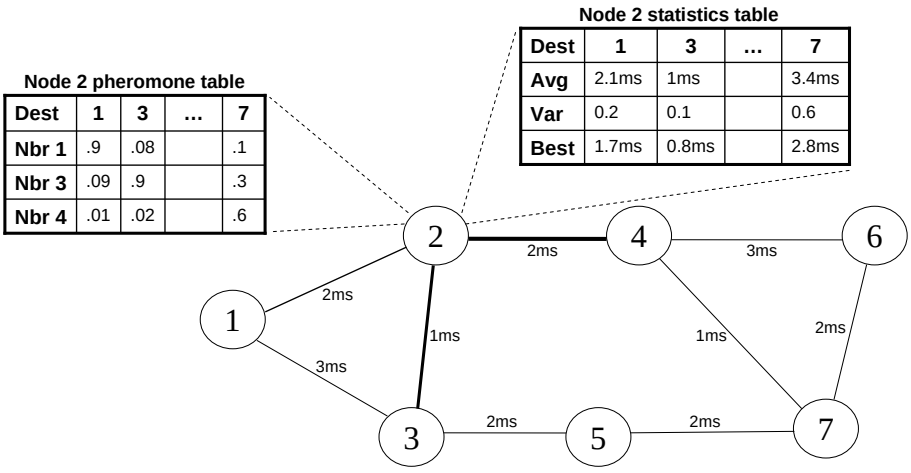


Figure 3.5: Data structures for the AntNet routing algorithm. We show for node 2 the pheromone routing table, with one row for each next hop and one column for each destination, and the local traffic statistics table, with the average, variance and best value for the trip time to each destination. The values given along each link are the expected delay of a data packet on this link. The line thickness for the outgoing links of node 2 indicate different preferences for ants going towards destination 7.

Pheromone values are normalized per destination. The local traffic statistics tables contain three entries for each destination. The first two keep a moving average of the estimated average and variance of the trip time to the destination. The third keeps the best trip time experienced over a moving observation window. The statistics are updated using the traffic times experienced by ants, and are used to evaluate the trip time experienced on new routes.

Every node s in the network sends small control packets out at regular intervals, to a randomly chosen destination d (the probability of choosing a particular destination d depends on the amount of data traffic that is currently being generated in the node for d). These control packets play the role of artificial ants, and are called forward ants. They are similar to probing packets. Their aim is to find a path to the destination and evaluate it. While traveling to d, the forward ant records the times that elapse going from node to node until the destination. As the forward ants are placed in the same queues as data packets, these time recordings are similar to the delays experienced by data packets.

The route chosen by each ant is the result of stochastic routing decisions taken at every hop: in each intermediate node, the ant chooses a next hop according to the rule of equation 3.4. This rule gives the probability that an ant in node i chooses node j as next hop on its way to d. It is partly based on the pheromone value τ_{ij}^d, which represents information learned from previous ants,

and the heuristic value η_{ij}, which is based on the length of the queue for the link to j in node i. The role of η_{ij} is to provide an estimate of local traffic, in order to help guide the ant routing process. The value α in equation 3.4 is a parameter used to balance the relative importance of the pheromone and the heuristic value in the routing decision, and $|N_i|$ is the number of neighbors, used in the denominator for normalization purposes. Using this approach, forward ants build paths like the artificial ants in AS build solutions to the TSP: constructing the path hop by hop in a stochastic way using both pheromone and heuristic information. Through the use of probabilistic decision making, different paths are explored and made available for data routing.

$$P_{ij}^{d} = \frac{\tau_{ij}^{d} + \alpha\eta_{ij}}{1 + \alpha(|N_i| - 1)} \tag{3.4}$$

Once an ant reaches its destination, it turns around and becomes a backward ant which returns to the source node s. Backward ants do not use the same queues as data packets, but high priority ones (in order to distribute the new information quicker), and do not take probabilistic routing decisions, but follow the exact reverse path of their associated forward ant. When a backward ant arrives in an intermediate node i on its way back to s, it updates the entries in i's tables for destination d. First, the traffic statistics table is updated using the trip time t experienced by the ant: the average and variance are updated using moving averages, while the best value is updated using a moving window. Then the pheromone entry τ_{ij}^{d} is updated, with j being the neighbor over which the backward ant arrived in i. One important problem is how to define how good the trip time t of the newly received ant is. To this end, the traffic statistics are used: using equation 3.5, a reinforcement value r is derived, which reflects the relative goodness of the new route from i to d over j compared to what has been observed so far.

$$r = c_1 \left(\frac{W_{best}}{t} \right) + c_2 \left(\frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (t - I_{inf})} \right) \tag{3.5}$$

In this equation, W_{best} is the best trip time over the moving window, as recorded in the statistics table, so that the first term of equation 3.5 reflects how well the new time t scores with respect to the best known time. I_{inf} and I_{sup} are estimates of the lower and upper limits of an approximate confidence interval for the mean of the trip time to d. They are based on the average, variance and best trip time recorded in the statistics table. The second term of the equation reflects how well the new trip time scores with respect to these limits, so that equation 3.5 takes into account the variability in past measurements of trip time. It is meant as a correction for the first term. The parameters c_1 and c_2 allow to balance both parts of the equation. The result of the equation is the reinforcement value r, which, after the application of a further modification function (for which we refer to [71]), is used to update the pheromone value τ_{ij}^{d} through a moving average. This way, the new pheromone value incorporates the latest reported trip time. If enough ants are used, all pheromone values can

be kept up-to-date so that they give an accurate image of the current network situation.

Data packets are routed in a similar way as forward ants, choosing a next hop stochastically at every hop. Stochastic forwarding of data packets allows per packet data load balancing. The probability R_{ij}^d for a data packet in node i with destination d to take node j as a next hop is given in equation 3.6. Like the probability for forward ants, it is based on pheromone values. However, there are two important differences. First of all, the local heuristic η_{ij} based on the queue lengths is not used here. Second, the pheromone is raised to a power ϵ, which is normally larger than 1. This increases the probability of taking paths with higher pheromone values, so that data packets are only routed over the best paths, and are not used for exploration like the forward ants.

$$R_{ij}^d = \frac{(\tau_{ij}^d)^\epsilon}{\sum_{l \in N_i} (\tau_{il}^d)^\epsilon} \tag{3.6}$$

In simulation studies using the Omnet++ simulator [265], AntNet was compared to traditional routing algorithms such as OSPF and distance vector routing, and to other adaptive routing algorithms, such as Q-routing (see subsection 3.3.3 and [39]). In a wide range of test scenarios, using different networks such as NSFNET and the Japanese NTTnet, and using different patterns in terms of data load, AntNet was shown to be superior to all other algorithms in terms of packet throughput and end-to-end delay. It did produce more overhead than traditional static routing algorithms, but not more than other adaptive routing algorithms.

3.2.4 ACO routing principles

In the previous subsection, we took a detailed look at AntNet, an example of an ACO routing algorithm. Now, we take a more general point of view, and extract the main principles behind ACO routing with their properties. The material presented here has also been described in [76], in a slightly different form.

A first important characteristic of ACO routing algorithms is that they gather routing information through the repeated sampling of full paths. This is in line with the behavior of ants in nature, where a large number of ants continuously move between their nest and the food source, and with the working of ACO algorithms for combinatorial optimization, where multiple artificial ants repeatedly and in parallel construct sample solutions for the problem at hand. The ACO approach to routing is quite different from the distance vector approach, where routing information is derived from information provided by neighboring nodes, and from the link state approach, where routing information is calculated based on the update messages received from all other nodes in the network. Due to the use of redundant probing packets such as the forward ants in AntNet, the ACO approach to the gathering of routing information is quite robust: each ant is individually unimportant and loss of control packets can be tolerated. An important aspect of the whole process is the fact that

ants always sample full paths between source and destination. At no point is routing information derived from information provided by other nodes. This is in stark contrast to the distance vector approach, where all routing information is derived from estimates provided by neighbor nodes. Relying only on direct experiences from full path samples adds further to the robustness of the algorithm, as updates are not dependent on the correctness of the routing information available in other nodes. More about this will follow also in subsection 3.3.2. On the downside, the constant use of probing also leads to increased overhead, which can negatively influence the efficiency of ACO routing.

A second characteristic is the way ants choose a path to sample: they construct the path hop by hop in a stochastic way using pheromone information. The relation with the foraging behavior of ants in nature and with ACO for combinatorial optimization is again evident, as also there ants follow their path or construct their solution step by step using pheromone in a non-deterministic way. The use of the pheromone information allows to build on experiences gathered by previous ants. It is the key to the stigmergic process that guides the coordinated working of a highly distributed process. The fact that ants build their paths in a probabilistic way allows the exploration of multiple paths. This makes the algorithm adaptive to changes in the network environment. Moreover, it leads to the availability of multiple paths for data routing, each with an associated goodness value. This increases robustness, through the availability of backup paths, and allows to spread data over the multiple paths, increasing network throughput (see also subsection 2.4.3).

A third characteristic is the stochastic forwarding of data packets based on the pheromone information. This relates to the issue of using multiple paths discussed above. By forwarding data probabilistically, the data load is spread over the multiple available paths on a per packet basis. This allows to make better use of available network resources and obtain better throughput. The use of pheromone in this process ensures that data is focused on the best paths. If pheromone is always kept up-to-date, by using sufficient ants, data load balancing automatically follows the changes in the network. An important aspect in the stochastic forwarding of data is that it uses a different formula than the ants, focusing more on the best pheromone. This way, ants are more explorative, while data packets concentrate on exploiting the routing information provided by the ants. Here, the relation with ant foraging behavior in nature and ACO for combinatorial problems is less obvious: in those processes, the same ants need to take care of exploration of the solution space and exploitation of previously found information in order to find the best solution. Using separate mechanisms for exploration and exploitation allows to build a more flexible system.

3.2.5 Existing ACO routing algorithms for wired networks

Apart from the AntNet algorithm described in subsection 3.2.3, a wide range of different ACO routing algorithms have been proposed. Here, we give an overview of existing ACO routing algorithms for wired networks, while in the next subsection, we discuss ACO routing algorithms for AHWMNs.

ABC [235] was developed before AntNet, in 1996, and takes a significantly different approach, mainly because it was developed for a different type of networks: it assumes a network with symmetric path travel costs, in which data communication is organized through virtual circuits (rather than using the packet switched approach supported by AntNet). Like in AntNet, each node s periodically sends out ants to randomly chosen destinations. Each ant has an associated age, which is increased proportionally to the load of each visited node (the age is increased more when heavily loaded nodes are passed, so that it reflects the free space on the followed route). While traveling from its source s to its destination d, the ant updates the pheromone for the path backward to s, based on its age. This is an important difference with AntNet: ants update pheromone about the path to their source while going forward, and no backward ants are used. This is possible because of the assumption of symmetric path costs. Other major differences compared to AntNet are that no path statistics are used to evaluate path quality measurements reported by the ants, and that no local heuristic is used to help guide the ants (in AntNet, the local queue lengths are used). Finally, in ABC, it is not data packets that are routed according to the pheromone, but call setup messages. Moreover, these messages do not follow pheromone probabilistically, but greedily choose the directions with the highest pheromone level. Once a call has been set up successfully, data packets follow its circuit deterministically. ABC was tested in simulation on a model of the British Telecom network and was shown to give superior performance compared to other approaches.

The algorithm proposed in [248], called Ants Routing, builds on ABC. It is meant for networks with frequent node and link failures. The main difference compared to ABC is the use of so-called uniform ants. These are different from the regular ABC ants in the sense that they do not have a specific destination and do not follow pheromone information. Instead, they wander through the network choosing each next hop according to a uniform distribution, until they have reached a maximum time-to-live, after which they are discarded. The use of uniform ants improves exploration. This is particulary important when dealing with frequent topology changes; the algorithms AntNet and ABC were designed to be adaptive with respect to data traffic changes, rather than with respect to topology changes. Another advantage of the uniform ants is that they do not have a specific destination, so that they can be used in case a node does not know all possible destinations in the network. A disadvantage is that uniform ants can lead to inefficiencies, due to the overhead they cause and the suboptimal paths they follow.

A number of papers propose further adaptations to ABC. The algorithm proposed in [35] allows ants in ABC to update pheromone not only for their source node s, but also for all intermediate nodes on their path. The authors of [226] describe ABC-backward, which combines ABC with elements from AntNet, such as the use of forward and backward ants and the use of the ants' trip time for pheromone updating. In [230], the authors extend ABC with probabilistic routing of call setups and the use of anti-pheromone, which allows ants to decrease pheromone in some cases, instead of increasing it.

AntNet-FA [72] is an adaptation of AntNet, proposed by the same authors of the original algorithm. It is very similar to the original AntNet, but contains an improvement in the behavior of the forward ants: AntNet-FA's forward ants do not use the same queues as data packets, but instead take high priority queues like backward ants. The trip times experienced by the forward ants are therefore no longer representative for what can be expected for data packets; the trip time for data packets are instead calculated by the backward ants as the sum of local estimates maintained in each of the intermediate nodes. The main advantage of this approach is that ants can travel faster so that updates are done more in real-time.

Other papers propose further improvements to AntNet. In [80] and [205] some mechanisms to enhance the exploratory behavior of AntNet are presented. In [23], the authors propose other improvements to AntNet such as the possibility to explicitly take link and node failures into account, and a better initialization of the pheromone tables. In [149], adaptive-SDR is proposed. The main difference with AntNet is that in adaptive-SDR, the network is divided into clusters, and a distinction is made between inter-cluster and intra-cluster routing. This improves scalability, since routing tables do not have to maintain entries for all possible destinations. Scalability issues of AntNet were also investigated in [51]. Finally, in [286], the authors present Adaptive Swarm-based Routing (ASR). Differences with AntNet include the use of a momentum term in pheromone updating, and the fact that pheromone is updated for all intermediate nodes as destinations. In simulation, ASR was shown to outperform AntNet in terms of packet delay and throughput.

In [274] and other papers from the same authors, Routing By Ants (RBA) is proposed. It uses virtual circuits and supports both unicast and multicast routing. RBA has similarities both with ABC and AntNet. An interesting difference with these two algorithms is the fact that the parameters which define how routing decisions are derived from pheromone values are carried inside the ants, so that they can be different for each ant. These parameters are assigned to ants in their source node and are calculated using a genetic algorithm (GA). Some small improvements to this algorithm were proposed by different authors in [249].

The authors of [128] propose the Co-operative Asymmetric Forward (CAF) mechanism. It shows how forward ants can update routing information about the path to their source without sending a backward ant and without assuming symmetric path costs. In CAF, each data packet hopping from a node i to a node j leaves at j an estimate c_{ij} of the queuing and transmission time it has experienced while traveling from i. Later, ants traveling in the opposite direction read this estimate and store it. By adding all estimates over their forward path from their source node s till intermediate node i, the ants obtain a good estimate of the delay on the backward path from i to s, so that they can correctly update pheromone information about their full backward path.

ACO has also been applied to QoS routing. A first example is AntNet+SELA [78], an adaptation of AntNet for QoS routing in asynchronous transfer mode (ATM) networks. It integrates AntNet with stochastic estimator learning au-

tomata (SELA) [266], a framework for QoS provisioning in ATM networks that uses static reinforcement learning agents to derive routing and application admission strategies. The original SELA uses a link state approach to gather routing information. AntNet+SELA, on the other hand, uses ant based probing for the collection of routing information. An interesting feature of AntNet+SELA is that nodes have the possibility to reactively send out extra ants in order to search specific information that they need. For details about the AntNet+SELA system, we refer to [70, 78]. A number of other adaptations of ACO routing for QoS have also been proposed. Most of these aim to provide hard QoS guarantees, following the IntServ approach to QoS [40]. This is the case for Agent-based Routing System, proposed in [206], and for Q-Colony, proposed in [253]. Other approaches, such as the algorithm proposed in [192] and AntNet-QoS, presented in [50], combine ACO with a soft approach to QoS routing. The latter proposes an integration of AntNet with the DiffServ framework for QoS [153].

Recently, ACO routing has also been used for routing problems in new kinds of networks (other that the applications to the equally new field of AHWMNs, which are described separately in subsection 3.2.6). One interesting example is its use for the problem of dynamic routing and wavelength assignment in wavelength-division multiplexing (WDM) networks, where the aim is to set up a primary path between source and destination and one or more disjoint backup paths, in on-demand fashion. This work is described in [204] and other papers by the same authors. Another very interesting new application is the use of ACO routing in the domain of Networks-on-Chip (NoC). These are submicron scale networks that connect the elements on an integrated circuit. NoC have many characteristics in common with traditional wired networks, but pose additional challenges that mainly arise from the extremely limited available resources. In [64], the authors propose an AntNet based approach to routing in NoC.

Finally, we want to mention some algorithms that use biological metaphors that are different from the ant foraging behavior, but reference ACO routing and have elements in common with this approach. In [273], the BeeHive algorithm is proposed. This algorithm is inspired by the behavior of honey bees. Like AntNet, however, it gathers routing information using path probing packets (called bee agents here), and it builds stochastic routing tables for data forwarding. Different from ants in AntNet, bees are flooded (with a maximum number of hops) instead of unicast along a stochastically chosen path to a specific destination. Also different from AntNet, the network is divided into regions, so that not all destinations need to be put in the routing table of each node and better scalability can be provided. The authors of [173] also address this issue of scalability in AntNet. They propose GA-agents, which is based on the use of a distributed GA. In GA-agents, each node maintains a GA population, in which each individual represents a path in the network. Paths are encoded as a sequence of turns (rather than a sequence of nodes). Individuals are evaluated by letting them probe the path they represent. This way, they are similar to the ants in AntNet. Typical GA operations such as mutation and selection are executed to find the best paths. Finally, in [276] and other papers by the

same authors, the CE-ants approach is presented. This algorithm is inspired by the cross-entropy (CE) metaheuristic for combinatorial optimization proposed by Rubinstein [228]. The CE method is in principle quite different from ACO, since it was originally derived from techniques for rare event simulation. However, due to their use of repeated sampling, many CE algorithms have in practice strong similarities with ACO algorithms. The same is true for CE-ants: in overall architecture it is quite similar to ACO routing algorithms like AntNet. The main difference lies in the formulas used for pheromone updating, which bear the signature of the CE method. CE-ants has been applied to a variety of routing related problems, such as the problem of finding protection cycles [277], and the problem of finding primary and backup paths [275].

3.2.6 Existing ACO routing algorithms for AHWMNs

Due to its properties of adaptivity and robustness, ACO has also attracted attention as a paradigm for routing in AHWMNs. Here, we give an overview of algorithms that have been proposed in recent years.

A number of the ACO routing algorithms for AHWMNs that have been proposed have a structure that is quite similar to that of the ACO algorithms for wired networks described in subsection 3.2.5. The Accelerated Ants Routing algorithm described in [104, 191], is derived from the Ants Routing algorithm for wired networks. It contains small adaptations to this algorithm, such as the no-return rule (which simply states that ants cannot pick their previous hop as next hop, so that simple loops are avoided), and is shown in simulation to perform better than AntNet in MANETs. The ABC-AdHoc algorithm [255] on the other hand is based both on ABC and AntNet. While it uses forward ants that update pheromone for the path to their source, as in done in ABC, it uses formulas of AntNet to calculate pheromone updates and to make probabilistic routing decisions. In a simulation with rather limited mobility, the ABC-AdHoc algorithm was shown to perform better than AntNet. In [288], the authors propose adaptations of AntNet for use in sensor networks. These include an informed initialization of pheromone values at node activation, the possibility to flood forward ants, rather than unicast them to their destination, and to piggyback these flooded forward ants on top of data packets. The adaptations of AntNet are shown to perform better than the original algorithm in sensor networks.

The problem with following the design of ACO routing algorithms for wired networks too closely is that it results in proactive routing algorithms, which, as has been explained in subsection 2.4.1, is not always the best approach to routing in AHWMNs. This problem and a solution to it are well illustrated in [24]. In this work, the authors first propose an algorithm that is very similar to AntNet. The main differences are that some uniform ants are used to improve exploration (see also the Ants Routing algorithm described in subsection 3.2.5), and that data are routed deterministically over the path with the best pheromone. In simulation tests, this algorithm was found to perform worse than AODV, mainly due to inefficient route discovery and large amounts of overhead. Then,

the authors propose a new algorithm, called Probabilistic Emergent Routing Algorithm (PERA). This is a purely reactive algorithm: forward ants are only sent out at the start of a communication session, or when all existing routing information is out of date. They are flooded towards the destination. For every copy of the forward ant that reaches the destination, a backward ant is sent to the source, so that multiple paths are created at route setup. It is not clear whether data are routed stochastically or not. In simulation studies, PERA is found to have a performance that is comparable to AODV. Unfortunately, it is also clear from the description of the algorithm that this is mainly due to the fact that PERA is quite similar to AODV, with forward ants and backward ants replacing respectively RREQ and RREP messages. Of the original ACO ideas, not much is left.

The approach of building a reactive kind of ACO routing algorithm has been followed by several other researchers in the field. Ant-Colony-Based Routing Algorithm (ARA) [119] is quite similar to PERA (and hence to AODV). One difference is that both forward and backward ants leave pheromone behind: forward ants update pheromone about the path to the source, while backward ants update pheromone about the path to the destination. Another difference is that also data packets update pheromone, so that paths which are in use are also reinforced while the data session is going on. This comes down to repeated path sampling, so that ARA keeps more of the original ACO characteristics than PERA. In simulation, ARA was found to perform better than AODV but worse than DSR in highly dynamic environments. Also the Termite algorithm [224] follows a reactive approach. Important differences with ARA and PERA are that forward ants are not flooded, but follow a random walk. And backward ants do not necessarily follow the exact same path of the RREQ back to the source, but are themselves routed stochastically (this can be an advantage if unidirectional links are present; see subsection 2.3.2). Pheromone updating is done by all packets (also by data), and is always done with respect to the source that a packet is coming from. This means that cost symmetric paths are assumed. In a small set of simulation tests, Termite was shown to perform better than AODV for varying values of node speed [225]. Ad hoc Networking with Swarm Intelligence (ANSI) [220] is again a variation on the same approach. It only uses ants at route setup time. A mechanism using forward and backward ants is applied, and like in Termite and ARA, data packets also deposit pheromone, in order to reinforce the paths they use. Data are routed deterministically over the best paths. ANSI evaluates paths based on the congestion rates (defined by the free versus occupied space in the node IP queues) of nodes along the path. ANSI was shown to perform better than AODV in simulation. Finally, also Emergent Ad Hoc Routing Algorithm (EARA) [177] follows a similar approach. Different in EARA is that during the course of a data session, paths are reinforced by ants that are sent out from the destination. New paths are detected using ants that do random walks through the network.

A number of other algorithms use ants in a different way. The authors of [194] propose to use a set of mobile agents that are quite independent from network nodes or data sessions: these agents are generated at network setup time, and

they stay around indefinitely. They perform a continuous random walk through the network, keeping a history of the last N nodes they have visited. At each new node they arrive, paths are extracted from this history list in order to update routing information. The algorithm proposed in [190] is a combination between this approach and AODV. Also here, a number of ant agents keep going through the network indefinitely, performing a random walk and keeping a history list of the last N visited nodes. However, here the network nodes also run AODV. They are therefore not solely dependent on the information given to them by the randomly moving ants. Instead, the ants only provide possibilities to improve on routing information gathered by AODV, or to avoid AODV route setups. The authors show that their Ant-AODV method performs better than Ants Routing or AODV separately. In [45], the authors propose an algorithm that combines geographic routing (see subsection 2.4.3) with ants. In their approach, each node keeps a database of the positions of all other nodes. To keep these databases up-to-date, information is exchanged locally among neighbors, and globally by sending ants to nodes further away. The destination nodes for the ants are chosen according to a probabilistic mechanism, and nodes can copy the routing information from ants that pass by. So, in this approach, ants replace flooding as a way to spread routing information over the network. The Mobile Ants-Based Routing Protocol (MABR),proposed in [126, 127] was designed for WAN scale AHWMNs. The algorithm divides the AHWMN area in rectangular zones, corresponding to geographical areas. All nodes of a zone together make up a logical router. Long distance routing is done between logical routers, with the aid of location information. Ants are used at this level, to proactively update routing tables between logical routers. In simulation, MABR compared favorably to Terminodes routing, a different algorithm for WAN scale AHWMNs.

Finally, a number of ACO routing algorithms for QoS routing in AHWMNs have also been proposed. Ant-based Distributed Routing Algorithm (ADRA) [289] follows a reactive approach, similar to the PERA algorithm described above. A difference is that, in order to support QoS, nodes check resource availability before they forward a forward ant, so that paths are only set up when their QoS requirements can be met. In case available resources change and an existing path can no longer rely on the necessary resources, nodes send so-called anti-ants to erase the path and inform downstream nodes that they need to find a new path. In simulations, ADRA was found to outperform the DSR routing algorithm, especially in highly dynamic scenarios. Ant colony based Multi-path QoS-aware Routing (AMQR) [176] uses ants to set up multiple, link disjoint paths. The source node stores information about the paths followed by different ants, and combines it to construct a topology database for the network. Based on this database, it calculates n different link disjoint paths, and it send data packets over these different paths. These data packets update pheromone. The use of a topological database is different from most other ACO routing algorithms, but can also be found in the AntNet+SELA algorithm for QoS routing in wired networks (see subsection 3.2.5). It allows the source node to have better control over the paths that are set up. In simulation tests, AMQR

was shown to outperform ADRA and DSR, especially in low mobility scenarios.

3.3 Routing and machine learning

In this section, we investigate the relationship between routing and the field of machine learning. We show that the problem solved by a distributed routing algorithm fits well into the description of reinforcement learning problems. This allows to investigate the learning algorithms behind several existing routing algorithms in a unified framework. The aim is to view these algorithms from a machine learning point of view, and to provide the reader with the necessary foundation to follow some of the discussion further on in this thesis. The focus will be on the distance vector and the ACO routing algorithms described before, as these have the most relevance for the work presented later in this thesis.

In what follows, we first describe the reinforcement learning framework. Then, we discuss two elementary solution methods in this area, namely dynamic programming and Monte Carlo sampling, and we show how these methods relate to the distance vector and ACO routing algorithms respectively. We end with a description of temporal difference learning and Q-routing. Temporal difference learning is a more advanced solution method for reinforcement learning problems that combines elements from dynamic programming and Monte Carlo sampling, while Q-routing is an adaptive routing algorithm for wired networks that was directly inspired by one of the most popular a temporal difference algorithms, namely Q-learning.

3.3.1 The reinforcement learning framework

Reinforcement learning (RL) [252] is the name of a class of problems in machine learning. Generally speaking, one can say that it refers to the task of learning actions by trail-and-error in order to maximize a reward. RL is a relatively new topic of research. Its characteristic trait of learning from experience sets it apart from the traditional distinction in machine learning between the classes of supervised learning and unsupervised learning problems. In supervised learning, the aim is to learn from examples of correct behavior, while in unsupervised learning, one searches for any kind of pattern that seems interesting without any clear guidance (what is learned depends on the inductive bias present in the learning algorithm). An example of a RL problem is given by the gridworld of figure 3.6. A learning agent A enters the gridworld at position S. It can move around to different positions by taking one of four possible actions: north, south, east or west. The agent receives a reward of 0 in all of the positions, except for position G, where it receives a reward of 10. After reaching position G, the agent is automatically moved back to S. The agent should learn the optimal action to take in each of the different positions, so that it maximizes its total reward.

Let us now provide a more formal description of the RL problem framework. A graphical presentation is given in figure 3.7. Central to each RL problem is

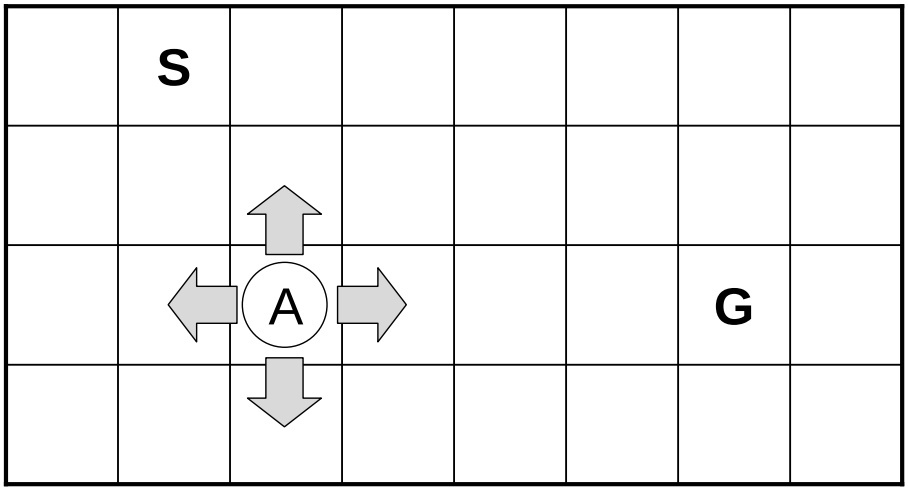


Figure 3.6: An example of a reinforcement learning problem. A learning agent A enters the gridworld at position S. It can move to different positions by taking one of four possible actions: north, south, east or west. The reward is 0 in each position, except for position G, where the reward is 10. After reaching position G, the agent is automatically moved back to S. The agent should learn by trial and error to find which movements maximize its total reward.

the interaction between an agent and its environment. The agent is a learner and decision maker. The environment is defined as the system that the agent interacts with; this includes everything outside the agent. At each moment in time, the agent finds himself in a certain situation in its environment. Such a situation is called a state. In the example of figure 3.6, each position of the gridworld forms a state. The environment provides the agent with information about its state. In each state, the agent selects an action. Based on this action, the environment provides the agent with a new state and with a corresponding reward. The aim for the agent is to learn which actions should be taken in which states in order to maximize the total reward. This includes current and future rewards, since actions often do not give immediate rewards, but do bring the agent to states which allow it to receive a better reward later. Future rewards are usually discounted so that actions that lead to these rewards faster are preferred.

In order to solve the learning task, the agent maintains a policy π, which indicates for each state s the probability $\pi(s, a)$ of taking each possible action a. By trying out different actions in the different states, and observing the (possibly delayed) rewards that follow the action, the agent learns the policy π^* that maximizes its total amount of reward in the long run in the environment. This learning is always an iterative process, in which the policy is improved in each iteration. Most solution methods make use of a state value function V or

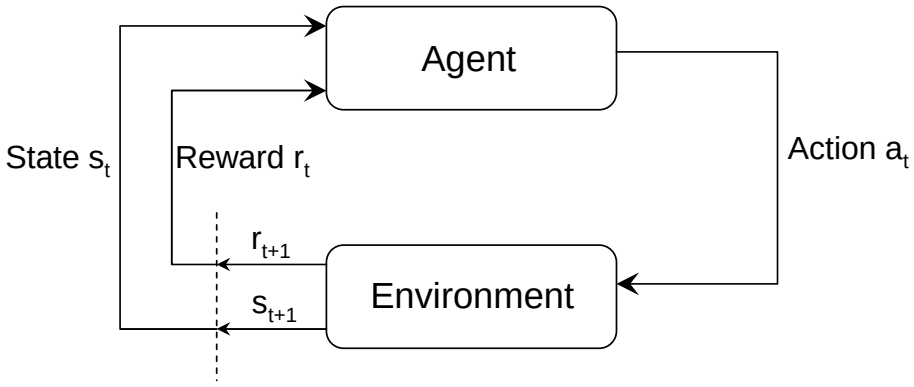


Figure 3.7: A formal model for the RL problem. The agent interacts with its environment. At time step t, the agent is in state s_t in the environment, and receives the corresponding reward r_t. In response, it takes action a_t. The environment, in turn, responds to the action in time step $t+1$, providing the agent with a new state s_{t+1} and reward r_{t+1}. Figure taken from [252].

an action value function Q in this process. Given a policy π, the state value function V^π contains for each state s the value $V^\pi(s)$, which corresponds to the total future reward that can be expected when the agent starts from state s and advances according to policy π. Alternatively, the action value function Q^π contains values $Q^\pi(s,a)$, which correspond to the total future reward to be expected if the agents starts in state s, takes action a, and follows π after that. Formal definitions of $V^\pi(s)$ and $Q^\pi(s,a)$ are given in equations 3.7 and 3.8 respectively, where s_t, a_t and r_t are the state, the action and the reward of the agent in time step t, and γ is the discount factor for future rewards. The calculation of V^π (or Q^π) is often used as an intermediate step in the learning process: first the value function V^π corresponding to π is calculated, then π is updated to π' so that the agent chooses states and actions with maximal values, and after that, the new value function $V^{\pi'}$ corresponding to π' is calculated. Methods to calculate policies and/or state and action value functions are described later in subsections 3.3.2 and 3.3.3. For details we refer to [252] and references therein.

$$V^\pi(s) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right\} \tag{3.7}$$

$$Q^\pi(s,a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\} \tag{3.8}$$

The problem of routing can easily be mapped onto the RL problem framework. The learning agent is the routing logic, which is distributed over the nodes of the network. The environment is the network, and the states are the nodes. The initial state is the source node. The reward is zero in each node,

apart from the destination node, where it is a number larger than zero. Shorter paths are preferred because they lead faster to a non-zero reward. The agent's possible actions are the next hops it can take in each of the nodes. The agent's policy, which indicates which action (next hop) to take in each of the states (nodes) in order to maximize reward (to find the destination fastest), is kept in a distributed way: it is spread over the routing tables maintained in the different nodes. The relationship between the task of routing and the RL framework was first pointed out in [39].

3.3.2 Elementary solution methods for reinforcement learning: dynamic programming and Monte Carlo sampling

A wide range of different solution methods have been developed for RL problems. The two most elementary ones are dynamic programming and Monte Carlo sampling. Here, we describe these approaches, and discuss how they relate to strategies used in routing. Later, in subsection 3.3.3, we describe another solution method, temporal-difference learning, and discuss how one of its instances, namely Q-learning, formed the basis for a different adaptive routing algorithm, called Q-routing.

A first solution method for RL problems is dynamic programming. This is a general technique for solving optimization problems proposed in [26], and has been in use for a long time before it was applied to RL. Dynamic programming has been discussed before in 3.1.1, as it was at the basis of the development of distance vector routing algorithms. The basic idea behind this method is to split an optimization problem into subproblems, and use the solutions to these subproblems to construct an optimal solution for the main problem. A straightforward way of using dynamic programming in RL is policy iteration. The agent starts with a random initial policy π_0, and uses dynamic programming to calculate the state value function V^{π_0} corresponding to it. Then π_0 is updated to π_1 by choosing in each state the actions that optimize the values of the states visited by the agent with respect to V^{π_0}. Then, dynamic programming is used again to calculate V^{π_1}. The formula for calculating each state value $V^{\pi}(s)$ using dynamic programming is given in equation 3.9, where $\mathcal{P}^a_{ss'}$ is the probability of reaching state s' after taking action a in state s, and $\mathcal{R}^a_{ss'}$ is the expected reward when reaching state s' after taking action a from state s. Compared to the formula of equation 3.7, the infinite sum of future rewards is replaced by the sum of the expected immediate reward and the estimated values of the neighbor states s'. This is the trademark approach of dynamic programming: the task of calculating the value of a state is solved based on the solution of a number of subtasks, namely the values of the neighboring states. This characteristic approach of calculating estimates based on other estimates is in the context of RL also called information bootstrapping.

$$V^{\pi}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')] \tag{3.9}$$

Dynamic programming formed the inspiration for distance vector routing algorithms such as RIP. Details about this have been given before in subsection 3.1.1. Here, we only want to point out how this approach to routing is affected by the use of the characteristic technique of information bootstrapping. This provides high efficiency, due to the reuse of solutions that were calculated for different subproblems (routing estimates calculated for neighboring nodes). However, it is not very robust or adaptive: if estimates are wrong in one node (e.g. due to changes in the environment), they are propagated over the network since other estimates are based on them, and it can take a long time until the error is fixed and all the routing information converges again to correct values. This limits the usability of a pure dynamic programming approach for adaptive routing in AHWMNs.

A second elementary solution method for RL is Monte Carlo sampling. Also this method has been around long before RL itself was a topic. In fact, the term Monte Carlo refers to any problem solving method in which solutions are learned from experiences obtained by repeated sampling. In the field of RL, Monte Carlo sampling can be used to calculate value functions in a policy iteration approach, like the one described for dynamic programming above. One difference is that in Monte Carlo learning usually the action values $Q^\pi(s,a)$ are used rather than the state values $V^\pi(s)$. To calculate the value $Q^\pi(s,a)$ of action a in state s, the agent tries it out: it runs a full episode of the task (i.e. until it reaches the goal state), starting from state s with action a and then following policy π after that. By executing repeated samples and taking the average of the total rewards experienced in each of these samples, a good estimate of the expected value of the total future reward can be obtained. In practice, a moving average can be used, applying the formula of equation 3.10 after each newly collected sample experience. In this formula, $R^\pi(s,a)$ is the total reward experienced when taking action a in state s and following π after that, and α is a constant factor used for the moving average. A special case of Monte Carlo sampling is the off-policy version, in which the agent collects sample experiences following a different policy than the one it is actually learning about. This allows more freedom (e.g. to do exploration), but is computationally more complicated, as one has to make an adjustment to the average experienced rewards in order to find the correct value for the other policy.

$$Q^\pi(s,a) \leftarrow Q^\pi(s,a) + \alpha[R^\pi(s,a) - Q^\pi(s,a)] \tag{3.10}$$

It is easy to see how the ACO approach to routing relies in its process for gathering routing information on the same ideas as the Monte Carlo approach to learning in RL. Ants learn action values (the value of taking a next hop in a certain node) by executing complete samples of solutions to the task (by sampling a path to the destination). The final pheromone values are the result of taking a sufficiently high number of samples. ACO routing algorithms in which different forwarding policies are used for ants and data forwarding are similar to off-policy Monte Carlo learning. The fact that the pheromone values do not always correspond to the average of the values experienced by the ants,

but are instead the result of more complicated evaluations of these values (e.g. in AntNet, described in subsection 3.2.3, the experienced delay is manipulated using the local traffic statistics), is reminiscent of actor-critic methods, which are more advanced RL learning methods [25]. The most important advantage of Monte Carlo sampling compared to information bootstrapping is that all estimates are based on direct experiences. This makes the estimates more reliable than when they are based on other estimates, giving increased robustness to the algorithm. This is particularly important in a changing and unreliable environment like AHWMNs. Adaptivity can in principle be a problem in Monte Carlo learning, namely when on-policy learning is applied and the policy has converged. In ACO routing, this problem is solved by using an off-policy approach in which ants follow a slightly different policy from data packets.

It is interesting to note that more traditional routing algorithms for AHWMNs, such as AODV, DSR and other reactive algorithms (see subsection 2.4.2), also rely on sampling. They construct a path based on a single sample. This is of course quite different from taking the average value obtained from repeated sampling such as in ACO routing. Nevertheless, it is a confirmation of the usefulness in AHWMNs of the increased robustness obtained through path sampling. This is especially clear in the development of AODV. This algorithm is presented as a distance vector algorithm as it was conceived as a successor to DSDV, a routing algorithm that relies purely on information bootstrapping. However, the only elements it still contains of distance vector routing is the shape of the routing tables. The information bootstrapping approach used in DSDV is deemed unworkable in the highly dynamic and unreliable AHWMN environment, and is replaced by a strategy that relies on sampling. However, it needs to be noted that the sampling in AODV and other reactive routing algorithms is not always done in a consequent way. E.g., AODV RREQ messages do not always go all the way until the destination, but return to the source as soon as they find a node that has routing information about the destination. At that point, the source node's routing information is again based on an estimate provided by another node.

3.3.3 Temporal-difference learning and Q-routing

Here we describe a more advanced class of solution methods for RL problems, namely temporal-difference learning. This approach combines elements of dynamic programming and Monte Carlo sampling, in order to get advantages from both methods. It is at this point important to point out that the AntHocNet algorithm for routing in AHWMNs presented in chapter 4 of this thesis also combines elements of both dynamic programming and Monte Carlo sampling. It does so in a completely different way than temporal-difference learning though, as will become clear later.

The simplest version of temporal-difference learning is the one step temporal-difference method, referred to as TD(0). The main idea behind this method is for the agent to take a sample of just one step, and then bootstrap information. Concretely, in TD(0), the agents calculates $Q(s, a)$ by taking action a in state s,

observing the immediately received reward r, and bootstrapping on the value of the new state it arrives in. An example update formula is given in equation 3.11, where α is a constant factor for calculating the moving average, and γ is a constant factor to discount rewards that are received in a future time step. The difference with the update formula for Monte Carlo learning (equation 3.10) is that only one step of experienced reward is used. The difference with the update formula for dynamic programming (equation 3.9) is that an experience is used for the one-step reward, rather than the expected value. Variations are possible in the choice of the action a' in the new state s' on the basis of which the bootstrapped value $Q(s', a')$ is chosen. E.g., the most popular TD(0) method, Q-learning, uses the action a' that gives the maximal value for $Q(s', a')$ [272]. This corresponds to an off-policy form of temporal-difference learning, and allows fast convergence to the optimal policy π^*. The one-step form of temporal difference learning found in TD(0) can be extended to an n-step algorithm, in which the agent takes a sample of n steps before bootstrapping the local state value. If n is equal to infinity or to the number of steps needed to reach the goal, this comes down to Monte Carlo sampling. Finally, when the results from several n-step samples are combined, with different values for n, we get the TD(λ) methods, which are also referred to as eligibility traces [271].

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) + \alpha[r + \gamma Q(s', a') - Q^\pi(s, a)] \tag{3.11}$$

Just like dynamic programming and Monte Carlo sampling, temporal-difference learning has been applied to routing. In particular, the Q-learning algorithm mentioned above has lead to the development of Q-routing [39]. In Q-routing, nodes keep routing tables that are similar to those in AntNet, with one entry per combination of next hop and destination. The routing table entry c_{ij}^d indicates the cost of going from i over neighbor j to destination d. Each time node i sends a data packet to a neighbor j for a destination d, j sends a control packet back to i. This packet contains the values t_{tot}, the total queueing and transmission delay experienced by the data packet to reach j from i, and c_j^d, the cost of the best path to destination d present in j's routing table. In node i, t_{tot} and c_j^d are summed to get an estimate of the cost from i to d over j, and the value c_{ij}^d in i's routing table is updated accordingly. When compared to Q-learning, t_{tot} corresponds to the immediate reward experienced in the one step sample, and c_j^d corresponds to the value of the next state. Like in dynamic programming approaches to routing, such as the basic distance vector routing algorithm, routing updates are based on the best paths reported by neighboring nodes. The main difference is that in Q-routing the cost of the single hop to the neighbor is taken from a sample, while in distance vector routing, it is an estimate of the expected value, obtained via local monitoring. In direct comparisons with the AntNet adaptive routing algorithm, Q-routing scored less well [70].

Another RL inspired approach to routing is SAMPLE [90]. This is an adaptive algorithm for MANETs. Similarly to Q-routing, it does an update of routing information with the transmission of each data packet. Also like Q-routing and dynamic programming, the best path available in neighboring nodes is used for

information bootstrapping in the update process. For the calculation of the cost of the one hop to the neighbor, SAMPLE relies on a model of the environment. Model-based RL [91] is not presented here, as it would lead us to far. However, the end result is an algorithm that is similar to both Q-routing and distance vector routing. Important for MANETs, SAMPLE uses an estimate of link qualities with similarities to the ETX measure (see subsection 2.4.3) to define path costs.

3.4 Conclusion

In this chapter we have described existing adaptive routing algorithms. The aim of the chapter was on the one hand to provide information about other algorithms in the field, and on the other hand to get the reader acquainted with the techniques and terminology that will be used further in this thesis.

In a first section, we have described adaptive routing algorithms in the internet. Two main approaches were discussed, namely distance vector routing and link state routing. For either of these approaches, we have investigated how they perform in terms of the major challenges for AHWMN routing: adaptivity, robustness, efficiency and scalability. We have also indicated to what extent both approaches to routing have been adapted to work in AHWMNs.

Then, in a second section, we have described ACO and ACO routing. ACO is a metaheuristic for the solution of optimization problems that was inspired by the shortest path finding behavior of ant colonies in nature. ACO routing is an approach to routing that is based on the ACO framework. It forms an important source of inspiration for the work presented in this thesis. Here, we have described the major characteristics of ACO routing in detail, and have given an extensive overview of existing ACO routing algorithms, both for wired networks and for AHWMNs.

In a last section, we have treated adaptive routing from a machine learning point of view. To that end, we have described RL, an important class of learning problems. We have shown that the problem of routing in telecommunication networks fits well into this problem framework. Then, we have investigated how the strategies for the gathering of routing information used by distance vector and ACO routing algorithms correspond to two important learning methods for RL problems, namely dynamic programming and Monte Carlo learning. Finally, we have described temporal difference learning, a more advanced solution method for RL problems that combines elements of both dynamic programming and Monte Carlo learning, and we have explained the Q-routing algorithm that was derived from it. The AntHocNet routing algorithm for AHWMNs described in the next chapter also combines elements from both of these learning methods, but in quite a different way.

Chapter 4

AntHocNet: an adaptive routing algorithm for ad hoc wireless multi-hop networks

In this chapter, we describe AntHocNet, an adaptive routing algorithm for AHWMNs. AntHocNet is a hybrid routing algorithm, in the sense that it contains elements from both reactive and proactive routing. Specifically, it combines a reactive route setup process with a proactive route maintenance and improvement process. AntHocNet was in the first place inspired by the ACO approach to routing. This is evident in the way that it gathers, stores and uses routing information. Consequently, the terminology used in this chapter is mostly related to the ACO routing literature. Nevertheless, AntHocNet also contains elements from distance vector routing. In particular, the information gathering process used in its proactive route maintenance and improvement process combines the route sampling strategy from ACO routing with an information bootstrapping process that is similar to the one used in distance vector routing algorithms. The way both approaches are combined is novel and allows the algorithm to get the best of both worlds. AntHocNet was in the first place developed for MANETs and WMNs. Descriptions of the AntHocNet algorithm have been published in [73–75, 94, 95, 98].

The rest of this chapter is organized as follows. First, we give a general overview of the AntHocNet routing algorithm. This includes a high level description of the algorithm in words, and a schematic representation. Then, we give a detailed description of each of the algorithm's components. Finally, we provide further discussions on the presented material, such as an investigation of the relations between AntHocNet and other AHWMN routing algorithms, a discussion of how AntHocNet relates to the RL solution methods presented

in chapter 3, and a description of elements that were present in older versions of AntHocNet. Evaluations of AntHocNet and experimental comparisons with other routing algorithms will be provided in the following chapters.

4.1 General overview of the AntHocNet routing algorithm

In this section, we give an overview of the algorithm. We start with a general description in words. Then, we also provide a schematic representation in the form of a finite state machine.

4.1.1 Algorithm description

AntHocNet is a hybrid algorithm, containing both reactive and proactive elements. The algorithm is reactive in the sense that it only gathers routing information about destinations that are involved in communication sessions. It is proactive in the sense that it tries to maintain and improve information about existing paths while the communication session is going on (unlike purely reactive algorithms, which do not search for routing information until the currently known routes are no longer valid). Routing information is stored in pheromone tables that are similar to the ones used in other ACO routing algorithms. Forwarding of control and data packets is done in a stochastic way, using these tables. Link failures are dealt with using specific reactive mechanisms, such as local route repair and the use of warning messages. Below, we describe the general working of the AntHocNet routing algorithm. Details will follow later in section 4.2.

In AntHocNet, routing information is organized in pheromone tables, similar to the ones used in other ACO routing algorithms such as the earlier described AntNet (see subsection 3.2.3). Each node i maintains one pheromone table $\mathcal{T}_i$, which is a two-dimensional matrix. An entry $\mathcal{T}_{ij}^d$ of this pheromone table contains information about the route from node i to destination d over neighbor j. This information includes the pheromone value τ_{ij}^d, which is a value indicating the relative goodness of going over node j when traveling from node i to destination d, as well as statistics information about the path, and possibly virtual pheromone (see later). Apart from a pheromone table, each node also maintains a neighbor table, in which it keeps track of which nodes it has a wireless link to. Details about the data structures maintained under AntHocNet are described in subsection 4.2.1.

At the start of a communication session, the source node of the session controls its pheromone table, to see whether it has any routing information available for the requested destination. If it does not, it starts a reactive route setup process, in which it sends an ant packet out over the network to find a route to the destination. Such an ant packet is called a reactive forward ant. Each intermediate node receiving a copy of the reactive forward ant forwards it. This is done via unicasting in case the node has routing information about the ant's

destination in its pheromone table, and via broadcasting otherwise. Reactive forward ants store the full array of nodes that they have visited on their way to the destination. The first copy of the reactive forward ant to reach the destination is converted into a reactive backward ant, while subsequent copies are destroyed. The reactive backward ant retraces the exact path that was followed by the forward ant back to the source. On its way, it collects quality information about each of the links of the path. At each intermediate node and at the source, it updates the routing tables based on this quality information. This way, a first route between source and destination is established at completion of the reactive route setup process. The full process is repeated later if the source node falls without valid routing information for the destination of the session while data still need to be sent. Details about the reactive route setup process are provided in subsection 4.2.2.

Once the first route is constructed via the reactive route setup process, the algorithm starts the execution of the proactive route maintenance process, in which it tries to update, extend and improve the available routing information. This process runs for as long as the communication session is going on. It consists of two different subprocesses: pheromone diffusion and proactive ant sampling. The aim of the pheromone diffusion subprocess is to spread out pheromone information that was placed by the ants. Nodes periodically broadcast messages containing the best pheromone information they have available. Using information bootstrapping, neighboring nodes can then derive new pheromone for themselves and further forward it in their own periodic broadcasts. Details about this process will be given later, in subsection 4.2.3. Here, it is sufficient to know that the pheromone diffusion process is similar to the dynamic programming approach used in distance vector routing. As was pointed out earlier in subsections 3.1.1 and 3.3.2, such approaches to gathering routing information are very efficient, but can be slow to adapt to dynamic situations, possibly temporarily providing erroneous information. Therefore, the pheromone diffusion process can be considered as a cheap but potentially unreliable way of spreading pheromone information. Because of this potential unreliability, the pheromone that is obtained via pheromone diffusion is kept separate from the normal pheromone placed by the ants, and is called virtual pheromone; the pheromone placed by the ants will in what follows be called regular pheromone. The virtual pheromone is used to support the second subprocess of proactive route maintenance, which is proactive ant sampling. In this subprocess, all nodes that are the source of a communication session periodically send out proactive forward ants towards the destination of the session. These ants construct a path in a stochastic way, choosing a new next hop probabilistically at each intermediate node. Different from reactive forward ants, they are never broadcast. When calculating the probability of taking a next hop, proactive forward ants consider both regular and virtual pheromone. This way, they can leave the routes that were followed by previous ants, and follow the (potentially unreliable) routes that have emerged from pheromone diffusion. Once a proactive forward ant reaches the destination, it is converted into a proactive backward ant that travels back to the source and leaves pheromone along the

way (regular, not virtual pheromone), just like reactive backward ants. This way, proactive ants can follow virtual pheromone and then, once they have experienced that it leads to the destination, convert it into regular pheromone. One could say that pheromone diffusion suggests new paths and that proactive ants check them out. The ant based full path sampling provides the reliability that is lacking in the efficient information bootstrapping process. Details about the proactive route maintenance process are given in subsection 4.2.3.

Data packet forwarding in AntHocNet is done similarly to other ACO routing algorithms: routing decisions are taken hop-by-hop, based on the locally available pheromone. Only regular pheromone is considered, as virtual pheromone is not considered reliable enough. Each forwarding decision is taken using a stochastic formula that gives preference to next hops that are associated with higher pheromone values. The formula is different from that used by the forward ants, so that data packets can follow a less exploratory strategy. Via parameter tuning, it is possible to vary between spreading the data packets over all possible available paths and deterministically sending them over the best path. While the former can in principle provide higher throughput through the use of multiple paths (see subsection 2.4.3), the latter allows greedy exploitation of the learned information. Later, in chapter 5, we compare both strategies empirically. Details about data packet forwarding in AntHocNet are provided in subsection 4.2.4.

Link failures can be detected in AntHocNet via failed transmissions of data or control packets, or through the use of hello messages. Hello messages are short messages that are periodically sent out by all nodes in the network. The reception of a hello message is indicative of the presence of a wireless link, while the failure to receive such messages point to the absence of a link. In practice in AntHocNet, the function of hello messages is fulfilled by the same periodic messages that are used for pheromone diffusion. When a node detects a link failure, it controls its pheromone table, to see which routes become invalid due to the failure, and whether alternative routes are available for the affected destinations. Then, it broadcasts a link failure notification message to warn neighboring nodes about all relevant changes in its pheromone table. In case the link failure was associated with a failed data packet transmission, the node can also start a local route repair to restore the route to the destination of this data packet. To this end, it sends out a repair forward ant. Repair forward ants are similar to reactive forward ants, in the sense that they follow available pheromone information where possible, and are broadcast otherwise, but they have a limited maximum number of broadcasts, so that they cannot travel far from the old failed route. Upon arrival at the destination, the repair forward ant is converted into a repair backward ant that travels back to the node that started the repair process and sets up the pheromone for the repaired route. A last tool in dealing with link failures is the use of unicast warning messages. These are needed when data packets for a lost destination still arrive at the node after a link failure notification has already been broadcast. This can be due to bad reception of the broadcast notification message. In this case, the node unicasts a warning to the node it received the data from, in order to inform

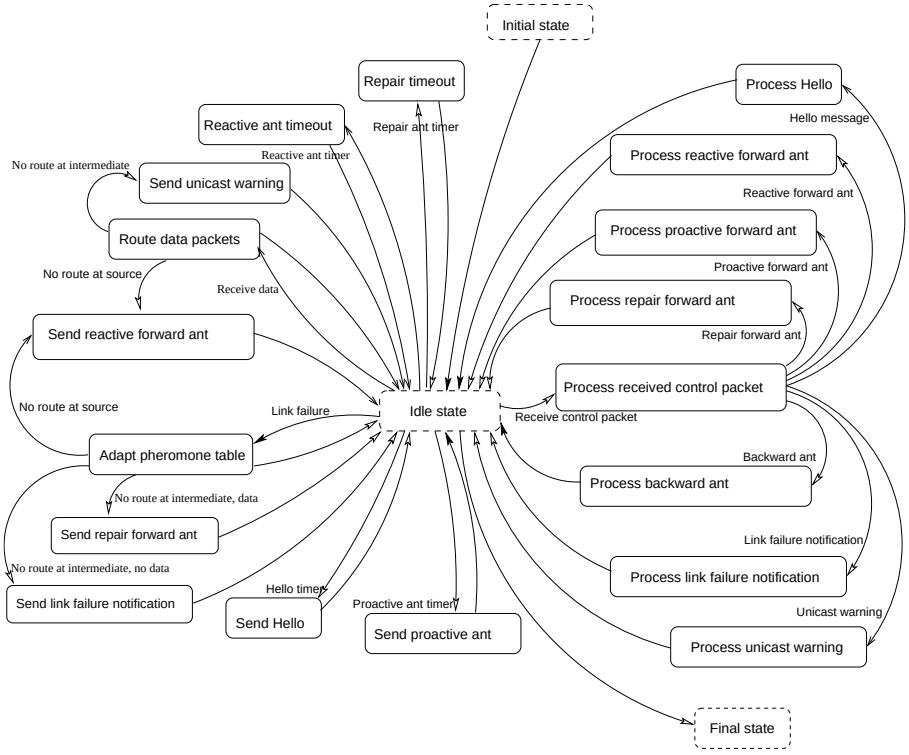


Figure 4.1: A finite state machine representation of the AntHocNet routing algorithm.

it that it can no longer forward data for this destination. Details about how AntHocNet deals with link failures are described in subsection 4.2.5.

4.1.2 Schematic representation

Figure 4.1 gives a schematic representation of the AntHocNet routing algorithm. It contains a finite state machine showing the most important components of the algorithm. Below we explain the structure of the finite state machine. For details about each of the components, we refer to section 4.2.

The algorithm is started up in its initial state, in which internal variables are initialized. Then, it moves to its idle state, where it waits for events to happen. The events that can take place are the arrival of a data packet, the arrival of a control packet, the detection of a link failure, and a number of timer events.

In case of a data reception event, the algorithm tries to route the data. Details about data forwarding are given in subsection 4.2.4. This might be impossible, due to the lack of routing information about the destination. If the current node is the source node of the data packet, the unavailability of routing

information can be because the data packet is the first of a new communication session, or because it belongs to a session for which all routing information has become invalid. In both cases, the node starts a reactive route setup process. Details about this process are given in subsection 4.2.2. If the current node is not the source of the data packet, it concludes that the upstream node of the data has wrong routing information, and sends it a unicast warning message. This is further explained in subsection 4.2.5, which talks about link failures.

In case of a control packet reception event, the algorithm checks which type of control packet it is dealing with. In case it is a hello message, the node needs to take note in its neighbor table that it has a wireless link with the packet's sender, and it needs to extract the routing information inside to update its own virtual pheromone information. This is part of the proactive route maintenance process described in subsection 4.2.3. In case it is a reactive, proactive or repair forward ant, the node needs to execute the correct forwarding action, or, if it is the final destination of the ant, create a backward ant and send it back towards the source. In case the control packet is a backward ant (reactive, proactive and repair backward ants have essentially the same behavior, and are therefore collapsed here), the node needs to adapt its pheromone table, and forward the ant if it is not its final destination. Details about the treatment of reactive, proactive and repair forward and backward ants are given respectively in subsections 4.2.2, 4.2.3 and 4.2.5. In case the control packet is a link failure notification, the node needs to update its pheromone table, and possibly forward the notification. Finally, if it is a unicast warning, it needs to update its pheromone table, removing the erroneous route. Details about link failure notifications and unicast warnings are given in subsection 4.2.5.

In case of a link failure event, the node first of all adapts the information in its pheromone table to reflect the changed situation. Then, if destinations have become unreachable due to the link failure, it needs to take action. If the current node is the source of a session to one of the lost destinations, it starts a reactive route setup process. If, on the other hand, it is an intermediate node on the lost route, it controls whether the link failure event involved the unsuccessful transmission of a data packet. If this is the case, it starts a local route repair process. Otherwise, it broadcasts a link failure notification message. Details on how link failures are dealt with are given in subsection 4.2.5.

The different timers are events that are scheduled by the node itself, in order to plan delayed actions. Hello timer events are scheduled at regular intervals, from the moment the node is switched on and for as long as it is up and running. Reception of a hello timer event provokes the node to send a hello message, in which it includes its best pheromone information. This is part of the proactive route maintenance process described in subsection 4.2.3. Proactive ant timer events are also scheduled at regular intervals, but only from the moment a session is started, and until the end of it. Reception of a proactive ant timer event leads the node to send out a proactive forward ant. Details about this are given in subsection 4.2.3. Repair timer events are scheduled after a repair forward ant has been sent out. At reception of a repair timer event, the node checks whether a repair backward ant was received, and in case not, it broadcasts a

link failure notification. This is part of the process of dealing with link failures, described in 4.2.5. Finally, reactive ant timer events are similarly sent out after the transmission of a reactive forward ant. At reception of a reactive ant timer event, the node controls whether it has already received a reactive backward ant. In case not, it can send out a new reactive forward ant (in case the maximum number of retransmissions has not yet been reached), or conclude that the destination is currently unreachable, and drop queued data packets for it. This is described in detail in subsection 4.2.2, which explains the reactive route setup phase.

4.2 Detailed descriptions

In this section, we give a detailed description of the different components of the AntHocNet routing algorithm. We follow the same structure as in subsection 4.1.1. First we describe the data structures that are maintained in each node. Then, we give details about the reactive route setup process. Subsequently, we discuss the proactive route maintenance process. Next, we talk about data packet forwarding. After that, we discuss the algorithm's behavior with respect to link failures. Finally, we talk about the routing metrics used in AntHocNet.

4.2.1 Data structures in AntHocNet

Here, we describe the different data structures that are maintained by each of the network nodes under AntHocNet. In particular, we talk about pheromone tables and neighbor tables.

Pheromone tables

Under AntHocNet, each node i maintains a pheromone table $\mathcal{T}_i$, which is a two-dimensional matrix. An entry $\mathcal{T}_{ij}^d$ of this matrix contains information about the route from node i to destination d over neighbor j. This includes a regular pheromone value τ_{ij}^d, a virtual pheromone value ω_{ij}^d, and an average number of hops h_{ij}^d. The regular pheromone value τ_{ij}^d is an estimate of the goodness of the route from i to d over j. Goodness is expressed as the inverse of a cost. Exact values depend on the metric that is used to evaluate the cost of routes. More about the use of different metrics will follow in subsection 4.2.6. Regular pheromone is updated by backward ants. These can be reactive, proactive or repair backward ants. Details about the updating process for regular pheromone are given in subsection 4.2.2. The virtual pheromone value ω_{ij}^d forms an alternative estimate of the goodness of the route from i to d over j. Differently from τ_{ij}^d, it is obtained through information bootstrapping using goodness values reported by neighbor nodes during the proactive route maintenance process. The updating of virtual pheromone is discussed in subsection 4.2.3. The average number of hops h_{ij}^d is, like the regular pheromone, updated by backward ants.

This updating process is described in subsection 4.2.2. h_{ij}^d is used when deciding how long to wait for repair backward ants (see subsection 4.2.5).

Nodes do not necessarily always have values available for τ_{ij}^d, ω_{ij}^d, and h_{ij}^d for each possible combination of destination and next hop. This is in the first place because nodes do not maintain routing information about all possible destinations in the network (they only gather routing information for destinations which communication sessions are going on with), and because for a specific destination, nodes do not necessarily have a route available over each of their possible neighbors (for instance, during a reactive route setup phase only one route is set up, so that the source node has exactly one outgoing next hop for the involved destination). Also, since regular and virtual pheromone are obtained through different processes, it is possible that a node has a value for τ_{ij}^d, but not for ω_{ij}^d, or vice versa. On the other hand, since τ_{ij}^d and h_{ij}^d are both obtained from backward ants, nodes that have a value for one of the two will also have a value for the other.

Neighbor tables

Apart from the pheromone table, each node also maintains a neighbor table. The neighbor table $\mathcal{N}_i$ kept by node i is a one-dimensional vector with one entry for each of i's neighbors. The entry $\mathcal{N}_{ij}$ corresponding to i's neighbor j contains a time value th_{ij} indicating when i last heard from j. Node i uses this time value to derive whether there is a wireless link with node j, and to detect link failures. More on the detection and handling of link failures will follow later in subsection 4.2.5.

4.2.2 Reactive route setup

The reactive route setup process is triggered whenever a node receives a data packet that was locally generated (i.e., the current node is the packet's source) for a destination for which no routing information is available. This lack of routing information can happen either because the data packet in question is the first of a new communication session, and no routing information for its destination is available from a different or previous session, or because the data packet belongs to an ongoing session for which all routes have become invalid (e.g., due to node movements). The reactive route setup process involves the sending of a reactive forward ant from source to destination, and a reactive backward ant from destination to source. Below, we discuss each of these separately.

Reactive forward ants

At the start of the reactive route setup process, the source node s creates a reactive forward ant. This is a control packet that has as a goal to find a path from s to an assigned destination d. At the start, the ant contains just the addresses of s and d. Later, as it proceeds through the network, it collects a list $\mathcal{P} = [1, 2, \ldots, d-1]$ of all the nodes that it has visited on its way from s to d.

After its creation at s, the reactive forward ant is broadcast, so that all of s's neighbors receive a copy of it. At each subsequent node, the ant is either unicast or broadcast, depending on whether the current node has routing information for d. If routing information is available, the node chooses a next hop for the ant probabilistically, based on the different pheromone values associated with next hops for d. Concretely, a node i chooses node n as next hop for the ant with probability P_{in}^d, as calculated by equation 4.1. In this equation, N_i^d is the set of neighbors of i over which a path to d is known, and β_1 is a parameter value which can control the exploratory behavior of the ants. In our experiments, we keep β_1 relatively high, on 20. This is because we want to obtain the initial route as fast as possible, and limit the time we spend on exploration at this stage.

$$P_{in}^d = \frac{(\tau_{in}^d)^{\beta_1}}{\sum_{j \in N_i^d} (\tau_{ij}^d)^{\beta_1}}, \quad \beta_1 \geq 1, \tag{4.1}$$

In case the intermediate node i does not have routing information for d, it broadcasts the reactive forward ant. Due to this broadcasting (and also the initial broadcasting at s), a reactive forward ant can proliferate quickly over the network, with different copies of the ant following different paths to the destination. In order to limit the amount of overhead that is created this way, nodes only forward the first copy of the ant that they receive. Subsequent copies are simply discarded. In previous versions of AntHocNet, nodes were to some extent allowed to forward multiple copies of the same ant, in order to improve the creation of multiple paths (see also subsection 4.3.4). However, this lead to a lot of overhead.

At the destination, the reactive forward ant is converted into a reactive backward ant, which follows the list of nodes $\mathcal{P}$ visited by the forward ant back to s. If more than one copy of the forward ant is received, only the first is accepted and converted into a backward ant, while subsequent copies are discarded. This way, only one route is set up during the reactive route setup process. The reason is again to reduce overhead created during this procedure. For the creation of multiple routes, AntHocNet relies on the proactive route maintenance process, which extends the initially created route into a full mesh of routes during the course of the communication session (see subsection 4.2.3).

Reactive backward ants

The reactive backward ant created by the destination node in response to a reactive forward ant contains the addresses of the forward ant's source node s and destination node d, as well as the full list of nodes $\mathcal{P}$ that the forward ant has visited. The reactive backward ant is unicast from d and between the nodes of $\mathcal{P}$ back to s.

The aim of the reactive backward ant is to update routing information in each of the nodes of $\mathcal{P}$ and in s. At each node i that it visits, it updates the number of hops h_{in}^d and the regular pheromone value τ_{in}^d in the pheromone table

entry $\mathcal{T}_{in}^{d}$, where n is the node that it visited before i on its way from d. The updating of h_{in}^{d} is done using a moving average, as shown in equation 4.2. In this equation, h is the number of hops that the backward ant has traveled between d and i, and α is a parameter regulating how quickly the formula adapts to new information. In our experiments, α is always kept on 0.7.

$$h_{in}^{d} \leftarrow \alpha h_{in}^{d} + (1-\alpha)h, \quad \alpha \in [0,1] \tag{4.2}$$

Updating of the regular pheromone τ_{in}^{d} is done based on the cost of the route from i to d. This cost can be calculated using different metrics, such as the number of hops, the end-to-end delay, etc.. Later on, in subsection 4.2.6, we comment on different metrics used in AntHocNet. Here, we talk in terms of a generic cost c, where c_i^d is the cost of the route from i to d, and c_i^j is the cost of the link from i to its neighbor j (it is the cost of a one-hop route). Under AntHocNet, each node maintains a local estimate of the cost c_i^j to go to each of its neighbors j. Details about how these local estimates are calculated depend on the metric and are discussed in subsection 4.2.6. The reactive backward ant reads at each node i the local estimate c_i^n of the cost to go from i to the next hop n that the ant is coming from. It adds this cost to the total cost c_n^d of the route from n to d (which it has been calculating on its way back from d), which is stored inside the ant. The new cost c_i^d is used to update the pheromone value τ_{in}^{d} in node i, using the moving average formula of equation 4.3. In this equation, γ is a parameter regulating the speed of adaptation of the pheromone to new cost values. In our experiments, γ was kept on 0.7. The cost value c_i^d is inverted to calculate the pheromone value τ_{ij}^{d}, as pheromone indicates the goodness of a route, rather than its cost.

$$\tau_{ij}^{d} \leftarrow \gamma \tau_{ij}^{d} + (1-\gamma)(c_i^d)^{-1}, \quad \gamma \in [0,1] \tag{4.3}$$

It is interesting to note that in terms of gathering route cost information, there is an important difference here with the AntNet algorithm described in subsection 3.2.3 and other ACO routing algorithms. Rather than relying on the cost values experienced by the forward ants, AntHocNet uses the estimates c_i^j calculated locally by the nodes. This is in order to improve reliability of the measured values. Depending on the cost metric used, the high variability of the wireless medium can cause large differences between values measured by subsequent samples. For example, the delay incurred on a link can vary strongly in case of congestion if IEEE 802.11 is used as a MAC layer protocol (see subsection 2.3.3). Using local estimates that are based on more than one sample can take away some of this variability.

4.2.3 Proactive route maintenance

The proactive route maintenance process serves to update and extend available routing information. In particular, it allows to build a mesh of multiple routes around the initial route created during the reactive route setup process. The proactive route maintenance process consists of two subprocesses: pheromone

diffusion and proactive ant sampling. Pheromone diffusion is aimed at spreading available pheromone information over the network through the use of periodic update messages and information bootstrapping. Proactive ant sampling is aimed at controlling and updating pheromone information through path sampling using proactive forward ants. While proactive ant sampling is started by the source node of a communication session at the start of the session and continues for as long as the session is going on, pheromone diffusion is executed by all nodes throughout their whole lifetime, and is not particulary bound to the occurrence of a single session. Below, we describe each of the two subprocesses of proactive route maintenance separately.

Pheromone diffusion

The reactive route setup process described in subsection 4.2.2 leads to the availability of a single route from the source of a communication session to its destination, indicated by regular pheromone values in the pheromone tables of the nodes. Moreover, each neighbor node of the destination also has a one-hop route to the destination. This is independent of the running session, and is simply due to the fact that neighboring nodes are aware of each other's presence, as is explained further in subsection 4.2.5. The aim of the pheromone diffusion process is to spread all this pheromone information out, so that a field of pheromone pointing towards the destination becomes available in the network. This field of pheromone is indicated in the virtual pheromone values in the pheromone tables of the nodes. The fact that pheromone is spread out is similar to the normal diffusion of real pheromone in nature [185], which allows ants further away to sense it. In the example of figure 4.2, a communication session is going on between node 1 and node 8. Regular pheromone is indicated by solid arrows. It consists of a single route from 1 to 8 over the nodes 3, 6 and 7 that is the result of reactive route setup, and a one-hop route from 5 to 8 that is there independently of the running session, because 5 is aware of the presence of its neighbor 8. The field of virtual pheromone that is the result of pheromone diffusion is indicated with dashed arrows.

A crucial role in the pheromone diffusion process is played by hello messages. These are short messages broadcast every t_{hello} seconds asynchronously by all the nodes of the network throughout their whole lifetime. In AntHocNet, t_{hello} is set to 1 second. Hello messages are used in many existing protocols, such as ABR [257] and OLSR [61] (see subsection 2.4.2), to allow nodes to find out which are their immediate neighbors, and to detect link failures. While also in AntHocNet hello messages are used for this purpose (as is explained in subsection 4.2.5), they also serve a different goal, namely to carry information in the pheromone diffusion process. They serve as the periodic update messages that are needed in the information bootstrapping process of pheromone diffusion. The idea to convey extra routing information inside hello messages has been used in some other routing algorithms, such as the earlier mentioned OLSR.

Nodes include in each hello message that they send out routing information they have available. In particular, a node i constructing a hello message consults

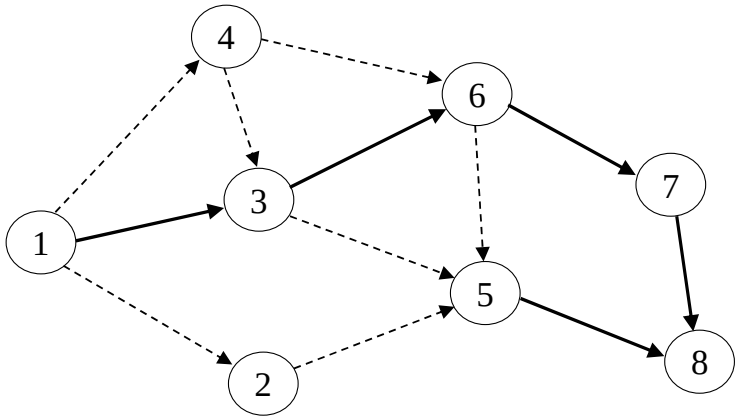


Figure 4.2: An example of available pheromone in an AHWMN. Node 1 is running a communication session with node 8 as destination. Regular pheromone is indicated by solid arrows. The route over the nodes 3, 6 and 7 is the result of a reactive route setup. The one-hop route from node 5 to node 8 is there independent of the running session: node 5 is aware that node 8 is its neighbor and therefore knows it has a one hop path to node 8. Virtual pheromone is indicated by dashed arrows. It forms a field pointing towards the destination node 8. Virtual pheromone is the result of the pheromone diffusion process.

its pheromone table, and picks a maximum number k of destinations it has routing information for. k is normally kept on 10, but in chapter 5 we also present results varying this parameter. If more than k destinations are available, k of them are picked out randomly. For each one of these destinations d, the hello message contains the address of d, the best pheromone value that i has available for d, υ_i^d, and a bit flag. This best pheromone value υ_i^d is taken over all possible values for regular pheromone τ_{ij}^d and virtual pheromone ω_{ij}^d associated with d in i's pheromone table $\mathcal{T}_i$. The bit flag is used to indicate whether the reported value was originally regular or virtual pheromone. In the example of figure 4.2, node 3 has the choice of reporting the regular pheromone value about the route over node 6 to node 8, or reporting the virtual pheromone value about the route over node 5 to node 8. In case the route cost metric in use is hop count, it will prefer to send out the virtual pheromone, as it points to a better route.

A neighboring node j receiving the hello message from i goes through the list of reported destinations. For each listed destination d, it derives from the hello message an estimate of the goodness of going from j to d over i, by applying information bootstrapping: it combines the reported pheromone value υ_i^d, which indicates the goodness of the best route from i to d, with the locally maintained estimate of the cost c_j^i of hopping from j to i. The exact formula is given in equation 4.4. The inversions are needed first to convert the goodness value υ_i^d

into a cost value so that it can be added to the cost value c_j^i, and then to convert the total sum again into a goodness value. The result of the calculation is what we call the bootstrapped pheromone value κ_{ji}^d. In the example of figure 4.2, node 3 receives a hello message from node 5 reporting the one hop route from 5 to 8. Node 3 extracts this reported pheromone and uses it to derive bootstrapped pheromone for the route over node 5 to node 8.

$$\kappa_{ji}^d = \left((\upsilon_i^d)^{-1} + c_j^i\right)^{-1} \tag{4.4}$$

With κ_{ji}^d, node j has obtained a new estimate for the goodness of the path to d over i in a relatively cheap way. Thanks to the use of information bootstrapping, all that was needed in terms of communication overhead was the sending of the value υ_i^d from i to j. Moreover, since υ_i^d was piggybacked on top of a hello message, which i needed to send out anyway in order to support link failure detection, the overhead is limited to a few extra bytes in transmission. In AHWMNs, this is an important detail, since a major part of the cost of data transmission is formed by channel access control activities (see subsection 2.3.3), which only need to be executed once for each packet, making the transmission of one large packet favorable compared to the transmission of several small packets. On the downside, the cheap procedure to obtain κ_{ji}^d comes at a price, in the form of reliability: since κ_{ji}^d is derived from the estimate υ_i^d reported by i, it is only correct as long as υ_i^d is correct. This can be problematic in a highly dynamic environment like AHWMNs, where routing information can get out of date quickly, and especially if the value υ_i^d reported by i was in itself the product of pheromone diffusion (i.e., if the value reported by i was originally virtual pheromone). We have provided discussions on the reliability of information bootstrapping and dynamic programming approaches in subsections 3.1.1 and 3.3.2. Since we do not adopt any additional mechanisms to ensure the reliability of the bootstrapped pheromone (in order to keep the system simple), and since the bootstrapping process is relatively slow using the periodic hello messages (in order to keep it efficient), we have to be aware that the bootstrapped pheromone value κ_{ji}^d is potentially unreliable. This influences the way node j can use κ_{ji}^d to update its pheromone table.

As described earlier in subsection 4.2.1, node j maintains in its pheromone table entry $\mathcal{T}_{ji}^d$ two distinct pheromone values for the route over its neighbor i to destination d: the regular pheromone value τ_{ji}^d and the virtual pheromone ω_{ji}^d. Of these, only the virtual pheromone value ω_{ji}^d is normally updated with the new bootstrapped pheromone value κ_{ji}^d. This way, the pheromone obtained via the pheromone diffusion process is kept separate from the regular pheromone, which is the product of ant based route sampling and is therefore considered more reliable. The updating is done by replacing ω_{ji}^d by κ_{ji}^d. In the example of figure 4.2, node 3 would use the earlier derived bootstrapped pheromone about the route over node 5 to node 8 to update its virtual pheromone. The approach of keeping virtual and regular pheromone separate means that bootstrapped pheromone is not used directly for the forwarding of data packets,

since data packets only consider regular pheromone when choosing a next hop (see subsection 4.2.4). Virtual pheromone is used when forwarding proactive forward ants towards their destination (more on this follows below, when we talk about proactive ant sampling). When reaching the destination, proactive forward ants are converted into proactive backward ants, which do deposit regular pheromone, which in turn is used for routing data packets. So, in this way, bootstrapped pheromone influences data forwarding indirectly. One could say that the potentially unreliable bootstrapped pheromone provides hints about possible routes, which are then explored and verified by the proactive forward ants.

There is one situation that forms an exception to this normal mode of operation, in which we do allow the bootstrapped pheromone value κ_{ji}^d to be used for updating the regular pheromone value τ_{ji}^d and influencing data forwarding directly. This is the case when the following two conditions are fulfilled: a) j already has a non-zero value for the regular pheromone τ_{ji}^d, and b) the bootstrapped pheromone κ_{ji}^d was derived from a reported pheromone value υ_i^d that was based on regular pheromone in i, rather than virtual pheromone (remember that i indicates this in a bit flag in its hello message). In the example of the network of figure 4.2, the described situation arises for instance when node 3 receives a hello message from node 6 reporting the regular pheromone value of the path from node 6 to node 8 going over node 7: node 3 already has non-zero regular pheromone for the route over node 6 to node 8, and the hello message received from node 6 reports regular pheromone. Under these conditions, we know that there is a reliable route from j to d over i, since the presence of regular pheromone indicates that this route has been sampled by ants in the past. Also, we know that κ_{ji}^d reflects information about this reliable route that is available in the next hop i, since it was based on a value υ_i^d that reflects regular pheromone about d available in i. This means that the bootstrapped pheromone is in fact a one step update of the routing information about this specific route. So, under these strict conditions, we consider it reliable enough to update regular pheromone: we replace τ_{ji}^d directly by κ_{ji}^d. This way, pheromone on current paths is kept up-to-date.

Proactive ant sampling

The proactive ant sampling process is started by the source node of a session at the moment the first data packet of a new session is received, and continues for as long as the session is going on. The aim of the process is to use ant based sampling to gather routing information for ongoing sessions. To this end, proactive forward ants are generated. These ants can follow regular pheromone, which is routing information placed by previous ants, or virtual pheromone, which is the result of the pheromone diffusion process described above. While the former leads the ants to update goodness estimates of existing routes, the latter allows them to find new routes based on the hints provided by the pheromone diffusion process. This way, the single route that was initially constructed in the reactive route setup process is extended to a full mesh of multiple paths. In the

example of figure 4.2, a proactive forward ant arriving in node 3 can follow the regular pheromone over node 6 to node 8, or the virtual pheromone indicating the shorter route over node 5 to node 8.

Each node which is the source of a communication session periodically (normally, we use a period of t_{hello} seconds, but we have also done tests with other values; see subsection 5.3.3) schedules the transmission of a proactive forward ant towards the session's destination. In order to improve efficiency, the actual sending of a proactive forward ant is conditional to the availability of good new virtual pheromone: only if the best virtual pheromone is significantly better (in our experiments: at least 10% better) than the best regular pheromone, a proactive forward ant is sent out. The aim of the proactive forward ant is to find a route towards the destination, and to store the list of nodes $\mathcal{P}$ that it visits on the way. The proactive forward ant takes a new routing decision at each intermediate node i, using the formula of equation 4.5 to calculate the probability of choosing each possible next hop n. In this formula, the function $max(a, b)$ takes the maximum of the two values a and b, and β_2 is a parameter that defines the exploratory character of the ants. Like for reactive forward ants, β_2 is normally kept on 20, but in chapter 5 we also compare results using different values for β_2. As can be seen from the equation, unlike reactive forward ants, proactive forward ants rely both on regular and virtual pheromone for their routing decisions: they use the maximum between regular and virtual pheromone to calculate the probability of each next hop. Also different from reactive forward ants is that proactive forward ants are never broadcast: when they arrive at a node that does not have any routing information for their destination, they are discarded.

$$P_{in}^d = \frac{[max(\tau_{in}^d, \omega_{in}^d)]^{\beta_2}}{\sum_{j \in N_i^d} [max(\tau_{ij}^d, \omega_{ij}^d)]^{\beta_2}}, \quad \beta_2 \geq 1, \tag{4.5}$$

When a proactive forward ant arrives at its destination, it is converted into a proactive backward ant, which is sent back to the source. Proactive backward ants have the same behavior as reactive backward ants: they follow the exact path $\mathcal{P}$ recorded by their corresponding forward ant back to the source, and update regular pheromone entries at intermediate nodes and at the source. For details about this behavior, we refer to the description of reactive backward ants in subsection 4.2.2.

An important aspect to note here is that while the proactive forward ants can follow both regular and virtual pheromone, proactive backward ants always deposit regular pheromone. This way, the proactive ant sampling process can investigate promising virtual pheromone, and if the investigation is successful turn it into a regular route that can be used for data. In the example of figure 4.2, a proactive forward ant following the virtual pheromone from node 3 over node 5 to node 8 is at its arrival in 8 converted into a backward ant, which deposits regular pheromone on the link from node 3 to node 5. The process of proactive ant sampling increases in this way the number of routes available for data routing, which can grow to a full mesh, and allows the algorithm to

exploit new routing opportunities in the ever changing AHWMN topology. As stated earlier, the proactive ants provide through their full path sampling the necessary control to verify the reliability of new routes obtained through the information bootstrapping process of pheromone diffusion.

4.2.4 Data packet forwarding

Data packets are forwarded from their source to their destination in hop-by-hop fashion, taking a new routing decision at each intermediate node. Routing decisions for data packets are based only on regular pheromone. This means that they only follow the reliable routes that are the result of ant based sampling, and leave the virtual pheromone information that is the result of information bootstrapping out of consideration. The combination of the reactive route setup and the proactive route maintenance processes leads to the availability of a full mesh of such reliable routes between the source and destination of each session.

Nodes in AntHocNet forward data packets stochastically, based on the relative values of the different regular pheromone entries they have available for the packet's destination. The probability P_{in}^d for a node i to pick next hop n when forwarding a packet with destination d is given in the formula of equation 4.6. This formula is very similar to the one used for reactive forward ants (see equation 4.1), but uses a different parameter, β_3, for the power function of the pheromone values. This way, the relative preference for the best routes can be adapted separately for data and for ants (as is common practice in ACO routing algorithms, see section 3.2).

$$P_{nd} = \frac{(\tau_{in}^d)^{\beta_3}}{\sum_{j \in N_i^d} (\tau_{ij}^d)^{\beta_3}}, \quad \beta_3 \geq 1 \tag{4.6}$$

By adapting the β_3 parameter, one can make data forwarding less or more greedy with respect to the best available routes. By setting β_3 low, data is spread over multiple routes, considering also low quality ones. Using multiple routes for data forwarding can improve throughput, as the data load is spread more evenly over the available network resources (see also subsection 2.4.3). By setting β_3 high, on the other hand, data is concentrated on the best routes. This can also be a good choice, since the routes that according to the ant sampling give the best performance, are exploited as much as possible. In our experiments, we normally keep β_3 on 20, which is relatively high and only allows data load spreading when there are several good routes of more or less equal quality. In chapter 5, we also compare results when using different values for β_3.

4.2.5 Link failures

In AHWMNs, link failures can occur due to physical changes such as the movement or disappearance of a node, or due to changes that influence the connectivity of the wireless communication, such as an increase of radio interference or

a decrease in the used transmission power. Since AHWMNs are usually highly dynamic, such events are expected to occur frequently, and AHWMN routing algorithms should be prepared to deal with them effectively. The components of AntHocNet described so far already offer some basic protection against link failures. The reactive route setup process allows source nodes to rebuild entire routes if needed, and the proactive route maintenance process offers protection in a proactive way through the creation of new paths, which can serve as backup routing possibilities. In this subsection, we outline further mechanisms in AntHocNet that are specifically aimed at dealing with link failures.

The first step in dealing with link failures is their detection. In AntHocNet, link failures are detected if lower layer protocols report the failure of the unicast transmission of a control or data packet, or if a node fails to receive periodic hello messages from its neighbors. Once a failure is detected, the next step is to take action to neutralize its effect. In AntHocNet, the action to be taken depends on the way the failure was detected. If the detection was through the failed transmission of a control packet or through the missed reception of hello messages, the node that detected the link failure broadcasts a link failure notification message, in which it warns downstream nodes about changed routes. If the detection was through the failed transmission of a data packet, the node starts a local route repair process in order to repair the route to the destination of the failed data packet. A final action that can be taken is the sending of a unicast warning message. These are messages that are used when an earlier broadcast link failure notification message got lost. Below, we first discuss the detection of link failures, then the use of link failure notification messages, next the process of local route repair, and finally the use of unicast warning messages.

Detecting link failures

Link failures can be detected through the failed unicast transmission of control or data packets, or via the use of hello messages. Detection through a failed unicast transmission is straightforward. MAC layer protocols usually have mechanisms that inform it about the success or failure of a unicast transmission. For instance, the IEEE 802.11 DCF protocol, which is often used in AHWMNs (see subsection 2.3.3), requires receiving nodes to send an acknowledgement upon successful reception of a unicast transmission. AntHocNet gives MAC layer protocols the possibility to report the failure of a transmission, and assumes in that case that the corresponding link has failed.

Relying solely on this mechanism to detect link failure is not satisfactory however. First of all, it does not allow to detect a link failure in advance, but only at the moment that it causes damage. Second, on a more technical note, many implementations of MAC layer protocols do not include the possibility to warn higher layers about a failed transmission. Therefore, AntHocNet also uses hello messages to detect link failures. These are messages that are sent out by all the nodes of the network asynchronously at a fixed interval of t_{hello} seconds. When a node i receives a hello message from a new node j, it can assume that j is its neighbor, and create an entry $\mathcal{N}_{ij}$ for j in its neighbor

table, indicating in it the last time that it has heard from j. It also makes an entry $\mathcal{T}_{ij}^{j}$ in its pheromone table, indicating that there is a one-hop route from i to j over next hop j. After this, i expects to receive a message from j at least every t_{hello} seconds. If i does not hear from j for a certain number of t_{hello} second intervals (set to 2 intervals here), i assumes that the wireless connection to j has disappeared.

In AntHocNet, hello messages are not only used to detect link failures, but also to carry pheromone information in the pheromone diffusion process (see subsection 4.2.3). This means that hello messages in AntHocNet are larger than those used in many other routing algorithms (such as e.g. in AODV [213], see subsection 2.4.2). For link failure detection, this can actually be an advantage. It has been pointed out in [56] that since hello messages are usually smaller than data packets, they can more easily be received correctly over shaky wireless connections, and therefore give a false image of link availability for data packets. The authors propose exactly the use of larger hello messages to get a better image of the real network topology.

Link failure notifications

When a node i detects that the link with a neighboring node j is lost, it removes j from its neighbor table. Then, it updates its pheromone table $\mathcal{T}_i$, building a link failure notification message in the process. It scans its pheromone table to control which destinations d have a non-zero regular pheromone value τ_{ij}^{d} (i.e., for which destinations d neighbor node j is used as a next hop from i). For each such destination, i sets τ_{ij}^{d} to 0. Furthermore, it checks whether the lost pheromone τ_{ij}^{d} was the best or only regular pheromone value available for d. If this is the case, it adds the address of the destination d to the link failure notification message, together with the new best regular pheromone it has available for d. If τ_{ij}^{d} was the only non-zero regular pheromone entry for d, this is also indicated in the link failure notification message.

Once the link failure notification message is fully constructed, it is broadcast. All of i's neighbors receive the message, and update their routing tables for the routes going over i to the involved destinations, using the new estimates reported in the message. To this end, they use the same formula that is applied for information bootstrapping in the pheromone diffusion process, as given in equation 4.4. In case they in turn loose their best or only route to one of the involved destinations due to the reported failure, they in turn construct their own link failure notification message, in the same way as i did, and broadcast it further. This way, all involved nodes eventually get warned and can update their pheromone tables.

Local route repair

When a node i detects a link failure through the failed transmission of a data packet, and i does not have any alternative routing information available for the destination d of this data packet, i does not include d in the link failure

notification it sends out. Instead, it starts a local route repair process to try to repair the route to d, so that the data packet can still be delivered.

At the start of the local route repair process, i creates a repair forward ant. Repair forward ants are identical to reactive forward ants, and are forwarded in the same way: they are broadcast when no routing information is available, and are otherwise unicast to a stochastically chosen next hop using the formula of equation 4.1. The only real difference with reactive forward ants is that repair forward ants can only be broadcast a limited maximum number of times (we set this number to 2). Therefore, they can only travel far if they are unicast over existing pheromone. Concretely, this means that repair forward ants need to stay close to existing routes in order to reach the destination, so that they really focus on the repair of the lost route. Upon arrival at the destination d, the repair forward ant is converted into a repair backward ant, which travels back to the node i that launched the local route repair process. It does so in exactly the same way as a reactive backward ant traveling back to its source (see subsection 4.2.2), updating regular pheromone entries on the way. Once the repair backward ant is back at the original node i, this node can send its stored data packet to d.

Node i uses a timer to decide how long to wait for a repair backward ant. The value of this timer is an estimate of the time it takes to go from i to d and come back, and is calculated as shown in equation 4.7. In this equation, t_{hop} is a fixed delay value per hop (set to 50 milliseconds), and h_{ij}^{d} is the number of hops to the destination as reported by the backward ants and stored in i's pheromone table (see subsections 4.2.1 and 4.2.2). The multiplication with 2 is to account for the way to go to the destination and come back. If no backward ant has been received before the timer runs out, i discards the stored data packet, and broadcasts a link failure notification about destination d.

$$timer = 2 * t_{hop} * h_{ij}^{d} \tag{4.7}$$

Unicast warning messages

A final aspect of dealing with link failures in AntHocNet is the use of unicast warning messages. These are emergency messages that are needed when link failure notification messages are not delivered correctly. This can happen because link failure notifications are broadcast. The IEEE 802.11 DCF MAC layer protocol, which is very often used in AHWMNs, does not provide any guarantees for the delivery of broadcast messages. This makes broadcast transmission a lot less reliable than unicast transmissions, which are supported with extra mechanisms to improve reliability (see subsection 2.3.3). Suppose now that a node i has lost its only route to a destination d due to a link failure, and warns other nodes about this in a link failure notification message as described above. If a neighboring node n, which has a route to d using i as next hop, does not receive this message correctly, it will continue sending data packets for d to i. At this point, i cannot forward the data packets. It therefore answers to the data packet by unicasting a short warning message to n, indicating that it has

no routing information for d. Upon reception of this message, n removes the erroneous routing information from its pheromone table.

4.2.6 Routing metrics

So far, we have not given any details about how paths are evaluated in AntHocNet, and have instead talked in terms of a generic cost value. In principle, this generic cost value can be replaced by any possible route cost metric. Concretely, we have explored the use of the following ones: the number of hops, the end-to-end delay, a combination of hops and end-to-end delay, and a metric based on the signal-to-interference-and-noise ratio of links along the route. While the calculation of the number of hops is trivial, the other three are a bit more complicated. Therefore, we explain in what follows for each of these metrics how a node i locally estimates the cost c_i^j of the link to its neighbor j. How local estimates of link costs are then combined into full route costs has been explained earlier in the description of the working of reactive backward ants (see subsection 4.2.2). In the experimental results reported in chapters 5 and 6, we normally use the signal-to-interference-and-noise ratio metric, as this gave the best results. Comparisons with versions of AntHocNet using the other metrics will also be reported in chapter 5.

End-to-end delay

When using the end-to-end delay cost metric, the cost estimate c_i^j maintained locally by node i reflects the expected delay incurred by a data packet when following the wireless link from i to its neighbor j. Concretely, c_i^j is calculated by the formula given in equation 4.8. In this formula, q_{mac}^i is the number of packets that are currently in queue at the node to be sent, and $\hat{t}_{mac}^i$ is an estimate of the time it takes to send one packet via unicasting. $\hat{t}_{mac}^i$ is calculated as a moving average of the time elapsed between the arrival of a packet at the MAC layer and the end of a successful transmission, which is indicated by an acknowledgement received from the next hop. This is shown in equation 4.9, where t_{mac}^i is the latest observed send time, and η is a parameter defining how quickly the moving average adapts to new observations (η is kept on 0.7).

$$c_i^j = (q_{mac}^i + 1)\hat{t}_{mac}^i \tag{4.8}$$

$$\hat{t}_{mac}^i \leftarrow \eta \hat{t}_{mac}^i + (1-\eta)t_{mac}^i, \quad \eta \in [0,1] \tag{4.9}$$

As can be seen from equations 4.8 and 4.9, the calculation of c_i^j here is in fact independent of j. This is because we assumed network nodes that have only one wireless interface which is an omnidirectional antenna. In this case, the data traffic for all next hops needs to go over the same outgoing queue, and needs to access the same wireless channel, so that a packet queuing to be sent to a node j might need to wait behind packets for any other neighboring node, experiencing also their delays.

End-to-end delay combined with number of hops

We also considered the possibility to combine the end-to-end delay with the number of hops. While the end-to-end delay is adaptive to the local traffic situation, it can be quite unstable, showing large oscillations due to variations in the quality of the wireless channel and local interference. The number of hops, on the other hand, is not adaptive, but is a stable metric. The goal of combining both is to have a metric that is both adaptive and stable.

The formula for the calculation of c_i^j using the combined metric is given in equation 4.10. The first part of this formula corresponds to the calculation of the delay, and is identical to the formula of equation 4.8. The second part of the formula reflects the number of hops. While the number of hops to reach neighbor j from node i is obviously always 1, here we use a different constant value, namely t_{hop}. This is a fixed estimate of the time needed to take one hop in unloaded conditions (we kept t_{hop} on 0.003 sec). Using the constant t_{hop}, rather than 1, allows to scale the number of hops to the same order of magnitude as the time estimation.

$$c_i^j = (q_{mac}^i + 1)\hat{t}_{mac}^i + t_{hop} \tag{4.10}$$

Signal-to-interference-and-noise ratio

The signal-to-interference-and-noise ratio (SINR) between a node i and a node j is the ratio between the strength of the signal received by node i from node j and the general noise and interference from other radio signals present around i. This value can be calculated at the physical layer of the node. SINR is a crucial factor defining the success of a wireless reception. When SINR is high, reception has a high probability of being successful, whereas when it is low, reception is impossible. In between there is a range for which reception is possible with some probability. Note that also factors other than SINR can influence reception, so that it is sometimes also possible to have bad reception when SINR is high (see e.g. [15]). Nevertheless, SINR is an important indicator of link quality.

When using SINR to define c_i^j in AntHocNet, we are not interested in fine variations in the SINR level, but rather in a coarse grained distinction between "good" and "bad" wireless links: we want to capture the difference between links on which reception has a high probability of being successful, and links on which reception is possible but with a lower probability of success. Therefore, we apply a simple approach using a critical SINR value $SINR_c$ as threshold, in which links with an SINR value lower than $SINR_c$ are penalized. Concretely, we use the formula of equation 4.11, where c_i^j is set to 1 for links with SINR higher than $SINR_c$ (this corresponds to using normal hop count as a metric), and to a constant value $c_{const} > 1$ for links with SINR lower than $SINR_c$. In simulation tests using IEEE 802.11b radios sending at 2Mbps, we empirically set $SINR_c$ to 17dB and c_{const} to 3. The value of 17dB for $SINR_c$ is in line with critical values of SINR found in empirical research on wireless LANs using the same radio technology [200].

$$c_i^j = \begin{cases} 1, & \text{if } SINR > SINR_c \\ c_{const}, & \text{if } SINR \leq SINR_c \end{cases} \tag{4.11}$$

4.3 Further Discussions

In this section, we provide further discussions related to the AntHocNet routing algorithm. First, we consider AntHocNet in the light of the RL framework presented in chapter 3, and discuss its particular strategies for information gathering from this point of view. Then, we take a look at the different challenges for AHWMN routing that were pointed out in chapter 2, and investigate qualitatively how AntHocNet deals with these. Next, we discuss how AntHocNet relates to other existing routing algorithms. In particular, we search in how far components of AntHocNet are also used elsewhere. Finally, we write a few words about mechanisms that were present in older versions of AntHocNet, and about why they were discarded.

4.3.1 AntHocNet and reinforcement learning

In chapter 3 we have described RL, an important class of problems in machine learning, and we have explained how the problem of routing fits into this framework. Then, we have discussed two basic solution methods for RL problems, namely dynamic programming and Monte Carlo sampling, each with their own advantages and disadvantages, and we have shown how existing routing algorithms relate to them, with distance vector routing being a direct implementation of dynamic programming, and ACO routing relying mainly on Monte Carlo sampling. Subsequently, we have also described a more advanced learning method, temporal difference learning, which combines elements of both basic methods, and we have shown the Q-routing algorithm that was based on it. In what follows, we investigate how the AntHocNet routing algorithm proposed in this chapter relates to all of this. In particular, we describe how AntHocNet uses both the Monte Carlo sampling and dynamic programming learning methods, but combines them in a way that is different from temporal difference learning.

Monte Carlo sampling is used extensively in AntHocNet, since it is the learning method applied in ACO routing, which was the main source of inspiration of our algorithm. This is most evident in the proactive ant sampling subprocess of the proactive route maintenance process (see subsection 4.2.3). In this subprocess, ants are regularly sent out by the source node of each session in order to sample a path towards the destination of the session. This is very much in line with the continuous path sampling done in other ACO routing algorithms. Apart from this, also the reactive route setup process (see subsection 4.2.2) and the local route repair process (see subsection 4.2.5) use a form of Monte Carlo sampling: the reactive and repair forward ants used in these processes each take a single sample of a path through the network, and use it to set up a new route. As was explained earlier in subsection 3.3.2, this approach of using Monte Carlo sampling by taking a single sample of a path through the network

is also applied in many existing reactive routing algorithms. An important point to note here is that AntHocNet is more consistent in its use of sampling, since both reactive and repair forward ants always go all the way till the destination. In most existing reactive routing algorithms, including the AODV and DSR protocols described in subsection 2.4.2, this is not the case: RREQ messages that are searching for a path to the destination can be returned by intermediate nodes that have routing information about the destination available. At that point, the obtained routing information relies on the estimate provided by the intermediate node, so that a form of information bootstrapping is applied.

The most important advantage of using Monte Carlo sampling here is that it provides a high level of reliability. This is because all routing information is the result of direct experiences, giving a certain guarantee about its correctness. A disadvantage is that it can be inefficient. The need to send sampling packets from source to destination can lead to high levels of overhead. The use of IEEE 802.11 DCF as MAC layer mechanism can deteriorate this problem, because this protocol creates a lot of overhead for each sent packet, making the transmission of multiple small packets particulary problematic. Traditional reactive routing algorithms deal with the efficiency issue by using just a single sample. AntHocNet, on the other hand, improves efficiency by combining Monte Carlo sampling with a supporting dynamic programming process.

AntHocNet uses dynamic programming in the pheromone diffusion subprocess of its proactive route maintenance process (see subsection 4.2.3). This subprocess works more or less in the same way as distance vector routing algorithms do, using information bootstrapping in each node to derive routing information from estimates calculated by neighboring nodes. Such an approach has as an advantage that it is highly efficient, as the information obtained by each node is optimally reused in the calculation of the information needed by other nodes. In AntHocNet, this efficiency is further increased by piggybacking routing updates on top of hello messages, which avoids the transmission of multiple small control packets. An important disadvantage of using dynamic programming and information bootstrapping is that it can lead to processes that are slow to converge, so that routing information can be temporarily unreliable. This is especially a problem in dynamic situations. In existing distance vector routing algorithms for AHWMNs, such as DSDV (see subsection 2.4.2), extra techniques are applied to ensure reliability in the face of the highly dynamic network environment. Unfortunately, these techniques introduce extra overhead, and can severely reduce the efficiency of the process. Therefore, in AntHocNet, we have chosen a different strategy: the dynamic programming part of AntHocNet is kept very simple and lightweight, and is not expected to provide information that is 100% reliable. This way, we focus maximally on its efficiency, and do not worry about its unreliability. Instead, we are aware that the information produced by the process can contain errors and therefore we do not use it directly for routing. We use it as a guideline for the Monte Carlo sampling of the proactive ant sampling subprocess. Using this guideline, the ants do not have to explore the whole network, but can concentrate on routes that are suggested by pheromone diffusion. This reduction in the need

for exploration makes the sampling process more effective, so that less ants are needed, leading to better efficiency. This way, we obtain an adaptive algorithm that combines the efficiency of dynamic programming with the reliability and robustness of Monte Carlo sampling.

Note that the way Monte Carlo sampling and dynamic programming are combined here is very different from temporal difference learning methods. In n-step temporal difference learning (see subsection 3.3.3), the learning agent takes a sample of a few steps, after which it arrives in an intermediate state i, where it reads the local value estimate, which it uses to bootstrap on. This approach can be highly efficient, but does not avoid the potential unreliability of dynamic programming, since it still uses information bootstrapping. It is interesting to see that the temporal difference learning approach to information gathering is similar to the returning of RREQ messages by intermediate nodes in reactive routing algorithms as described above: the RREQ samples a path till an intermediate node that has routing information available about the destination, at which point it bootstraps on this information and returns to the source. In AntHocNet, we aimed to be more consistent, and keep sampling and information bootstrapping strictly separate.

4.3.2 Challenges for routing in AHWMNs

In chapter 2, we have defined a number of challenges for routing in AHWMNs. These included adaptivity, robustness, efficiency and scalability. Here, we investigate qualitatively how AntHocNet is equipped to deal with each of these.

Adaptivity is very important in AHWMNs, as the dynamic network environment constantly presents the routing algorithms with new changes. Adaptivity is in AntHocNet provided in two different ways. On the one hand, the algorithm has a wide range of reactive mechanisms at its disposal. These include the reactive route setup process and the different mechanisms to deal with link failures, such as link failure notifications, local route repair, and unicast warning messages. These provide the algorithm with tools to react immediately in case a disruptive event takes place. On the other hand, AntHocNet applies also proactive mechanisms, in its proactive route maintenance process. These allow the algorithm to take adaptive actions without the need for being prompted by an event. Proactive actions can avoid problems with disruptive events in the future, by providing backup routes, or exploit new possibilities that arise from the changes in the environment.

Robustness is in general obtained from the extensive use of ant based full path sampling, a practice that is taken over from ACO routing. As was explained in chapter 3, using full path sampling as a method to gather routing information leads to increased robustness in two ways. First of all, each individual sample is unimportant, so that the loss of control packets only has a marginal effect, not immediately leading to erroneous routing information. Second, since ants always sample full paths, they provide a certain guarantee that the path actually exists and the reported information is correct. This is in contrast with distance vector routing and link state routing, which are both in principle more vulnerable in

a highly dynamic environment.

Efficiency is taken care of in AntHocNet in two ways. First of all, the algorithm's hybrid architecture combining reactive and proactive components allows to concentrate on the routing information that is most needed. Second, the use of a highly efficient dynamic programming approach, which uses piggybacking on top of hello messages, as a way to guide ant based sampling in the proactive route maintenance process allows to improve efficiency during the gathering of routing information.

Good scalability is expected to arise from the provided efficiency, adaptivity and robustness. We refer here to the empirical results in chapter 5 that show the scalability of the algorithm compared to other, state-of-the-art routing algorithms for AHWMNs.

4.3.3 AntHocNet related to other routing algorithms

In this subsection we take a look at routing algorithms that are related to AntHocNet. We isolate mechanisms that are used in AntHocNet, and investigate to what extent they are also applied in other AHWMN routing algorithms. In particular we will talk about multipath routing, the use of path sampling, and the approach to combine reactive route setup with proactive route improvement.

Multipath routing is used in many AHWMN routing algorithms. An overview has been given earlier in subsection 2.4.3. The main objectives when using multipath routing is to improve failure resilience by providing backup paths, and to improve throughput. The former is inherent to all algorithms that set up multiple paths. The latter can only be obtained when data traffic is spread over the multiple paths. In AntHocNet, this is possible by choosing a low value for the parameter β_3 in equation 4.6, which leads to a stochastic spreading of the data load according to the relative quality of the different available paths. Such an approach is typical for ACO routing algorithms, and can therefore also be found in some of the other algorithms that apply ACO routing for AHWMNs, such as ARA [119] and Termite [225] (nevertheless, many ACO routing algorithms also forward data deterministically over the best path; see subsection 3.2.6 for an overview). Outside the area of ACO routing, adaptive data load spreading is far less common. Most algorithms that do spread data over multiple paths do so in a simple, even way (see e.g. [164]). A few algorithms explore the idea of adaptive data load spreading depending on the estimated quality of the paths (e.g. [108, 269]). Stochastic data load spreading is to the best of our knowledge unexplored outside the area of ACO routing.

The use of path sampling as a strategy for obtaining routing information has been discussed amply in chapter 3 and in subsection 4.3.1. AntHocNet applies sampling both reactively, using a single sample to set up a route between source and destination, and proactively, using repeated samples to update existing routing information and explore new possibilities. As was mentioned before, reactive use of single path samples is quite common in AHWMNs. It is at the basis of the working of some important algorithms, such as AODV [213] and DSR [140] (see subsection 2.4.2). Proactive use of repeated path sampling is

far less common. Some algorithms apply it in a limited way, using sampling to get up-to-date information about existing routes, but not to explore new ones. This is the case in [108, 269], and in many of the ACO routing algorithms for AHWMNs (see subsection 3.2.6). The use of sampling to proactively find new routing information is quite rare in AHWMNs, even for ACO routing algorithms. Exceptions are the Termite [225] and EARA [177] algorithms, in which ants (or in the case of Termite any kind of packets) can take random routing decisions, so that they leave existing routes, and start exploring new ones. Such random exploration is quite blind. A system where the exploration is guided such as in AntHocNet's proactive route maintenance process has to our knowledge not been explored outside the work presented in this thesis.

A hybrid strategy of combining reactive route setup with proactive route improvement like the one used in AntHocNet is not very common in AHWMN routing. It is applied in some ACO routing algorithms such as Termite and EARA through the use of random exploration decisions during the path sampling process, as is explained above. Outside the area of ACO routing, the approach can to some extent be found in the reactive routing protocol DSR. As was explained in subsection 2.4.2, DSR uses source routing, which means that each data packet carries the full route from its source to its destination, as a list of addresses. Nodes that are not on this route can overhear the data packet, and extract the routing information it is carrying. This allows these nodes to discover new routes. Unlike AntHocNet, however, DSR does not include any mechanism to verify the reliability of these new routes, and in experiments this mechanism has often been found ineffective, as it allows erroneous routing information to be copied by other nodes, leading to a quick "pollution" of routing tables throughout the network. Nevertheless, DSR's approach to route improvement has recently been taken over in two new routing algorithms: LQSR [92, 93] and Srcr [30]. Quite interestingly, both algorithms have been developed specifically for use in real WMN deployment projects: LQSR for Microsoft's MCL architecture, and Srcr for MIT's Roofnet project. A very different approach to proactive route improvement is found in the LUNAR algorithm [259], which is in essence a reactive algorithm in which improvements are obtained by repeating the route setup phase every 3 seconds. For efficiency reasons, the algorithm is limited to networks of maximally 3 hops. Like LQSR and Srcr, also LUNAR was developed based on experiences with a real WMN deployment project. The fact that proactive route improvement was chosen as the approach in several WMN deployment studies is an indication of its usefulness in realistic settings.

4.3.4 Older versions of AntHocNet

In the first papers about AntHocNet [73, 74], an older version of the algorithm was described. This version contains some important differences compared to the version described in this chapter. Specifically, it uses different mechanisms in the reactive route setup process and in the proactive route maintenance process. Here, we describe these different mechanisms, and explain why we dropped them.

The main difference in the reactive route setup process of the older version of AntHocNet is that not one but multiple routes are set up. To this end, nodes that receive multiple reactive forward ants belonging to the same route setup process do not immediately discard these duplicate ants. Instead, they compare the path traveled by each ant to that of the previously received ants of this route setup. If the number of hops and travel time of a newly received ant are both within an acceptance factor a_1 of those of the best previously received ant of the same route setup, the new ant is accepted and forwarded; otherwise, it is discarded. a_1 is set quite low (to 0.9), in order to only allow the best ants through and avoid too much proliferation of forward ants in the network. The multiple reactive forward ants arriving in the destination are converted into backward ants, which return to the source, so that a number of multiple, good paths are set up simultaneously. A problem with the approach is that due to the use of strict acceptance criteria when comparing to the best previously received ant (using the low acceptance factor a_1), the process can lead to a situation where the different resulting paths are all just small variations of the best one. In general, it is better to have more disjoint paths, as this gives better protection in case of link failures. To boost the creation of disjoint paths, a different mechanism is applied, which takes into account the first hop taken by each ant. If this first hop is different from those taken by previously accepted ants, we apply a higher (less restrictive) acceptance factor a_2 than in the case the first hop was already seen before (a_2 was set to 2). A similar strategy can be found in [186].

The strategy of setting up multiple routes during the reactive route setup process has as an obvious advantage that multiple routes are available from the start of the communication session, so that the session is better protected against link failures and can start data load spreading immediately. However, through experiments we experienced that it is hard to get a good balance between the number of routes that are obtained and the overhead that is created. High levels of overhead were often experienced. Therefore, we decided to restrict reactive route setup to the creation of just one single route, and to rely on proactive route maintenance to obtain multiple routes.

The proactive route maintenance process of the older version of AntHocNet is considerably different from the new one. It consists of only the proactive ant sampling subprocess, and does not apply pheromone diffusion. Proactive forward ants can be forwarded through unicasting or through broadcasting. The unicasting is done in the same way as in the current version of AntHocNet: a next hop is chosen probabilistically according to available pheromone information. However, since no pheromone diffusion is done, no virtual pheromone is spread out, and the only available pheromone information for the proactive forward ants is regular pheromone. This means that through pheromone guided unicasting, ants can check out existing routes, but not explore new ones. This is where the broadcasting comes into play. At each node, proactive forward ants have a small probability (set to 10%) of being broadcast. After such a broadcast the ant arrives in all the neighbors of the broadcasting node. This way, it can leave the currently existing paths and start the exploration of new ones. It is

possible that in these neighbors it does not find pheromone pointing towards the destination, in which case it is broadcast again. The ant will then quickly proliferate and flood the network, like reactive forward ants do. In order to avoid this, the number of broadcasts is limited to n_b (n_b was set to 2). If the ant does not find routing information within n_b hops, it is deleted.

Compared to the route exploration done in the current version of AntHocNet, this older mechanism has some important shortcomings. First of all, its exploration is completely blind: the broadcasts are done randomly, without using any information about whether it would be possible to find new good routes there. Second, it creates a lot of overhead. Due to the n_b possible broadcasts, each proactive forward ant can multiply quickly, leading to a lot of extra control packets in the network. Therefore, n_b needs to be kept quite low. As a consequence, however, exploration is more limited, and the number of exploratory moves cannot be more than n_b. The proactive route maintenance process adopted in the current version of AntHocNet is both more effective and more efficient.

4.4 Conclusion

In this section we have presented the AntHocNet routing algorithm for AHWMNs. AntHocNet is a hybrid algorithm that combines a reactive route setup process with a proactive route maintenance process. The reactive route setup is carried out at the start of a communication session or whenever the source of a current session has no more routing information available for the destination. It creates a single route for the session. The proactive route maintenance is run for the entire duration of the session. Its aim is to keep information about existing routes up to date and to explore new routes. Under impulse of this process, the initial single route created during reactive route setup is extended to a full mesh, and improved to exploit new possibilities in the changing AHWMN environment. Other features of AntHocNet are the possibility to do probabilistic data load spreading, and the use of a number of reactive components to deal with link failure, such as the transmission of failure notification messages and the possibility to execute local route repair.

Considered from a machine learning point of view, AntHocNet relies on two distinct strategies for information gathering, namely Monte Carlo sampling and dynamic programming. The Monte Carlo sampling approach is inherited from ACO routing, which was the main source of inspiration for AntHocNet. It is applied extensively throughout the different components of the algorithm. Collecting routing information through the sampling of full paths leads in general to reliable routing information. Dynamic programming is only used during proactive route maintenance. It is the basis of the pheromone diffusion subprocess, which uses information bootstrapping to spread earlier obtained routing information over the network. Using a dynamic programming approach allows to gather routing information in an efficient way. However, it can sometimes temporarily lead to erroneous information. Therefore, it is in AntHocNet combined

with full path sampling in order to improve reliability. The way ideas from dynamic programming and Monte Carlo sampling are combined in AntHocNet is novel in the area of machine learning.

The different mechanisms used in AntHocNet help the algorithm to deal with some of the important challenges of AHWMN routing that were pointed out in chapter 2, such as adaptivity, robustness and efficiency. Adaptivity is on the one hand provided by the availability of reactive algorithms such as the reactive route setup and the different mechanisms to deal with link failures, which make sure that disruptive events can always be dealt with. On the other hand, the use of proactive route maintenance allows to adapt to change in the environment before they cause disruptions. Robustness is in general provided through the use of full path sampling to establish routes for data traffic. Efficiency is obtained by combining the path sampling with a dynamic programming approach which allows to spread routing information in an efficient way.

Chapter 5

An evaluation study of AntHocNet

In this chapter, we present an evaluation study of the AntHocNet routing algorithm. We show results of an extensive set of simulation tests, in which we compare AntHocNet to a number of state-of-the-art AHWMN routing algorithms under a wide variety of different scenarios. We also present results of tests in which we introduce variations to the parameters and components of AntHocNet, in order to get a better understanding of the internal working of the algorithm. Since AntHocNet was in the first place developed for working in MANETs, we organized our simulation experiments according to common practice in the area of MANET routing research. In particular, we use open space scenarios in which nodes move according to typical MANET mobility patterns. This allows us to compare to existing algorithms on a fair basis. Later, in chapter 6, we present results of a different study, in which we use a WMN in an urban scenario.

The rest of this chapter is organized as follows. First, in section 5.1, we describe the setup of the evaluation study. Next, in section 5.2, we present the tests in which we compare AntHocNet to existing AHWMN routing algorithms. Finally, in section 5.3, we present results of the experimental investigation of the internal working of AntHocNet. The work presented in this chapter has been published in part in the earlier mentioned papers [73–75, 94, 95], as well as in [96] and in project deliverable [77].

5.1 Setup of the evaluation study

In this section, we describe the organization of the evaluation study presented in this chapter. First, we provide a general discussion regarding the use of simulation for the evaluation of AHWMN routing algorithms. Then, we talk about the simulator software we use. Next, we give specific details about the setup of the simulation studies we carry out. Then, we give a brief overview of

the routing algorithms we use for comparison. Finally, we discuss the measures we use to evaluate the results of the simulation studies.

5.1.1 On the use of simulation

For the evaluation of network algorithms, one can can consider two different options: an analytical study or an experimental one. For traditional telecommunication networks, analytical performance evaluation is a well studied topic [122]. Also for AHWMNs, there have been a number of interesting analytical studies, e.g. investigating physical properties such as the maximum possible throughput of a MANET [120], or the relationship between node density and connectivity [89] (see section 2.3 for descriptions of both studies). For AHWMN routing algorithms, analytical studies have been used for instance to compare load balancing properties of single path and multipath routing [109, 214] (see subsection 2.4.3), or to study the scalability of routing algorithms [231]. Such studies give interesting insights into some properties of algorithms, but are necessarily limited in scope. This is because AHWMNs are very complex environments. Mobility of network elements and interferences and loss of connectivity over wireless links cause constant changes in the network. Moreover, algorithms at different levels in the protocol stack are often quite complex (e.g. MAC protocols), precisely to deal with the dynamically changing environment, and can have unexpected interactions with each other. Analytical studies can therefore only be carried out under strict assumptions (e.g., no mobility, or perfect MAC mechanisms), which can have a strong impact on the obtained results.

Most of the research on AHWMNs therefore follows the second approach, in which new algorithms are evaluated through experiments. The basic idea in experimental research is to run the system under investigation and observe its behavior. When it is difficult to obtain sufficient observations from the real system, a possible alternative is to run the experiments in simulation. Generally speaking, simulation can be defined as the process of designing a model of a real system and conducting experiments with this model [136, 237]. Simulation is often used in computer network research, and has been particulary popular in AHWMN research. This is because it is expensive and technologically difficult to develop a real AHWMN testbed for research purposes, and because in simulation it is easier to carry out large and repeatable sets of tests. Nevertheless, in recent years, there has been a growing interest in real implementation tests as a way to validate and complement the results obtained in simulation. More about this will follow later in chapter 7.

An important issue when setting up experiments, be it in simulation or using a real implementation, is to choose the scenarios to be used in the study. Designing a scenario includes the definition of a number of variables, such as the size of the network, the movement patterns of the nodes, the data traffic between the nodes, the network protocols to be used, etc.. Each choice has an important impact on the network environment and on the measured performances, and it is therefore important that the scenarios are sufficiently representative for the applications that the network will eventually be used for. This is a problem in

AHWMN research, and especially in the field of MANETs. Until recently, the community remained rather vague about possible applications for this kind of networks, so that it was difficult to figure out what would be a realistic scenario. Hence, MANET algorithms have mainly been evaluated in experimental setups that make minimal assumptions about the environment and the use of the network: they use very simple scenarios, in which nodes move according to random patterns in a rectangular, open area and send data packets to each other at fixed rates. Only in the last few years there has been an increasing interest in experiments that use scenarios that are more complex and possibly more realistic. Especially urban scenarios are popular, as recently a number of real WMNs have been set up in urban environments, such as e.g. the public WMNs in San Francisco and Philadelphia. However, there are still considerably less studies available that use these different scenarios compared to those using simpler scenarios.

In the current chapter, our aim is to carry out a detailed evaluation study of AntHocNet, comparing it to existing state-of-the-art routing algorithms for MANETs and WMNs, and investigating its internal working. In order to obtain a better and more fair comparison with existing work in the research area, we decided to stick here with the common practice in the research community, and use simulation studies with simple scenarios that are similar to those used by other researchers. Later, in chapter 6, we will take a different approach and investigate the behavior of AntHocNet in a realistic urban scenario. Finally, in chapter 7, we go again a step further, and discuss the implementation of AntHocNet and other ACO routing algorithms in a real testbed.

5.1.2 The QualNet network simulator

When simulating a computer network, one needs to create models of the protocols and technologies that are used. Furthermore, in the case of AHWMNs, it is also necessary to develop models of the movement of the network nodes and of relevant physical phenomena, such as radio wave propagation and interference. Comparative tests have shown that differences in the used models can lead to significant differences in observed results (see [54]). It is therefore important that all the models are detailed and accurate. Such accurate models are provided in an integrated way in a number of different network simulator software packages that are available to the research community. These include ns-2 [262], OPNET [207], GloMoSim [263], SWAN [175] and QualNet [232].

For the work presented in this thesis, we used the QualNet simulator. This is a commercial simulation tool developed by Scalable Network Technologies as the follow up of the older GloMoSim simulator. QualNet offers a number of important advantages when compared to other simulators. First of all, it includes a wide range of models to support the simulation of both wired and wireless networks, and comes with an extensive library that is specifically related to MANETs and WMNs. Second, it uses a clear, modular organization, following the layered TCP/IP architecture. This makes it easy to understand and to plug in new protocols. Third, being a commercial product, it comes with

good documentation and support. Fourth, it is equipped with several graphical user interfaces, to support the design of new algorithms, the setup of simulation studies, etc.. Finally, QualNet has been specifically designed to simulate large AHWMNs, something that has traditionally been a problem in other network simulators such as ns-2.

5.1.3 Simulation scenarios

The aim of this chapter is to investigate the performance of AntHocNet in a range of different scenarios. In order to do tests in a controlled way, we define a common base scenario, from which all other scenarios are derived by varying relevant parameters such as the node speed, the data send rate, the network size, etc.. This base scenario was designed in line with the most commonly used scenarios in the research area, in order to allow a fair comparison with other algorithms. Here, we describe the properties of the base scenario. Later, in each of the experiments, we specify how the different applied scenarios were derived from it.

We consider a network of 100 nodes that move in a rectangular area of $2400 \times 800m^2$. It is an open area, in the sense that there are no obstacles that could limit node mobility or signal propagation. Node movements are defined according to the RWP mobility model (see subsection 2.3.1 and [140]). Under this model, each node starts from a randomly chosen initial position in the area, and independently chooses a random speed between a given minimum and maximum speed, and a random destination. Then, it moves at the chosen speed towards the chosen destination in a straight line. Upon arrival, it remains static for a fixed pause time, after which it chooses a new speed and destination. We use a minimum and maximum speed of respectively 0 and $10m/s$, and a pause time of $30s$. Each experiment has a duration of $900s$, and is repeated 20 times, using different random instances of the same scenario. Data traffic is generated by constant bit rate (CBR) sessions: 20 data sessions are run between randomly chosen source and destination nodes. Sessions start between 0 and $180s$ after the beginning of the simulation, and run till the end. Each session generates 4 packets of 64 bytes per second.

For the simulation of radio propagation, we use the two-ray signal propagation model, which is a common approach to model the propagation of wireless signals in open space [167]. The two-ray model assumes that a signal reaches a receiver over two different paths: one direct and one reflected over the ground. Compared to the signal that travels along the direct path, the one that travels along the reflected path arrives with a certain delay, which depends on the distance between the source of the signal and the receiver. Depending on this delay and the relative phase, the reflected signal can reinforce or disturb the direct signal. As a consequence, the two-ray model considers a different decay of the signal strength depending on the distance: it applies a decay of order R^2 (where R is the transmission distance) for short distance transmission and of order R^4 for long distance transmissions.

At the physical layer, we use the IEEE 802.11 protocol, with data transmis-

sion rate of $2Mbit/s$. The estimated radio range is $250m$. At the MAC layer, we use the IEEE 802.11 DCF protocol, which was described in subsection 2.3.3. Finally, at the transport layer, we use the UDP protocol, rather than TCP, as is common in AHWMN research. There are several reasons not to use TCP. First of all, TCP is known to behave badly in AHWMNs (see subsection 2.3.4). Second, TCP's various mechanisms to control the flow and to resend packets can influence the results in unforeseen ways so that it becomes difficult to evaluate the performance of the routing algorithms. Finally, UDP is the normal transport protocol to be used in combination with CBR applications.

5.1.4 Algorithms used for comparison

In the tests of section 5.2, we investigate how well AntHocNet performs in comparison to existing AHWMN routing algorithms. To this end, we have chosen a selection of algorithms that are representative for the wide class of available routing algorithms in the research area. The selection includes AODV, OLSR and ANSI. Here, we briefly discuss the choice for each one of them.

AODV [213] is a reactive routing algorithm. It is under investigation for standardization by the IETF MANET group [6], and has gained considerable status as the de facto standard routing algorithm for MANETs and WMNs. Most existing work in this research area uses AODV as a benchmark for comparisons, and we have followed this common practice. A short description of AODV can be found in subsection 2.4.2. Originally, we also included the DSR [140] routing algorithm in our comparative study. This is a different reactive algorithm that has also received a lot of attention in the community. However, the results obtained with DSR were quite bad and were therefore not included here.

OLSR [61] is a proactive routing algorithm. While proactive routing has often been considered a less good approach in AHWMNs [42], OLSR has received considerable attention since its publication in 2001. It is one of the most studied proactive algorithm, and it is together with AODV one of the prime candidates for standardization by the IETF MANET group. While it is less used than AODV as a benchmark in comparative studies, we consider it important to use also a proactive algorithm in our evaluation of AntHocNet, and therefore decided to include OLSR. A description of the OLSR algorithm has been given earlier in subsection 2.4.2.

ANSI [220] is, like AntHocNet, an ACO routing algorithm for AHWMNs. Due to this common source of inspiration, it has more similarities with AntHocNet than the previously mentioned algorithms. It uses full path sampling to gather routing information, sets up multiple routes, and applies to some extent a hybrid approach, where initial routes are set up reactively, and proactive sampling is used to keep this initial routing information up-to-date. The full algorithm, however, is quite different from AntHocNet. ANSI was chosen to make comparisons because we consider it important to also include an ACO routing algorithm. A brief description of ANSI has been given in subsection 3.2.6.

5.1.5 Evaluation measures

Here, we describe the measures that we use to evaluate the performance of the different routing algorithms in the experiments. We distinguish between measures of effectiveness and measures of efficiency. The measures we apply are all derived from recommendations made by the IETF MANET standardization group [62].

Measures of effectiveness are external measures of performance: they measure to what extent the algorithm manages to execute the task it was designed for. We use three different measures of effectiveness. The first one is the data delivery ratio. This is the fraction of correctly delivered data packets versus sent packets. This is an important measure in AHWMNs, as due to the constant changes in the topology it is difficult to deliver all data packets. As a second measure of effectiveness, we consider the end-to-end packet delay. This is the cumulative statistical measure of the delays experienced by packets traveling between their source and destination. Finally, as a third measure, we use the average delay jitter. This is the variation in the time interval between the arrivals of subsequent packets. It is calculated as shown in equation 5.1, where t_i is the time of arrival of the i^{th} packet, and n is the total number of packets received by a destination during a communication session. Delay jitter is an important measure for QoS applications, and also gives an indication of the algorithm's ability to respond smoothly to disruptive events in the network. In this sense, it is a measure of robustness and adaptivity.

$$jitter = \sum_{i=2}^{n} |(t_i - t_{i-1}) - (t_{i-1} - t_{i-2})| \tag{5.1}$$

Measures of efficiency are internal evaluation measures. They are concerned with the generated overhead. We consider two different measures of efficiency. The first is the overhead in number of packets. It is the total number of control packets transmitted by the nodes of the network versus data packets delivered at their destination. The second is the overhead in number of bytes. This is the total number of control bytes transmitted versus data bytes delivered. While both of these are closely related, the difference between the two measures is important in AHWMNs. As has been pointed out earlier in subsection 2.3.3, the limitations in available bandwidth in AHWMNs are for a large part due to MAC layer issues, rather than to intrinsic limitations in the possible data transmission rate. MAC layer overhead is incurred with the transmission of each packet, and the number of transmitted packets is therefore important when it comes to measuring efficiency. On the other hand, sending more bytes leads to longer channel occupancy, and also requires nodes to spend more energy. So also the overhead in number of bytes has its importance.

5.2 Comparisons to other routing algorithms

In this section, we present the results of a range of tests in which we compare AntHocNet to representative routing algorithms for MANETs and WMNs. For each of the tests, we derive scenarios from the above described common base scenario. We vary a different environmental property each time, in order to investigate its effect on the performance of the algorithms independently. We do tests changing the node mobility, the data traffic, the node density, and the network size. To change the node mobility, we run separate tests varying the maximum speed in the RWP mobility model, varying the pause time of the RWP model and using a different mobility model, namely the Gauss-Markov model (GM). To change the data traffic, we run tests varying the data send rate and the number of sessions. To change the node density, we vary the size of the area in which the nodes of the network move. Finally, to change the network size, we vary the number of nodes and the network area simultaneously.

5.2.1 Varying the maximum node speed for RWP mobility

In this first set of experiments, we vary the maximum node speed in the RWP mobility model, from $1m/s$ ($3.6km/h$, or the speed of a leisurely walk) up to $30m/s$ ($108km/h$, or the speed of a car on a highway), using as intermediate values 2, 5, 10 and $20m/s$. Varying the maximum speed in the RWP mobility model affects the node mobility directly in an obvious way: the higher the speed, the higher the mobility. Higher mobility leads to more frequent changes in the network environment, and therefore to more difficult scenarios. The results of the experiments are shown in figure 5.1, where we report (a) the delivery ratio, (b) the average end-to-end delay, (c) the average delay jitter, (d) the overhead ratio in number of packets, and (e) the overhead ratio in number of bytes.

The results for delivery ratio reflect the increasing level of difficulty of the scenarios: for all algorithms the delivery ratio decreases with increasing node speeds. The best results are obtained by AntHocNet, that even at the highest speeds is able to deliver almost 90% of all packets. This shows that AntHocNet is able to adapt well to the fast changes in the highly dynamic environment caused by high node mobility. AODV and ANSI give less good results, and with ANSI, the performance gap grows as the speed increases. The worst results are obtained by OLSR, that gives a delivery ratio of less than 40% for the highest speed scenario. This confirms the earlier mentioned observation that proactive routing algorithms have a hard time keeping up in highly dynamic environments (see subsection 2.4.1).

The results for average delay show similar trends. Like for delivery ratio, AntHocNet gives better results than AODV and ANSI. However, the gap in performance between AntHocNet and AODV decreases slightly for the highest delays. For ANSI, on the other hand, the performance gap increases, even stronger than when considering delivery ratio. One striking difference with the delivery ratio results is that for delay, OLSR gives good performances for the highest speed scenarios. For 20 and $30m/s$, OLSR even outperforms AODV

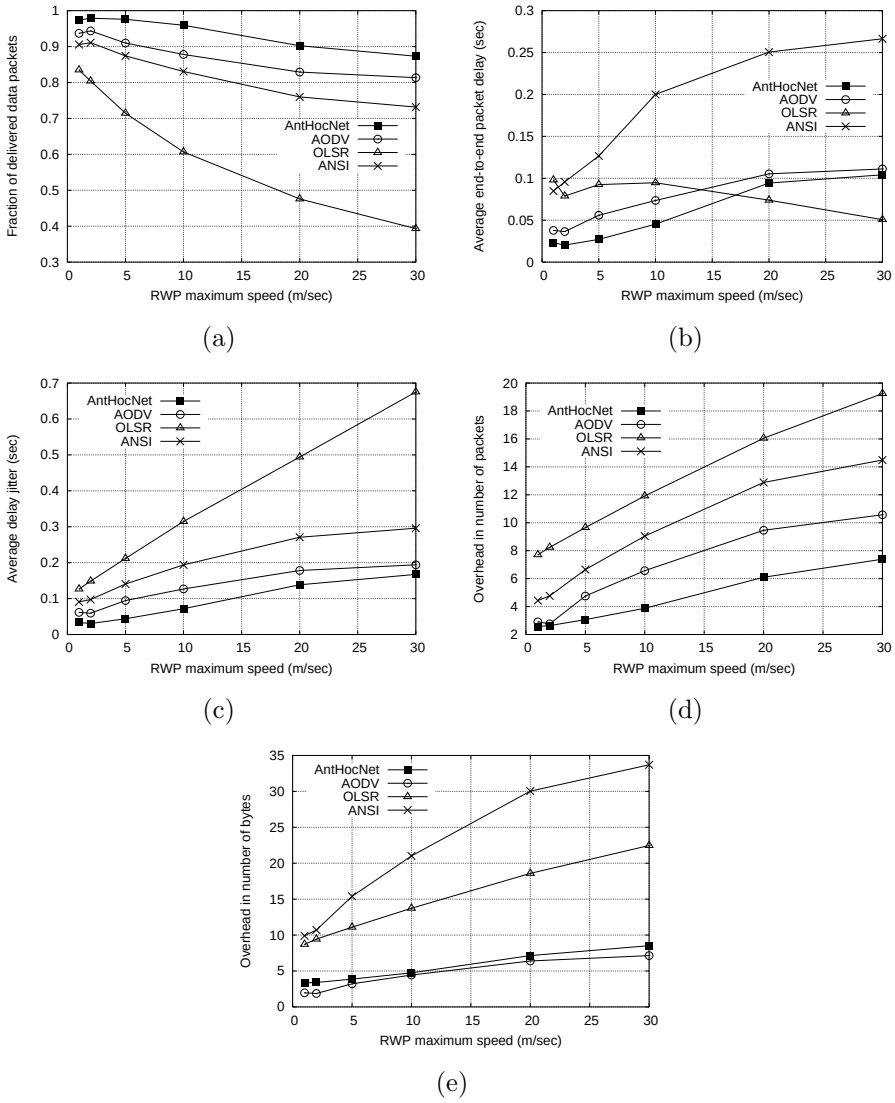


Figure 5.1: Results for AntHocNet, AODV, OLSR and ANSI using different values for the maximum speed in the RWP mobility model: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

and AntHocNet. These results need to be read with some caution though. The good delay results are obtained in a situation where OLSR delivers only a very low percentage of the packets, and have therefore little value.

The results for average delay jitter follow the same trend as those for delivery

ratio, with AntHocNet performing better than AODV, ANSI, and OLSR, in that order. The delay jitter measures the variation in the time between arrivals of subsequent data packets. It is an indicator of robustness and adaptivity, as low jitter shows that the algorithm is able to limit the effect of disruptive events.

Finally, also in the results for the overhead measures, that reflect the efficiency of the algorithms, we can observe similar trends. There is a difference, however, between the overhead in number of packets and the overhead in number of bytes. When considering the number of packets, we obtain the same order as before, with AntHocNet giving the best performance in terms of efficiency, followed by AODV, ANSI and OLSR. When considering the number of bytes, the ACO algorithms AntHocNet and ANSI suffer a bit more, with AntHocNet becoming slightly worse than AODV, and ANSI becoming a lot worse than OLSR. This indicates that ANSI and AntHocNet use considerably larger control packets than AODV and OLSR. One reason for this is that ants gather information about the full path that they have followed. In the case of AntHocNet, an obvious other cause of large control packets is the piggybacking of routing information on top of hello messages. As mentioned before in subsection 5.1.5, both the overhead in terms of number of packets and in terms of number of bytes have their importance in AHWMNs. When we compare to the measures of effectiveness, however, the slightly worse results in terms of overhead in number of bytes does not seem to affect the performance of AntHocNet directly.

5.2.2 Varying the pause time for RWP mobility

Here, we vary the pause time of the RWP mobility model, from $0s$ up to $480s$, with as intermediate values 15, 30, 60, 120 and $240s$. The results of the experiments are shown in figure 5.2. Increasing the pause time has two different effects on the general properties of the scenario that are relevant for routing. The first of these is a decrease in node mobility: since nodes stay still for longer periods, they are less mobile, and the network becomes less dynamic. As a consequence, the scenario becomes less difficult. The second effect is a bit less straightforward, and has to do with the distribution of nodes over the network area when the RWP mobility model is used. It has been shown that under RWP, there tends to be a higher node density in the center of the network area than on the edges, especially when pause times are low [29]. To understand this, consider the square network area of figure 5.3, where we follow a single node moving according to RWP. The node starts from a randomly chosen start point (A). It chooses a random destination point (B), moves to it in a straight line according to a random speed, and then pauses for a while. After that, it repeats this same sequence of actions till the end of the simulation (in the figure, the node moves subsequently to C, D, E and F). The initial start point and all subsequent destination points are chosen according to a uniform distribution, and can therefore be anywhere in the network area. However, the straight line between any two random points has a higher probability of going through the center than of visiting edge or corner areas. As a consequence, nodes that are

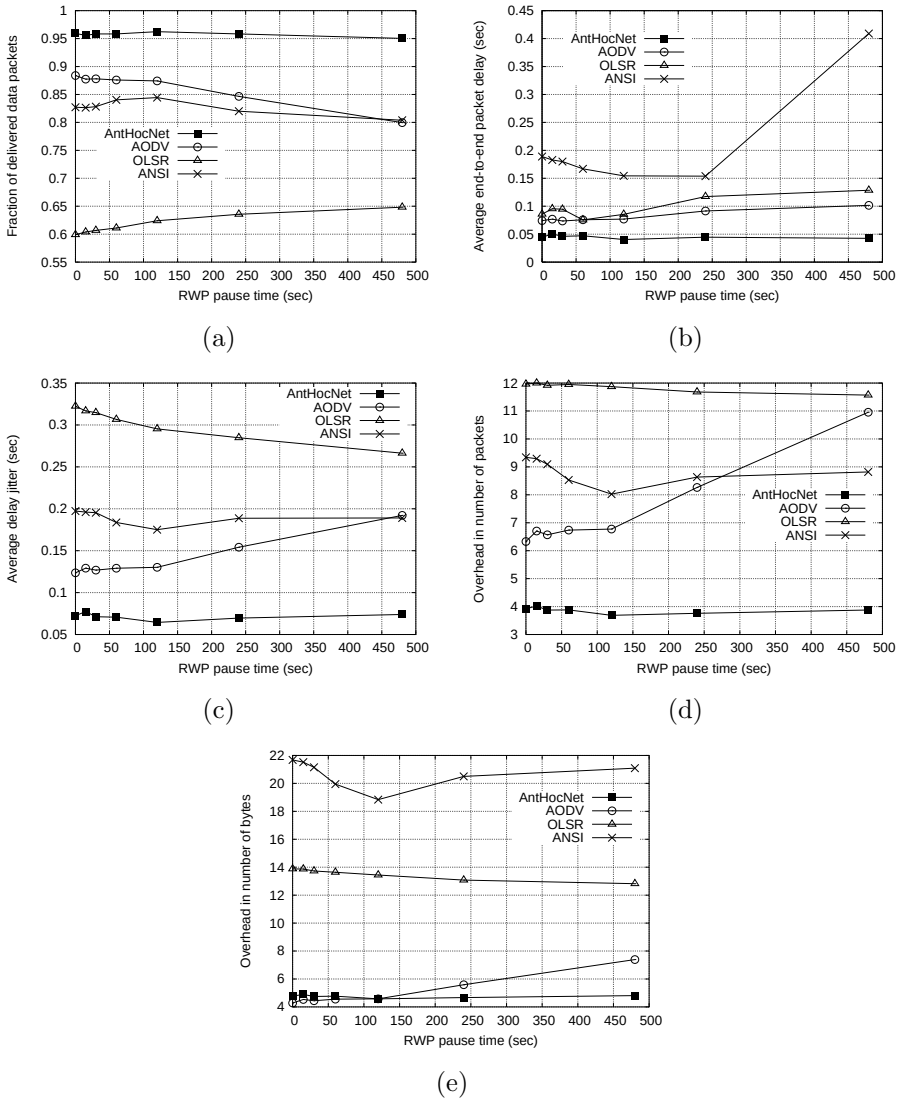


Figure 5.2: Results for AntHocNet, AODV, OLSR and ANSI using different values for the pause time in the RWP mobility model: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

pausing in their destination points are uniformly spread over the network, while nodes that are on the move are more clustered in the center, giving a higher node density there. Concretely, this means that when pause times are increased, nodes are more spread out, giving a lower effective node density to the network.

Earlier, in subsection 2.3.1, we have discussed how a lower node density makes a scenario more difficult to deal with, as the lower connectivity provides less routing alternatives. Hence, increasing the pause time can both decrease and increase the difficulty of the scenario for routing, depending on whether the used routing algorithm is more sensitive to high mobility or to low node density.

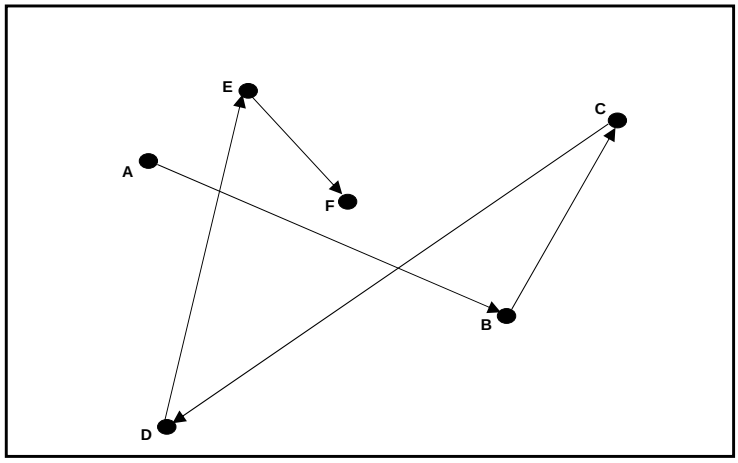


Figure 5.3: A node moving according to the RWP mobility model. The node starts in a randomly chosen initial point (A). It chooses a random destination point (B) and moves to it in a straight line. Then, it pauses for a fixed amount of time. After that, it repeats this sequence of actions till the end of the simulation (leading to the points C, D, E, and F).

When we consider the results for the delivery ratio, the ambiguity in the effect of increasing the pause time can be noted in the difference in performance between the algorithms. AntHocNet shows the best performance, and is rather insensitive to the increase in pause time. ANSI is also quite insensitive, but its level of performance is lower than that of AntHocNet. AODV shows a decrease in delivery ratio for higher pause times: from 88% for the lowest pause time down to 80% for the highest pause time. This is an indication that it is more sensitive to the decrease in connectivity than to the decrease in mobility. Finally, OLSR shows an opposite trend: its delivery ratio increases from 60% for the lowest pause time to almost 65% for the highest pause time.

For delay, the results are similar but slightly different. AntHocNet continues to show good results that are rather insensitive to the variance in pause time, AODV and OLSR show a slight drop in performance, and ANSI a strong one. For the other measures, jitter, overhead in terms of control packets and overhead in terms of bytes, we see the same trends as for delivery ratio: AntHocNet and ANSI show stable results, AODV displays a rather strongly deteriorating performance, and OLSR shows a slight improvement in performance.

We investigate in a bit more detail the difference between AntHocNet and

(a) Number of route setups per session							
Pause time (s)	0	15	30	60	120	240	480
AntHocNet	23.0	24.6	22.1	22.6	20.2	21.4	24.3
AODV	162.5	167.5	164.8	166.3	164.8	183.1	217.7
(b) Number of route retries per session							
Pause time (s)	0	15	30	60	120	240	480
AntHocNet	15.2	16.5	15.0	14.9	13.5	15.0	17.2
AODV	108.1	114.9	111.5	112.7	107.4	127.2	152.4
(c) Number of route repairs per session							
Pause time (s)	0	15	30	60	120	240	480
AntHocNet	24.2	25.9	23.8	24.0	22.2	22.1	21.4
AODV	16.1	17.9	17.5	18.7	21.0	25.1	31.8

Table 5.1: Different control packets used by AntHocNet and AODV in the experiments with increasing RWP pause times. We report the number of route setups, route retries and route repairs per session.

AODV. Here we refer to table 5.1, which reports on different types of control packets used by AntHocNet and AODV. In particular, the table gives the number of route setups, route retries (a new attempt at setting up a route, when an initial attempt has failed before) and route repairs used per session. Of these, the route setups and route retries involve the flooding of a RREQ (in the case of AODV) or a reactive forward ant (in the case of AntHocNet) over the network, and are therefore quite heavy. Route repairs involve a limited flooding and are less heavy. It is striking to see how AODV uses about 8 times as many route setups and route retries than AntHocNet. This is an indication that AntHocNet's strategy of constructing multiple paths proactively pays off. This is how AntHocNet manages to keep the overhead in number of packets low compared to AODV. When we consider the scenarios with high pause time, we see a large increase in the number of control packets needed by AODV, while AntHocNet remains quite stable.

5.2.3 Varying the speed for GM mobility

Here, we present results for tests using the GM mobility model. This is different from all other presented results, where we use the RWP mobility model. The reason for using a different model is that, while RWP is by far the most used model for the generation of node movement patterns in the literature, it has also received some criticism. This criticism concerns a number of different points. A first one is that RWP does not generate uniform node distributions [29]. This has been discussed in detail before in subsection 5.2.2. A second point of criticism is that the average node speed under RWP can be non-stationary and decreasing [287]. This is due to the fact that nodes are usually allowed to choose a random speed between 0 and a given maximum. Nodes that choose a speed that is very close to 0 may be traveling towards their next destination for a time that is longer than the duration of the simulation, and never choose a

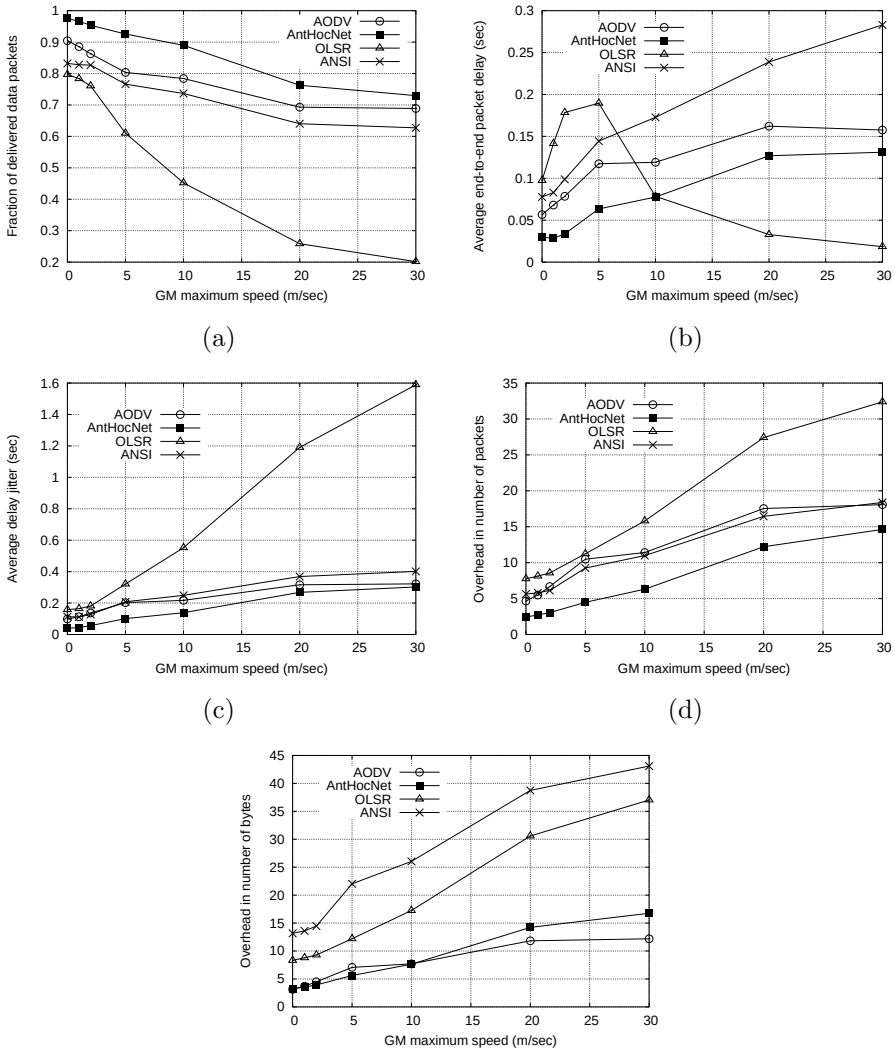


Figure 5.4: Results for AntHocNet, AODV, OLSR and ANSI using different values for the maximum speed using the GM mobility model: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

new random speed: they are stuck in the low speed and bring down the average speed of the nodes in the network. A third point of criticism for RWP is that it does not represent human mobility very well. In particular, it leads to abrupt, uncorrelated movements after each new routing decision.

Average link duration (s)						
Maximums speed (m/s)	1	2	5	10	20	30
RWP	380	269	155	111	60	45
GM	345	207	102	55	29	20

Table 5.2: The average link duration for RWP and GM mobility over a 900 second scenario using increasing maximum speeds.

The GM mobility model was originally proposed to model node movement in infrastructure based wireless networks [172], but has also been applied in AHWMN research [46]. A number of different implementations of the model have been described in the literature. Here, we use the one provided in the BonnMotion mobility pattern generation tool [68]. Each node starts from a randomly chosen initial point in the network area, and moves according to a randomly chosen speed and direction. At fixed time intervals, the speed and direction of all of the nodes in the network are changed. For each node, a new speed value is chosen from a gaussian distribution in which the mean is the node's previous speed value, and the standard deviation is a fixed parameter value. A new direction value is chosen in the same way. Speed values are limited to a certain minimum and maximum value: a value that is chosen outside this allowed range is replaced by the closest value that falls inside the range. When a node moves outside of the network area, its next direction is adapted to be one that brings it back into the area. The GM mobility model offers some solutions to the earlier mentioned problems of RWP mobility: it produces movements that are more smooth than the sudden turns that appear under RWP, and it does not give rise to non-stationary node speeds. Saying something about the node density under GM mobility is difficult, as this has not been investigated in as much detail as for RWP.

In our simulation tests, we have used GM mobility with an update frequency of $2.5s$, a standard deviation for speed of 0.5, and a standard deviation for direction of 0.4. The minimum speed is $0m/s$ and we vary the maximum speed using the same values as in the speed value experiments with RWP of subsection 5.2.1: from $1m/s$ up to $30m/s$, with as intermediate values 2, 5, 10 and $20m/s$. One important difference between GM and RWP mobility is that under GM no pause time is used. This makes GM scenarios in general more mobile. This difference in mobility is illustrated in table 5.2, where we show the average link duration in a $900s$ scenario under both mobility models for the different maximum speed values that we use. The average link duration is the time that elapses on average between the moment a link appears and the moment it disappears again, and has been shown to be a good indicator of the mobility of an AHWMN [229]. The values in the table show that the GM scenarios are consistently more dynamic and therefore more difficult than the RWP scenarios.

The results of our experiments using GM mobility are shown in figure 5.4. As can be expected, they follow more or less the same trends as those of the speed experiments using RWP mobility: in general, AntHocNet has the best

performance, followed by AODV, ANSI and OLSR, and all algorithms show a decreasing performance as the maximum speed increases. The fact that node mobility is higher under GM compared to RWP is clearly visible: for all measures and all algorithms, the results are worse under GM than under RWP. This is especially true for the highest speed values, where, according to table 5.2, the relative difference in link duration between RWP and GM is largest. We can also see that in terms of delivery ratio, the difference in performance between AntHocNet and AODV first increases with increasing node speed, and then decreases again. The initial increase confirms what we observed earlier in the tests with RWP, namely that AntHocNet is better able to deal with the growing number of changes in the network. The eventual decrease in the difference between AntHocNet and AODV shows that there is a limit to the adaptivity of AntHocNet. At the highest levels of mobility, the proactive mechanisms of AntHocNet get more difficulties keeping up with the changes in the network, and are less able to make a difference. As a consequence, we can also see a decreasing advantage of AntHocNet in terms of jitter and an increasing advantage of AODV in terms of overhead in number of bytes. So, for very high levels of mobility, AntHocNet keeps performing well, but looses a bit of its advantage over competing algorithms. Finally, we note that the advantage of OLSR in terms of delay for the highest speed values is even stronger here than in the RWP experiments. Again, however, this good delay is only obtained when less than 50% of all packets are delivered and has therefore little relevance.

5.2.4 Varying the data send rate

With the experiments described here and in the next subsection, we investigate the effect of the data load on the performance of the different routing algorithms. In the base scenario, 20 data sessions each sending 4 packets per second are run between randomly chosen start and destination nodes. Here, we investigate the effect of varying the data send rate, while in the next subsection, we investigates what happens when the number of sessions is changed. We do tests sending 1, 4, 8, 10 and 12.5 packets per second, or 1 packet every 1, 0.25, 0.125, 0.1 and 0.08 seconds. Increasing the data rate has as an effect that the network load gets higher, so that congestion and interference become more likely. The results for the tests with varying data rates are presented in figure 5.5.

In terms of delivery ratio, the effect of the increasing congestion is obvious: all algorithms have a monotonously decreasing performance. AntHocNet performs better than the competing algorithms, but also suffers starting from 8 packets per second, where the delivery ratio is down to 66%. It is remarkable to see how AODV's performance drops very suddenly: from 88% at 4 packets per second down to just 40% at 8 packets per second. To understand this behavior, it is important to realize that an AHWMN is a highly non-linear system where the interaction between different mechanisms can have dramatic consequences. In the current experiments, the increase of the data load augments the congestion, which leads to packet loss. AODV interprets this packet loss as an indication of a link failure, and reacts to it with a route repair or a new

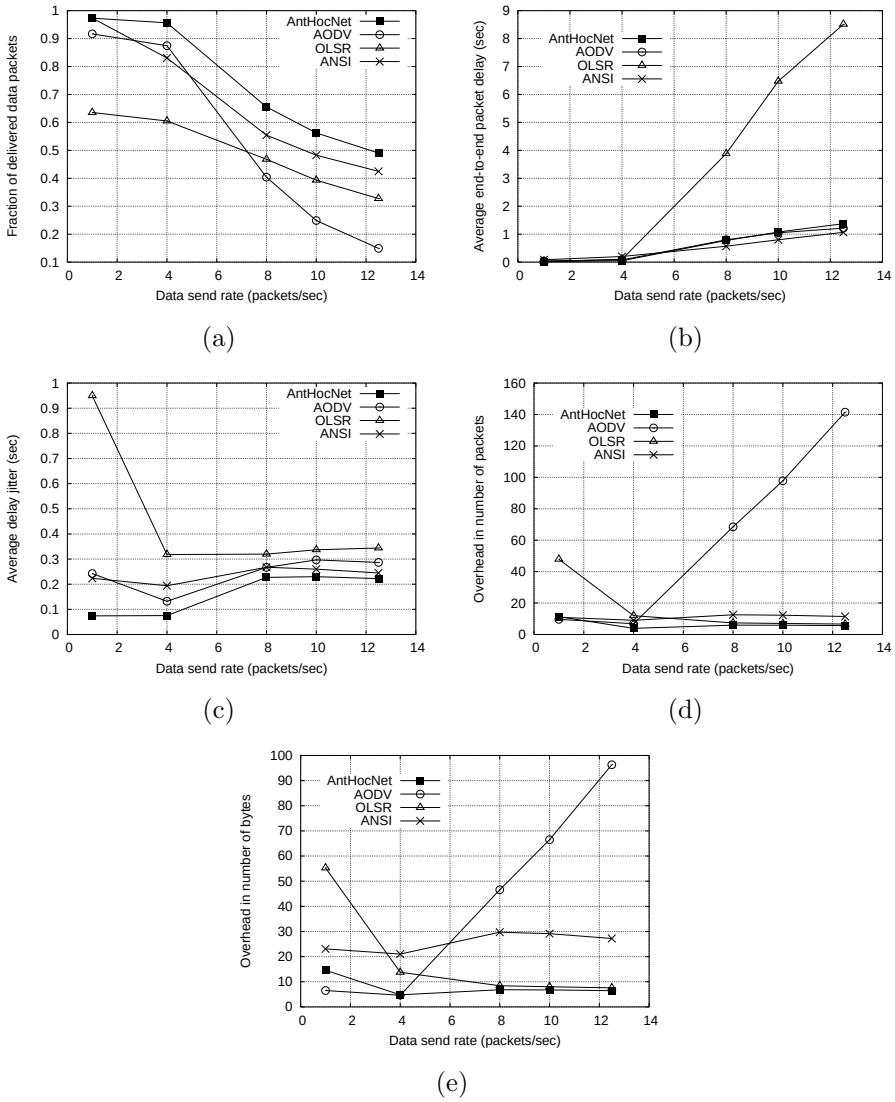


Figure 5.5: Results for AntHocNet, AODV, OLSR and ANSI using different values for the data send rate: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

route setup. This reaction in turn strongly increases the load in the network (the flooding of a RREQ creates a large amount of extra overhead) and makes the situation worse. Hybrid algorithms such as AntHocNet and ANSI suffer much less from such problems because they do not rely purely on reactive mechanisms to deal with events; e.g. AntHocNet's proactive route maintenance process makes

multiple routes available, which can serve as backup and help to avoid the need to execute a route setup process (see also the earlier presented table 5.1, where we show the difference in number of route setup processes used by AntHocNet and AODV). Finally, proactive algorithms such as OLSR are even less sensitive to this kind of interactions, as they normally do not react to events.

For the delay measure, we can see the same trends as for the delivery ratio: all four algorithms have decreasing performance (increasing delay). AntHocNet has the best performance for the lowest data rates, but is outperformed by ANSI starting from 8 packets per second. For those data rates, however, the delivery ratios for both algorithms are quite low. Compared to the delivery ratio results, AODV shows less problematic behavior here, in the sense that for the few packets that it manages to deliver, it gets a reasonably low delay. OLSR, on the other hand, suffers strongly from the increase in data load.

In terms of jitter, we see a slightly different picture. Between 1 and 4 packets per second, all four algorithms improve to some extent their performance. This is because at 1 packet per second, the network changes a lot between every pair of subsequent data packets, so that there are wide variations in delay, leading to higher jitter. At 4 packets per second, subsequent data packets have more probability of being able to follow the same path and encountering the same network conditions. They experience more similar delays, so that the performance becomes more stable and a better jitter can be obtained. Once above 4 packets per second, the effect of the higher congestion can be felt, and all four algorithms experience a drop in performance. Of all algorithms, AntHocNet is best able to provide a low jitter.

For the remaining two measures, overhead in number of packets and overhead in number of bytes, the results bear resemblance to those for jitter. Also here, the fact that at high data rates subsequent packets are sent closer after each other has a positive effect on the performance. This is because the information gathered by the routing algorithms can get used for more packets before it gets out of date, so that the routing algorithm can work in a more efficient way. This effect is especially visible for the OLSR algorithm, that has a monotonously improving performance. This is because OLSR works in a proactive way and does not react to changes in the data rate: the amount of control packets or bytes it generates is quite stable, and increasing the data rate just means that more data packets are available to be delivered, so that the denominator of the overhead measures increases. Also the other algorithms profit to some extent from the possibility to work more efficiently when data packets are sent at a higher rate: AODV, ANSI and AntHocNet all have a decreasing amount of overhead for the lowest data rates. However, for the higher data rates, the effect of the increased congestion becomes stronger. Especially AODV suffers a lot: while it has the lowest overhead both in terms of packets and bytes for the lowest data rate, it increases rapidly and is outperformed by all other algorithms for higher data rates. The reason for this has been explained before, when we commented on the results for delivery ratio: AODV's purely reactive nature makes it extra sensitive to the arrival of disruptive events. For the highest data rates, AntHocNet has the lowest overhead.

5.2.5 Varying the number of data sessions

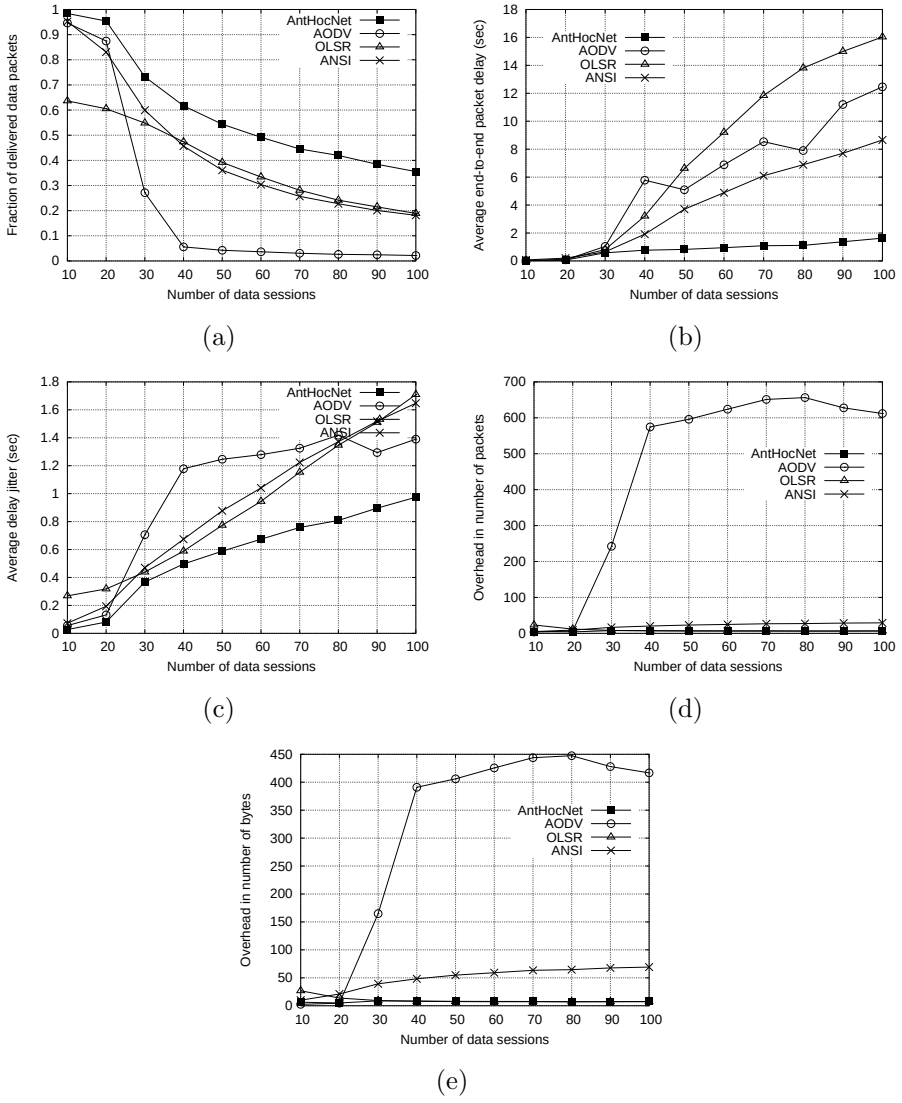


Figure 5.6: Results for AntHocNet, AODV, OLSR and ANSI using different values for the number of data sessions: (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

Here, we present results of tests with a varying number of data sessions: we use from 10 up to 100 sessions, with an increment of 10 each time. Each data session sends 4 packets per second, and source and destination nodes are

chosen randomly. When increasing the number of sessions, we increment the total data load in the network, just like we did when increasing the data send rate in the experiments of subsection 5.2.4. In particular, in terms of number of generated data packets, the scenarios with 40 sessions presented here are equivalent to those with 8 packets per second in the experiments of subsection 5.2.4, and the scenarios with 50 sessions to those with 10 packets per second (other approximate points of comparison between both sets of experiments are at 10 sessions, which corresponds to sending 2 packets per second, and at 60 sessions, which corresponds to sending 12 packets per second). Nevertheless, increasing the data load by augmenting the number of sessions can have a different effect than increasing it by sending at a higher rate. This is because the increased data load is spread over multiple sessions. When extra actions need to be performed on a per-session basis, as is the case for reactive routing algorithms, increasing the number of sessions can be more challenging. The results of the experiments with the number of sessions are shown in figure 5.8.

When considering the delivery ratio, we see a picture that is very similar to that for the data rate experiments: all four algorithms have a decreasing performance for increasing data load, AntHocNet shows the best results, and AODV has a more sudden drop in performance than the other algorithms. When comparing results directly, we can see that for AODV they are even lower here than in the data rate experiments (at 40 sessions, AODV delivers only 5% of all packets, compared to 25% when sending 8 packets per second in the data rate experiments). Apparently the higher number of route setups needed due to the presence of more sessions makes the algorithm even more sensitive to the increase in data load and congestion. Also ANSI suffers a bit more than in the data rate experiments: its delivery ratio even drops below that of OLSR when 40 or more data sessions are used. For AntHocNet, the difference with the data rate experiments is much smaller. Apparently its lower need for route setups (see table 5.1) protects its performances sufficiently. Finally, for OLSR there is practically no difference with the data rate experiments. This is because the algorithm works in a purely proactive way, so that increasing the number of sessions does not provoke higher overhead than increasing the data rate.

In terms of delay, we can notice the same trends as for the delivery ratio. Also here, all four algorithms show decreasing performance, and AntHocNet has the best results. Moreover, like for the delivery ratio, the results for OLSR and AntHocNet are very similar to those obtained in the experiments with increasing data rates. For AODV and ANSI on the other hand, the performance is worse than in the data rate experiments, with a larger difference for the purely reactive AODV algorithm and a smaller difference for the hybrid ANSI algorithm.

In terms of jitter, all four algorithms have decreasing performance, with AntHocNet showing the best results. The overall trends are considerably different from those obtained in the tests with increasing data rates, where all algorithms first showed an improvement in performance, and then a slow deterioration. In section 5.2.4, we explained that the performance improvement was due to the fact that at higher data rates, subsequent packets come closer after each other and therefore experience more similar conditions, leading to lower variations in

interarrival times. When increasing the number of sessions, this positive effect is not present.

Considering the last two measures, the overhead in number of packets and in number of bytes, we can see that the performance of OLSR improves with the number of sessions, while that of AntHocNet and ANSI is relatively stable and that of AODV shows a dramatic deterioration. Overall, AntHocNet shows the best performance. The improvement in overhead results of OLSR is again due to the fact that it is a proactive algorithm and therefore does not use extra control packets when more sessions are started; therefore, more data packets are delivered correctly while using the same number of control packets. Different from the data rate experiments, the other three algorithms do not show an improvement in overhead at the low end of the range of the data load. In the data rate experiments, such an improvement was possible because the higher send frequency of data packets allowed to use reactively obtained routing information for more data packets, thus improving efficiency. When increasing the number of sessions, this effect does not exist.

5.2.6 Varying the network area size

In this subsection, we present results of tests in which we vary the size of the area in which the nodes move. The sizes we use are $1200 \times 400m^2$, $1500 \times 500m^2$, $1800 \times 600m^2$, $2100 \times 700m^2$, $2400 \times 800m^2$, $2700 \times 900m^2$, $3000 \times 1000m^2$, $3300 \times 1100m^2$, and $3600 \times 1200m^2$. Increasing the size of the network area increases the average path length between nodes and decreases the node density. Both make the scenario more difficult. The importance of the node density in AHWMNs has been discussed before in subsection 2.3.1. Sparser networks form a more difficult environment because they are less well connected. In the best case, this means that there are few routing alternatives between the source and destination nodes of a session so that link failures are difficult to repair. In the worst case, there is just no connectivity, and data packets cannot be sent. The results for the network area tests are shown in figure 5.7.

When considering delivery ratio, we can see that for the scenarios with smallest network area, AntHocNet and AODV perform equally well, while the result is slightly worse for ANSI and a lot worse for OLSR. The similarity in performance between AntHocNet and AODV is to be expected. In a dense scenario on a small surface area, paths are short and many alternatives are available, and it is therefore relatively easy to rebuild routes after a link failure. As a result, the proactive route maintenance and the reactive route repair processes of AntHocNet, which are aimed at improving routes and avoiding the need for new route setups, create extra overhead without adding much value. Moreover, the large hello messages sent out by all nodes in AntHocNet can cause more interference than in sparse scenarios, since each node has many neighbors. For increasing area sizes, the scenarios become more sparse, and all algorithms show decreasing performance. AntHocNet is better able to deal with the difficulties of sparse scenarios than the other three algorithms, because here its different mechanisms do pay off. Especially with AODV there is a growing difference in

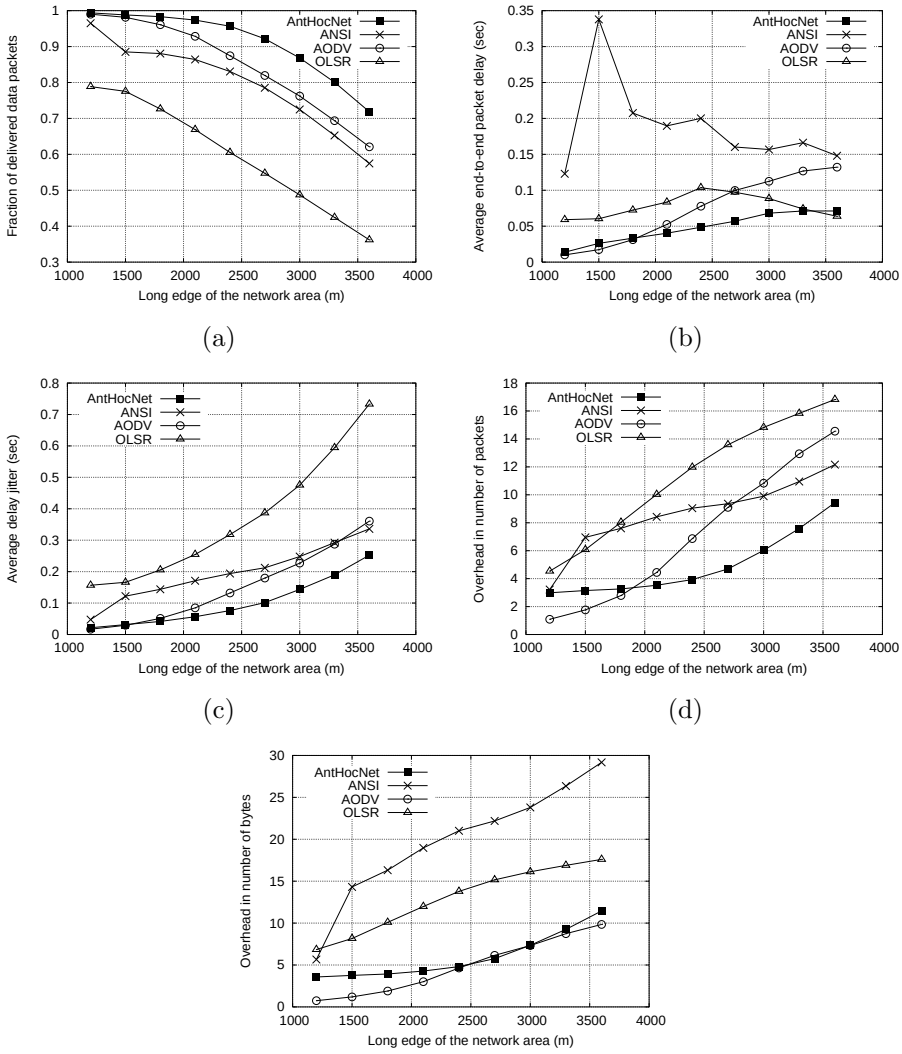


Figure 5.7: Results for AntHocNet, AODV, OLSR and ANSI using different sizes for the network area surface. The length of the long edge in meters is given on the x-axis, while the length of the short edge is always one third of this. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

performance.

When considering delay, the image is similar. Now, for the smallest network areas, AODV performs slightly better than AntHocNet. But again, as the area

increases in size, AntHocNet becomes the better algorithm. For OLSR, the results are a bit ambiguous, with first an increase of the delay and then a decrease. This behavior is similar to that shown in the speed experiments of subsections 5.2.1 and 5.2.3. Also there the delay gets low for OLSR in the most difficult scenarios, when the delivery ratio is already very low. For ANSI the delay results are quite bad and rather unstable.

Also for jitter, the results are similar to those for delivery ratio. For the smallest area size, AODV has similar or even better performance than AntHocNet. Then, as the size increases, AODV's jitter deteriorates faster, and AntHocNet becomes better. OLSR and ANSI both perform worse than AntHocNet, with OLSR giving the worst performance.

Finally, also for the overhead measures we see the same kind of patterns. Here, the advantage of AODV in the scenarios with smallest area is more pronounced. This confirms what we mentioned earlier, that in the scenarios with high density and short paths AntHocNet's mechanisms produce extra overhead without improving performance. AODV purely reactive approach is then better. However, for the scenarios with larger areas, the overhead in number of bytes is comparable for AODV and AntHocNet, while the overhead in number of packets of AntHocNet is much lower than that of AODV. ANSI and OLSR have worse results for both overhead measures.

5.2.7 Varying the number of nodes

In this subsection we investigate the scalability of our routing algorithm. We present the results of a set of tests with increasing network sizes: we increment the number of nodes from 100 up to 800 nodes in steps of 100. We increment the network area size proportionally, from $2400 \times 800m^2$ for the 100 node network up to $6800 \times 2250m^2$ for the 800 node network (with as intermediate steps $3400 \times 1130m^2$, $4150 \times 1390m^2$, $4800 \times 1600m^2$, $5370 \times 1790m^2$, $5800 \times 2000m^2$ and $6350 \times 2100m^2$), in order to keep the node density constant. The results of the experiments are shown in figure 5.8.

When we consider the results for delivery ratio, we can see that AntHocNet is able to deliver more packets correctly than the other three algorithms over the wide range of different network sizes. Moreover, the difference in performance grows with increasing network sizes. For the highest network sizes, AntHocNet still delivers more than 70% of all data. AODV and ANSI, on the other hand, fall below 50%. For OLSR, we did not run tests for more than 500 nodes, as the results were too low (and simulation times became very large). These results show that AntHocNet scales well. Its various mechanisms for proactive route maintenance and reactive route repair allow it to deal better with the longer paths in large networks, and help it avoid the need for new route setups. The latter is very important as a route setup involves the flooding of a reactive forward ant to all nodes in the network. The bad results of OLSR confirm that proactive routing is more difficult when the number of nodes gets higher, as it gets impossible to keep correct routing information for all possible destinations in all nodes.

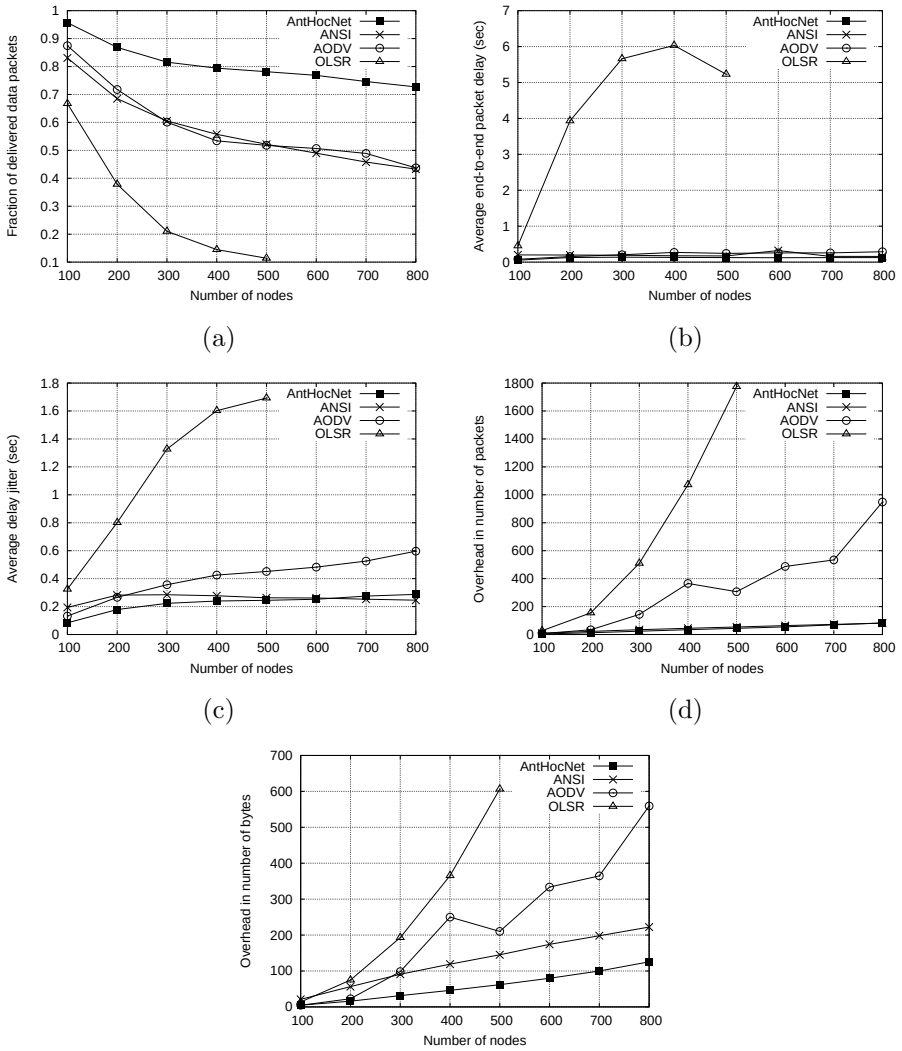


Figure 5.8: Results for AntHocNet, AODV, OLSR and ANSI using different network sizes. The number of nodes in the network is indicated on the x-axis. The network area size is incremented proportionally so that the node density remains the same as in the base scenario. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

When considering delay and jitter, the results are slightly different, but similar. For delay, AntHocNet shows the best results. However, ANSI is now not

much worse. For jitter, ANSI is slightly better than AntHocNet for the largest networks. For both measures, AODV lags more behind, and the difference grows with increasing networks sizes. OLSR, finally, performs really badly. Its delay goes down slightly for the largest networks, where it is only delivering the easiest data packets (a similar effect was also visible in the speed experiments of subsections 5.2.1 and 5.2.3 and the density experiments of subsection 5.2.6).

In terms of both overhead measures, finally, we again see similar results. AntHocNet and ANSI have the best performance, with a small difference when considering number of packets, and a larger one in the advantage of AntHocNet when considering number of bytes. AODV has worse performance, and the difference grows with increasing network sizes, indicating that it is less scalable than the two ACO routing algorithms. Finally, the proactive OLSR algorithm turns out to be highly inefficient for large network sizes.

5.2.8 Summary

In the results presented in this section, we have compared AntHocNet to a number of representative routing algorithms. We have varied many different environmental parameters, in order to investigate how each of these affects the performance of the algorithms in absolute and relative terms. In general, we could observe that AntHocNet shows very good behavior over the wide range of scenarios, and often outperforms the other three algorithms.

When considering the mobility experiments, we can see that AntHocNet can deal better than the other algorithms with increasing mobility. It is better able to deal with the network changes induced by mobility. In the RWP tests with increasing maximum node speed, we can see that as the network gets more dynamic, AntHocNet's advantage over the other routing algorithms grows. In the pause time tests, results are rather ambiguous, due to the various conflicting trends that are caused by changes in the pause time. In the test with increasing speed under GM mobility, we can see similar trends as under RWP mobility, but we can also see that there is a limit to AntHocNet's adaptivity: for the highest speed values AntHocNet's advantage over AODV becomes smaller. This is because mobility under GM gets higher than under RWP, and under the extreme mobility of these scenarios, it becomes hard for AntHocNet's proactive mechanisms to keep up and make a difference.

When considering the data load experiments, we can see that none of the considered algorithms are really able to deal with high data send rates or high numbers of sessions. The AHWMN capacity is just too limited. Nevertheless, we can see that AntHocNet keeps better up with the increasingly challenging environment. Only for the delay results in the data rate experiments, it is slightly worse than ANSI. So we can say that the performance of AntHocNet scales well with increasing data load. On the other hand, the purely reactive approach of AODV turns out to be quite sensitive to changes in the data load, since it creates too much overhead in reaction to disruptive events.

When considering the experiments with varying node density, we can observe that all algorithms suffer from the longer path lengths and lower connectivity as

scenarios get sparser. However, AntHocNet deals better with these challenges, and especially compared to AODV there is a growing gap in performance as node density decreases. On the downside, we can observe that AntHocNet has more difficulties in the densest scenarios. Its proactive route maintenance and reactive route repair mechanisms are rather useless there, as in those scenarios reactively rebuilding a route like AODV does can be more efficient than continuously trying to extend, improve and repair routes. This is most visible in the overhead results, where AODV clearly outperforms AntHocNet when the network is small and dense.

Finally, when we consider the experiments with increasing network sizes, we see similar patterns. Again, all algorithms suffer from the increasing scale, and especially OLSR turns out to be unable to cope with large AHWMNs. In terms of delivery ratio, we see again the same trend as before, with an increasing performance gap between AntHocNet on the one hand and AODV and ANSI on the other hand, showing that AntHocNet is better able to deal with the difficulties that arise in larger networks. In terms of the other performance measures, the same growing performance gap between AntHocNet and AODV remains visible, but ANSI's performance is more similar to that of AntHocNet. In general, the results of the tests with increasing network sizes show that AntHocNet is able to maintain its good performance as the network size increases, thereby showing its good scalability.

5.3 Analysis of AntHocNet's internal working

In this section, we present a number of tests in which we try to get a better understanding of the working of AntHocNet. To this aim, we make variations in the parameters and components used by the algorithm and observe the effect of these changes. In particular, we do tests switching off the proactive components and the local repair mechanism, using different routing metrics, varying the send frequency of proactive forward ants, varying the number of entries in the pheromone diffusion messages, varying the routing exponent of proactive forward ants, and varying the routing exponent of data packets. All tests are again carried out in the earlier described base scenario and adaptations of it. We use adaptations that are relevant for the analysis at hand. We use as evaluation measures the delivery ratio, end-to-end-delay, delay jitter and overhead in number of packets. For some of the experiments, we also include the number of hops taken by successfully delivered data packets. This is a measure of efficiency, as it indicates how many transmissions were needed to bring each of the data packets to its destination. The overhead in number of bytes was not included here, due to general similarity with the results for the other overhead measure.

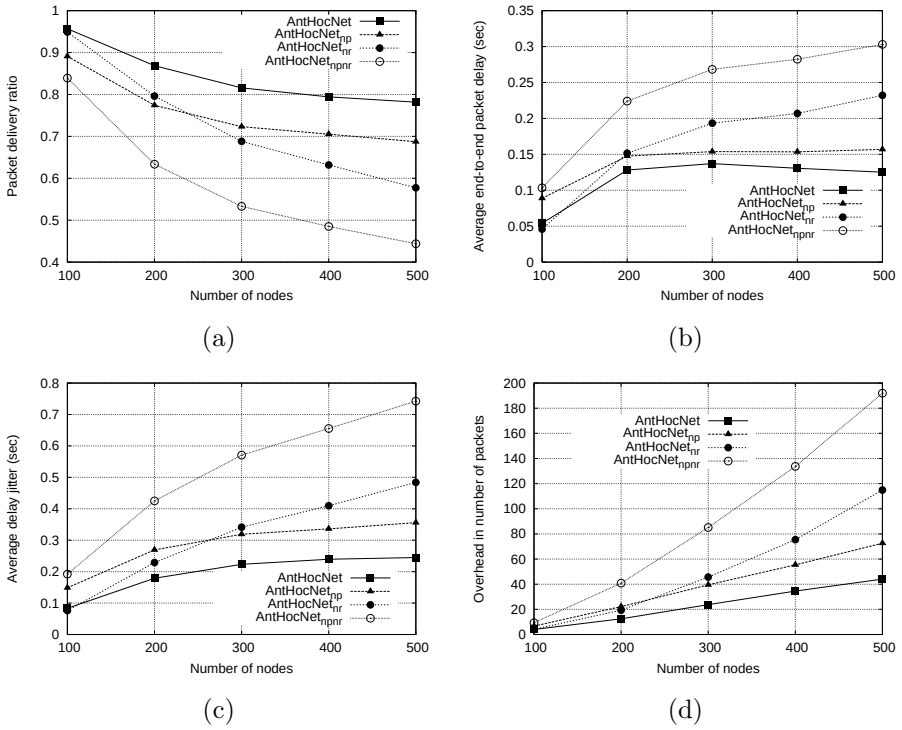


Figure 5.9: Results for AntHocNet, AntHocNet without proactive route maintenance process ("AntHocNet$_{np}$"), AntHocNet without local route repair ("AntHocNet$_{nr}$") and AntHocNet with neither proactive route maintenance nor route repair ("AntHocNet$_{npnr}$"). The tests are carried out in scenarios with increasing number of nodes, as in subsection 5.2.7. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

5.3.1 Switching off proactive actions and local repair

In the experiments presented in this subsection, we try to figure out what the individual effect is of different components of AntHocNet. In particular, we investigate the relevance of the proactive route maintenance process and the route repair process, as these are two components that we found to be defining for the algorithm's behavior. We compare the performance of the full AntHocNet algorithm with the performance of the algorithm without proactive route maintenance (which we refer to as AntHocNet$_{np}$), the algorithm without local route repair (which we refer to as AntHocNet$_{nr}$), and the algorithm with neither proactive route maintenance nor local route repair (which we refer to as AntHocNet$_{npnr}$). The tests scenarios that we use are the ones of subsection 5.2.7, where we increase the number of nodes and the network area simultaneously. The maximum number of nodes here is 500. The results are

presented in figure 5.9.

When we first consider the smallest network size (100 nodes), we can see that AntHocNet$_{nr}$ performs equally well as, or even better than, the full AntHocNet algorithm. This shows that the local repair component adds little or no value at this scale. On the other hand, the proactive route maintenance process does add a lot of value: AntHocNet$_{np}$ performs considerably worse than the full AntHocNet algorithm for all evaluation measures. It is also interesting to see that proactive route maintenance and local repair can substitute each other up to a certain extent. This can be concluded from the fact that AntHocNet$_{npnr}$ performs considerably worse than AntHocNet$_{np}$, while AntHocNet$_{nr}$ does not perform worse than the full AntHocNet algorithm: it seems that the local repair mechanism has more value in AntHocNet$_{np}$, where there is no proactive route maintenance, than in the full AntHocNet algorithm.

When we consider larger network sizes, we can see that the performance gap between the full AntHocNet algorithm and AntHocNet$_{np}$ grows steadily for all evaluation measures, indicating the continued importance of the proactive route maintenance component. On the other hand, the gap between AntHocNet and AntHocNet$_{nr}$ grows fast, indicating that in large network sizes, the local repair mechanism does become an important mechanism in order to maintain good performance.

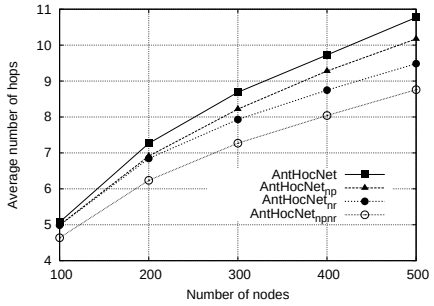


Figure 5.10: The average number of hops for different versions of AntHocNet in scenarios with increasing number of nodes.

Finally, we also present results for the average number of hops taken by successfully delivered data packets. The number of hops is an indication of how efficiently algorithms manage to bring data packets to their destination. The results are shown in figure 5.10. It is striking to see that the relative performances for this measure of efficiency go directly against the performances for all other measures. The full AntHocNet algorithm, which has the best results for delivery ratio, delay, jitter and overhead, uses the longest paths to deliver its data. On the other hand, AntHocNet$_{npnr}$, which has the worst results for the other measures, uses the shortest paths. An explanation for the shorter path lengths used by AntHocNet$_{npnr}$ is that due to the lack of proactive maintenance or repair, routes have to be rebuild from scratch after each link failure. When

building a new route, a reactive forward ant is flooded over the network, and the first copy of it to reach the destination is sent back to the source. This approach assures that a new short route is set up each time. On the other hand, when routes are repaired, or replaced by backup routes, there is no possibility to restart from scratch, so that longer routes are often used. Nevertheless, it is clear from the results that using shorter paths does not necessarily lead to good results. This has also been described earlier in subsection 2.4.3, and we come back to this issue in the next subsection, when we discuss the use of different metrics.

5.3.2 Using different routing metrics

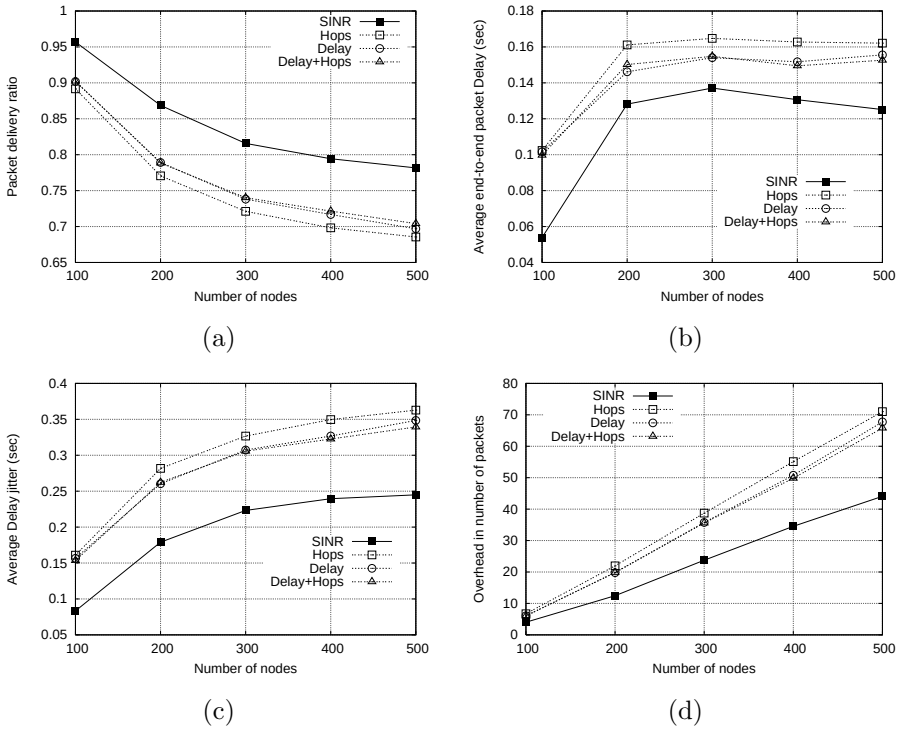


Figure 5.11: Results for AntHocNet using different routing metrics: signal-to-interference-and-noise ratio ("SINR"), number of hops ("Hops"), delay ("Delay"), and the combination of hops and delay ("Delay+Hops"). The tests are carried out in scenarios with increasing number of nodes, as in subsection 5.2.7. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

In this subsection, we present results of tests in which we use different routing metrics. The routing metric is the criterium used by the algorithm to compare

and choose routes. The different metrics we use here have been described in subsection 4.2.6. They are the number of hops, the end-to-end delay, the combination of hops and delay, and a metric based on the signal-to-interference-and-noise ratio (SINR), which penalizes the use of low quality links. The tests presented here were carried out in networks of increasing sizes, with a maximum of 500 nodes, as before. The results are presented in figure 5.11.

From the presented graphs, we can see that using the metric based on SINR gives by far the best results for all considered evaluation measures. So, it is clearly advantageous to be able to detect bad links and avoid them. The delay metric and the metric that combines delay and number of hops give worse results. The worst results, finally, are obtained when using the number of hops metric. This is despite the fact that this metric leads to the discovery and use of shortest paths, as is indicated in figure 5.12, where we plot the average number of hops used for each successfully delivered data packet. Choosing the shortest paths is a common practice in the AHWMN literature. The reason why this is not a good idea was pointed out in [67]: paths with a low number of hops usually consist of long hops, which can be of low quality and break easily as a consequence of node movement, and tend to go through the center of the AHWMN area, where congestion and wireless channel contention is higher.

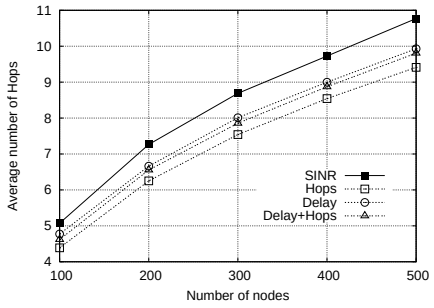


Figure 5.12: The average number of hops for AntHocNet using different routing metrics in scenarios with increasing number of nodes.

5.3.3 Varying the proactive ant send interval

Here, we present results of tests in which we vary the proactive ant send interval. This is the time between the launching of successive proactive ants in the proactive route maintenance process. It defines how often the algorithm looks for path improvements, and therefore how quickly it can adapt to new routing opportunities. We did tests with send intervals of 0.5, 1, 2, 5, 10, 20, and $50s$. We use two groups of scenarios. In the first group, we use variations of the base scenario with increasing mobility: we apply RWP with maximum node speeds of 2, 5, 10 and $20m/s$. We use these scenarios in order to investigate the interaction between the rate of change of the scenario and the adaptivity rate of

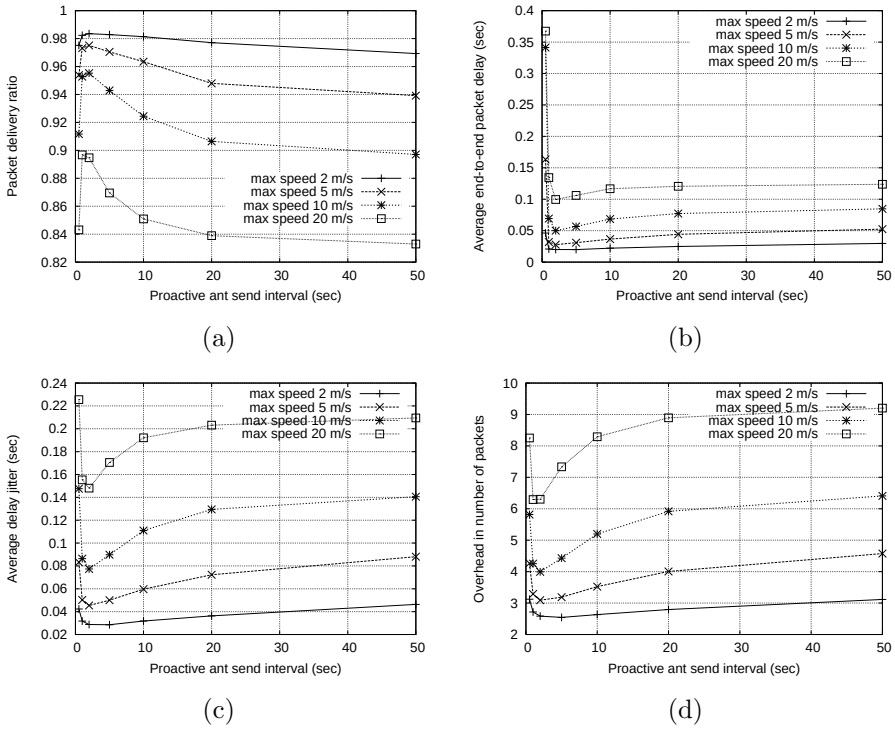


Figure 5.13: Results for AntHocNet using different send intervals for the proactive ants. We send 1 ant every 0.5, 1, 2, 5, 10, 20, and 50s. This is indicated on the x-axis. We use scenarios with varying mobility: we apply RWP with maximum speeds of 2, 5, 10 and 20m/s. The results for different speed values are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

the algorithm. In the second group, we use variations of the base scenario with increasing data send rate: we have data sessions sending at 1, 4 and 8 packets per second. We use these scenarios in order to investigate how the send rate of ants interacts with the send rate of data. The results of the experiments varying the node speed are given in figure 5.13, and those varying the data send rate in figure 5.14.

When considering both figures 5.13 and 5.14, we can observe a constant pattern for all different scenarios and evaluation measures. First, at very low ant send intervals, the algorithm shows bad performance. This is because too many ants get injected into the network, so that they cause congestion. Then, there is an optimum value at around 1 to 2s. After that, the performance decays because the algorithm is not sending enough ants to keep up with the changes in the network. When we focus specifically on the results using varying levels

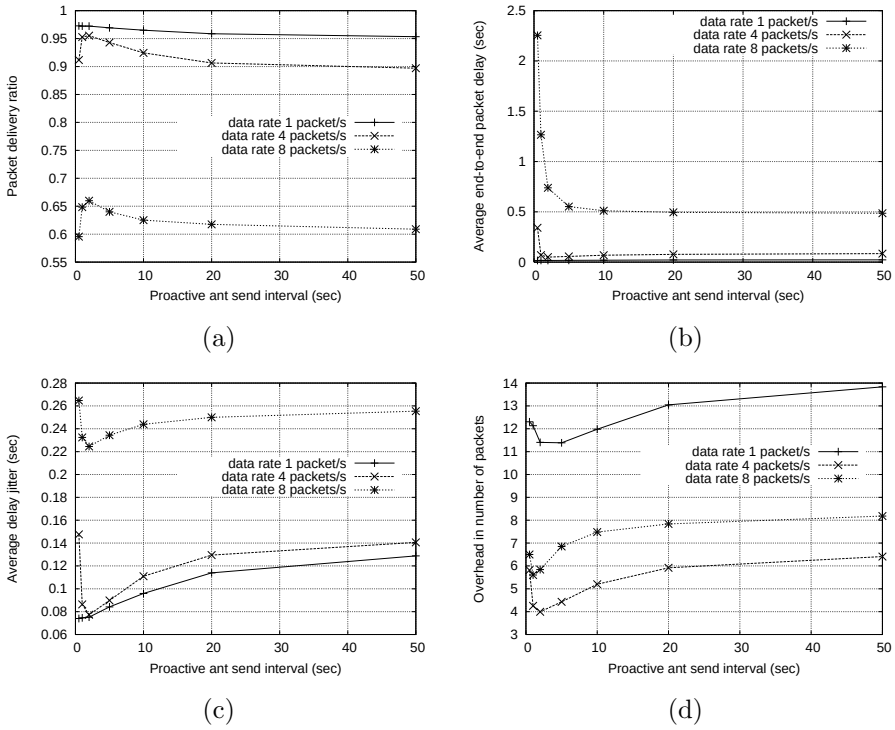


Figure 5.14: Results for AntHocNet using different send intervals for the proactive ants. We send 1 ant every 0.5, 1, 2, 5, 10, 20, and 50s. This is indicated on the x-axis. We use scenarios with varying data load: we use data sessions sending 1, 4 and 8 packets per second. The results for different data send rates are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

of mobility, we can see that the above pattern is less clearly visible for the low speed scenarios than for the high speed ones. This is because when the network changes slowly, it is less crucial to adapt quickly. When we zoom in on the results with varying data load, we can see that the pattern is best visible at the intermediate send rate of 4 data packets per second. When sending less data, paths often need to be rebuilt for each data packet (see also subsection 5.2.4), so that adaptivity is less able to make a difference. For high data rates, the performance generally deteriorates strongly due to high levels of congestion, and again it is more difficult to make a difference using proactive adaptivity. One interesting observation when comparing the results over all different scenarios is that the optimal ant send rate is relatively stable and independent from the node mobility or data send rate. Sending one ant every 2s almost always gives the best performance.

5.3.4 Varying the number of entries in the pheromone diffusion messages

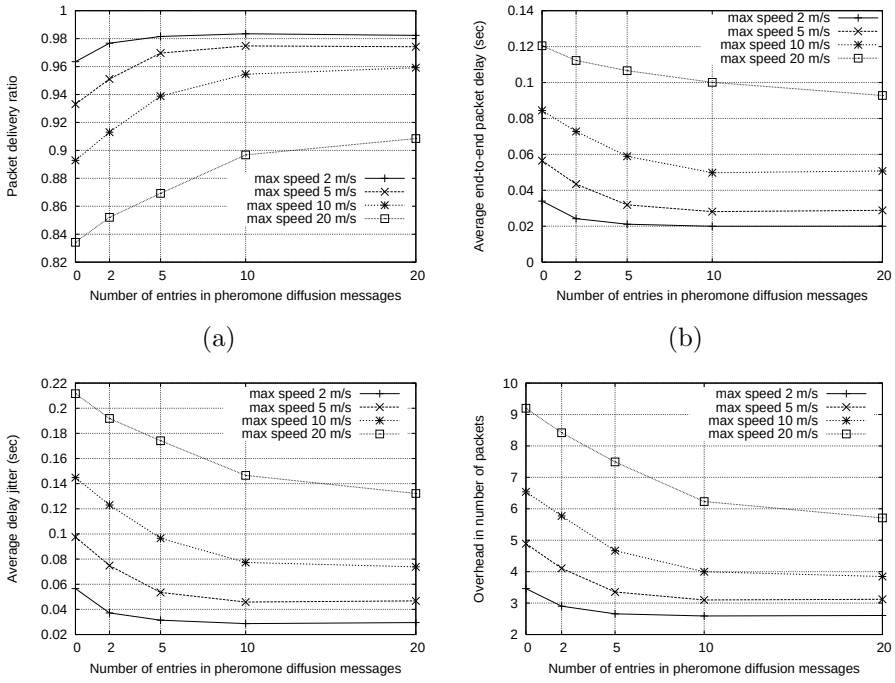


Figure 5.15: Results for AntHocNet using different number of entries in the pheromone diffusion messages. The number of entries used are 0, 2, 5, 10 and 20, and are indicated on the x-axis. We do experiments with varying node mobility: we apply RWP with maximum speeds of 2, 5, 10 and 20m/s. The results for different speed values are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

Here we present the results of tests in which we vary the maximum number of entries used in the pheromone diffusion messages (the hello messages). This number of entries defines how much information is sent out in each of the messages, and therefore how quickly information can spread over the network (see also subsection 4.2.3). We made tests using 0, 2, 5, 10 and 20 entries. 0 entries is the extreme case in which no pheromone diffusion takes place. In that case, no virtual pheromone is available in the network, so that proactive ants cannot find new routes and the proactive route maintenance process is effectively switched off. As scenarios, we again use different levels of mobility, applying RWP with maximum speeds of 2, 5, 10 and 20m/s. The results of our experiments are

shown in figure 5.15.

In general, the results for all evaluation measures show the same trend, with the performance monotonically improving with higher numbers of hello entries. This stresses the importance of having a quickly adapting proactive route maintenance process. Like for the results of subsection 5.3.3, the observed trend is more pronounced when using higher mobility, indicating that the importance of the effectiveness of the proactive adaptivity increases with increasing network change rates.

5.3.5 Varying the routing coefficient for ants

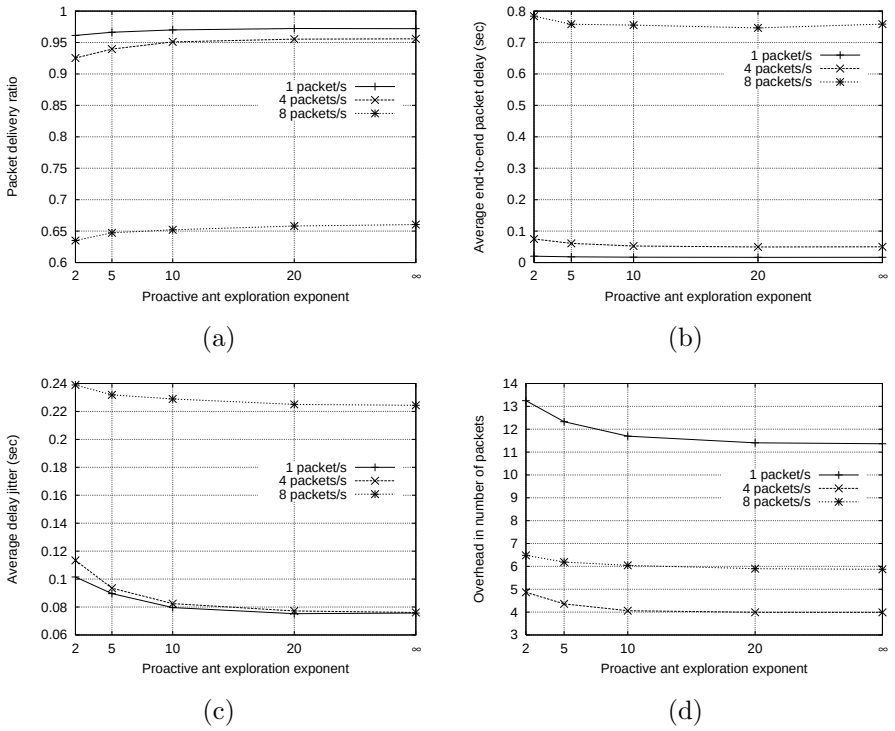


Figure 5.16: Results for AntHocNet using different values for the proactive ant routing exponent. The exponent values that we use are 2, 5, 10, 20 and ∞ (the latter represents deterministic forwarding of proactive forward ants along the best path). The tests are carried out in scenarios with varying data load: we use data sessions sending 1, 4 and 8 packets per second. The results for different data send rates are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

In the experiments presented here, we vary the routing coefficient used by the

proactive forward ants, the parameter β_2 of formula 4.5. This parameter defines the amount of exploration the ants are allowed to do when they are constructing a path towards their destination. When β_2 is high, the ants are concentrated on the paths with the best pheromone values, so that they limit their exploration to paths that have been indicated to be good either by previous ants or by the pheromone diffusion process. On the other hand, when β_2 is low, the ants can also follow paths with low pheromone. This way, paths that are better than what their pheromone values indicate (e.g. because of changes in the network or erroneous previous estimates) can be discovered. A similar parameter β_1 exists for reactive forward ants, and a parameter β_3 for data packets. β_1 is always kept high because when constructing an initial path with reactive forward ants, we want to get a route as quickly as possible and do not want to risk loosing time exploring different possibilities. The effect of β_1 is therefore not investigated. β_3 defines to what extent data packets can be spread over multiple paths and is investigated in the next subsection.

The results of the current experiments are presented in figure 5.16. We use β_2 values of 2, 5, 10 and 20, and also consider the possibility of deterministically following the best pheromone, which corresponds to a β_2 value of infinity. We use scenarios with increasing data load as before, in which sessions send at 1, 4 and 8 packets per second.

From the results, it is evident that the scenarios with 8 packets per second are much more challenging than the ones with 1 and 4 packets per second. Nevertheless, a constant pattern with respect to the β_2 parameter can be observed for all three sets of experiments. As β_2 increases, and the amount of exploration by the ants decreases, the performance improves for all evaluation measures. This is in contrast with other ACO algorithms, such as the AntNet algorithm for wired networks (see subsection 3.2.3 and [71]), where explorative behavior of the forward ants is an essential part of the algorithm. The reason is that in AntHocNet's proactive route maintenance process, the task of exploring new good paths is performed by the pheromone diffusion process (while in AntNet, ants are the only available mechanism to do exploration). This process indicates the good routes it finds through the virtual pheromone, and the role of the ants is mainly to control whether this indicated information is correct. When proactive forward ants are requested to do more exploration, the algorithm is less fast to adopt the best routes indicated by pheromone diffusion, and therefore slower to adapt to new network situations. Therefore, we always use a high value for β_2. In a sense, AntHocNet uses a clear separation of tasks compared to other ACO routing algorithms: the pheromone diffusion process executes exploration and the discovery of new routes, while the ants continuously control the provided routing information and set up routes based on it.

5.3.6 Varying the routing coefficient for data

In this subsection, we present results of tests in which we vary the data routing exponent, parameter β_3 of equation 4.6. This parameter controls the stochastic forwarding of data packets. It defines how strong the preference of data packets

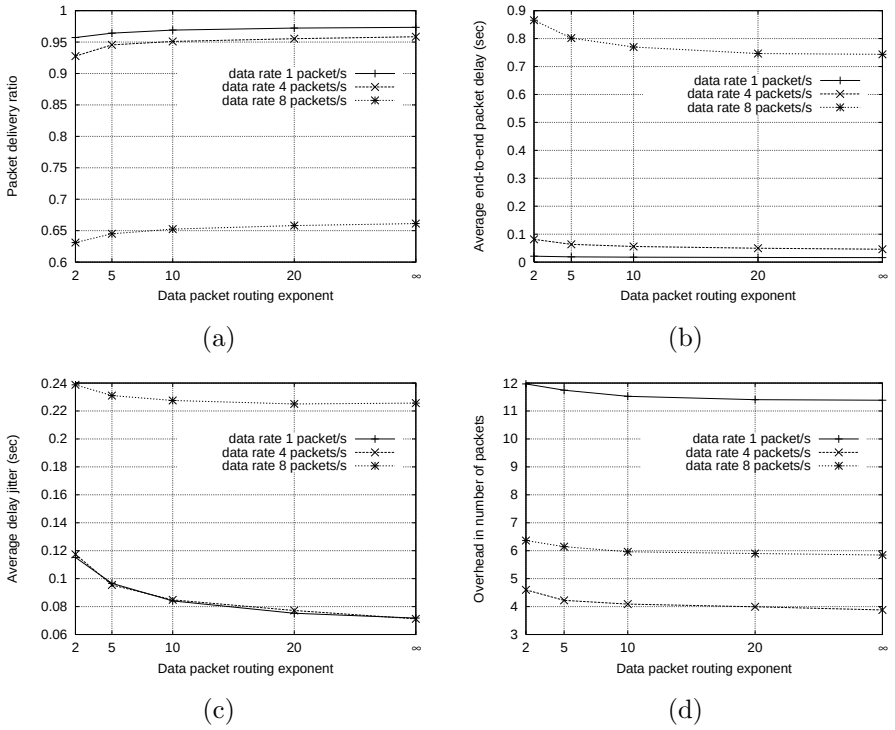


Figure 5.17: Results for AntHocNet using different values for the data routing exponent. The exponent values that we use are 2, 5, 10, 20 and ∞ (the latter represents deterministic forwarding of data along the best path). The tests are carried out in scenarios with varying data load: we use data sessions sending 1, 4 and 8 packets per second. The results for different data send rates are represented with different curves. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, and (d) overhead in number of packets.

for paths with high pheromone is. When β_3 is low, this preference is weak, and data can therefore be spread out over a range of multiple paths. On the other hand, when β_3 is high, data packets are only sent over the best paths. In the limit, where β_3 reaches infinity, data is forwarded deterministically over the best path. In our experiments, we use β_3 values of 2, 5, 10, 20, and infinity. We again use scenarios with increasing data load, in which sessions send at 1, 4 and 8 packets per second. The results are presented in figure 5.17.

The graphs show a similar trend as those for the tests with β_2. Also here, performance improves with increasing values for the routing coefficient under all scenarios and for all evaluation measures. So, it turns out that the feature of stochastically spreading data packets over multiple paths is not beneficial, but, on the contrary, deteriorates results. This is again in contrast with other ACO routing algorithms, where stochastic data forwarding allows to improve

performance by spreading the data load over multiple paths and making better use of the full network resources. An explanation of why this is not working in the case of AntHocNet can be found in the fact that in AHWMNs different paths between source and destination nodes are not very well separated: due to radio interference, data packets traveling over parallel paths can hinder each other, so that it is difficult to obtain any advantages. This issue can possibly be dealt with if specific mechanisms are used during the construction of the different paths, e.g. choosing paths that go over nodes that are outside each other's transmission range (see also subsection 2.4.3). However, such mechanisms would be quite complex, especially if we take into account the fact that nodes move, and paths that were originally sufficiently apart could move into each other's transmission range. In AntHocNet, we did not use such an approach. Instead, we keep the data routing coefficient high, and only use the very best paths. This way, we maximally exploit the best routes available.

5.3.7 Summary

In this section we have presented results of tests in which we switched on and off different components of the AntHocNet routing algorithm and varied its internal parameters. The aim was to investigate AntHocNet's internal working.

First, we have looked at the individual contribution to the algorithm's performance of the proactive route maintenance process and the reactive route repair mechanism. We have shown that the proactive route maintenance process has a strong positive influence on the performance over a range of different scenarios. On the other hand, the reactive route repair mechanism turned out to have a large positive contribution in large networks but very little or no contribution in small networks.

Next, we have investigated the use of different routing metrics. From the results, it was clear that the metric using the SINR was superior. The metric using delay and the one using a combination of delay and hops gave worse results. The worst results were for the number of hops metric, despite the fact that this metric lead to the use of the shortest paths and is very often used for routing in AHWMNs.

Then, we have investigated two parameters that are related to the speed of working of the proactive route maintenance process: we have done tests varying the send rate of proactive forward ants and varying the number of entries in the pheromone diffusion messages. Both tests indicated that a faster working proactive route maintenance process is better: sending ants more regularly and spreading out more information in each message during pheromone diffusion gave better results. Limits are present only when too much overhead is created. For example, sending more ants than one per second lead to rather bad results.

Finally, we have done tests varying the routing coefficient of proactive forward ants and data packets. In both cases, it turned out that increasing the coefficient lead to monotonically improving results. In the case of proactive forward ants, this shows that it is better to leave the task of exploring new paths to the pheromone diffusion process, and let the ants focus on the task of

controlling the obtained information and turn it into routes that can be used for data. In the case of data packets, it shows that it is difficult to get throughput advantage from spreading data over multiple paths, and that instead it is better to fully exploit the best routes found by the ants.

5.4 Conclusion

In this chapter we have provided an evaluation study of the AntHocNet routing algorithm. Tests were carried out in scenarios that are similar to those commonly used in the literature on MANET routing.

In a first set of tests, we compared AntHocNet to existing state-of-the-art routing algorithms. These included AODV, a reference reactive routing algorithm, OLSR, an important proactive routing algorithm, and ANSI, which is a representative of the class of ACO routing algorithms. Results showed that AntHocNet could outperform the other three algorithms over a wide range of different scenarios. Specifically, AntHocNet turned out to perform well in tests with increasing levels of mobility, to deal better than the other algorithms with different levels of data load, to cope well with sparse network situations, and to scale well to networks of increasing sizes. On the downside, AntHocNet's advantage was decreased when mobility got very high, and the algorithm was outperformed by AODV when very dense scenarios were used.

In a second set of tests, we investigated the internal working of AntHocNet. We switched on and off the use of different components of the algorithm, and varied various internal parameters. We found that the proactive maintenance process has a high contribution to the algorithm's performance over a range of scenarios, while the local repair mechanism did not have a positive effect in small networks, but was increasingly valuable in large ones. We also found that it is important that the proactive maintenance process works as fast as possible, as long as it does not produce excessive overhead. Among the possible routing metrics, we discovered that the SINR based metric lead to the best results. Finally, we saw that exploration in the proactive route maintenance process should be left to the pheromone diffusion process rather than to the ants, and that data packets should be forwarded over the best paths, with minimal data load spreading.

Chapter 6

Simulation of an urban scenario

In the previous chapter we have presented the results of a number of tests in which we evaluate the performance of the AntHocNet routing algorithm. All of those tests used scenarios that were derived from a common base scenario, in which a number of nodes move in an open, rectangular area along straight line segments, and data are sent at fixed rates. The reason to use this type of scenarios was to stay in line with the common practice in the literature, so that comparisons with other algorithms are easier and more fair. However, as was pointed out already in subsection 5.1.1, these scenarios are rather simple, and might not give a correct image of situations that occur in reality. Therefore, observed results are not necessarily representative for what can be expected when the network is deployed for a practical application. The aim of the current chapter is to complement the previous tests with new ones that use more realistic scenarios. For these new scenarios, we have chosen an urban environment, as the development of recent projects with public WMNs in some cities such as Philadelphia [12] and Taipei [10] indicate that this will be an important area of application for AHWMN technology. We take special care to simulate this environment and the use of an AHWMN in it in an accurate way. We use this detailed simulation to make new evaluations of AntHocNet's performance.

The rest of this chapter is organized as follows. First, we describe the simulation setup, giving special attention to the way we obtained a model of wireless communication in the urban environment and how we were able to simulate it in an efficient way. Then, we present results of a simulation study, in which we first investigate network properties of the AHWMN in the urban environment, and then perform a comparative test of the AntHocNet and AODV routing algorithms. The work presented here has been described in [97] and in project deliverable [49].

6.1 Design of the simulation study

In this section we describe the design of the simulation study. In what follows we first provide a general overview of the study. Then, we give detailed descriptions of the urban environment and the implementation of node mobility, of the used radio propagation models and their implementation, and of the simulated data traffic. Finally, we review related literature on the topic of simulations of AHWMNs in urban settings.

6.1.1 General description of the simulation study

The aim of the study presented in this chapter is to evaluate the behavior of the AntHocNet routing protocol in an urban environment through simulation. We have chosen the center of the Swiss town of Lugano as the setting of our study. Compared to the open space scenarios of the previous chapter, there are three important differences. The first concerns node mobility. We model urban mobility by limiting the movements of nodes to the streets and open areas in the town, and adjusting their speed to the typical speed of people in a urban environment, be it pedestrians, cyclists or slowly moving cars. We also use a certain percentage of static nodes, to model immobile network users. This makes that the network is not just a MANET, but also has some characteristics of a WMN. Details about the urban environment and the movement of nodes in it will be given in subsection 6.1.2. The second difference concerns the propagation of radio waves in the urban setting. We model the physical propagation of radio waves through the streets of the town using a ray-tracing approach, which accounts for interactions between radio waves and buildings, such as reflection and diffraction [221]. Details about this will be given in subsection 6.1.3. Finally, the third difference concerns the way the network is used at the application layer. Rather than using CBR sessions, we modeled different kinds of data traffic, in order to account for different possible uses of the network. The applications we considered range from an interactive short messaging service (SMS) to voice-over-IP (VoIP) traffic. Details about the modeling of data traffic will be given in subsection 6.1.4.

We run simulations of $500s$ each, and do 20 individual runs per scenario. We normally use AHWMNs of 300 nodes, but also carry out experiments varying the number of nodes from 100 up to 400. Like in the previous chapter we use the QualNet discrete event network simulator (see subsection 5.1.2 and [232]). We made some adaptations to the simulator code in order to be able to simulate the propagation of radio waves in an urban environment in an efficient way, as will be explained in subsection 6.1.3. Also for the selection of networking protocols, we stick with the choices of the previous chapter. At the physical layer we use the IEEE 802.11 protocol sending at a frequency of 2.4GHz and with a bit rate of 2Mbps. At the MAC layer, we use the IEEE 802.11 DCF protocol (see subsection 2.3.3). At the transport layer, we use UDP, due to the before mentioned problems of TCP in AHWMNs (see subsection 2.3.4). Finally, as benchmark routing algorithm to compare and evaluate the performance of

Figure 6.1: The setting of our simulation study: an area of $1561 \times 997m^2$ in the center of the Swiss town of Lugano.

our AntHocNet algorithm, we use AODV. The choice to use AODV is because it gives good results and because it is the most important benchmark algorithm in the research community.

6.1.2 The urban environment and node mobility

The urban setting used in our simulation study is the center of the southern Swiss town of Lugano. Lugano is a relatively small old town presenting an irregular street topology common to most European cities. We focused on an area of $1561 \times 997m^2$, which covers most of downtown Lugano. The street structure is shown in figure 6.1. As shown in the figure, the cityscape is basically composed of streets (the white lanes) and buildings (the gray polygons). Streets define the open spaces where nodes are free to move. Buildings are in our simulation study inaccessible to the nodes and basically play the role of obstacles that put constraints on node movements and shield signal propagation. Other elements are the lake, in the lower bottom of the image, and urban infrastructure such as parking lots and the train station. However, these latter do not play any role and are left in the image for the sole purpose of better showing the town organization.

Node movements were calculated in preprocessing, and fed to QualNet as a mobility trace. The movements were generated according to an adaptation of the RWP mobility model. Under the RWP model, nodes choose a random destination and speed, move in a straight line to the chosen destination at the

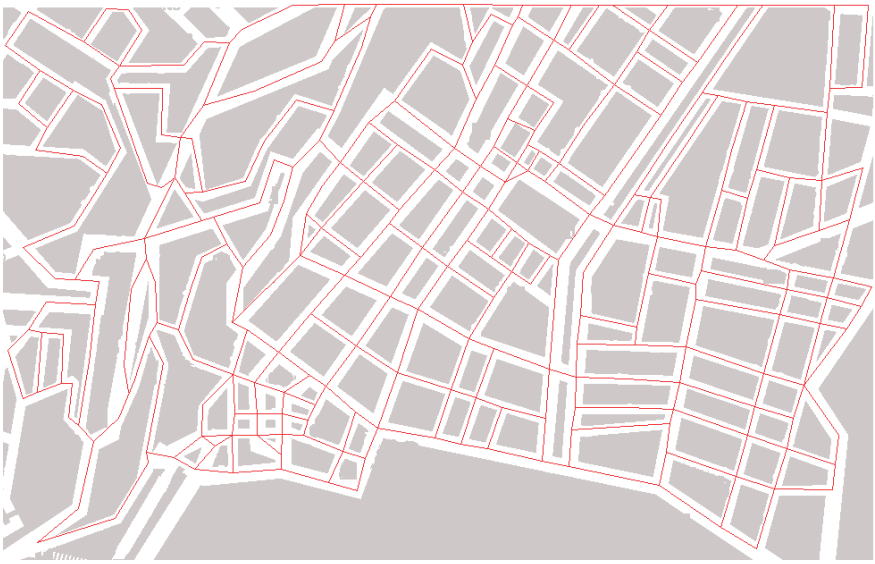


Figure 6.2: A graph representing street patterns in our urban environment. The graph is indicated by the red lines in the figure. This graph was used to calculate locations and movements for the network nodes.

chosen speed, and then pause for a certain time before picking a new destination and speed (about RWP, see also the subsections 2.3.1, 5.2.2 and 5.2.3). In our urban version of RWP, destinations are only chosen from among the open spaces in the town, and nodes do not move along a straight line to their destination, but instead follow the shortest path through the streets of the town. In order to define node destinations and movements, we derived a graph representing the street structure of the town, as shown in figure 6.2. Destinations were chosen from among all points that are located on an edge or in a vertex of the graph, and shortest paths were calculated in the graph using Dijkstra's algorithm [208]. Compared to the real situation in downtown Lugano, our graph contains one extra horizontal road on the northeast side of the area. This is a practical solution in order to avoid that too many streets leading north would have a dead end forcing nodes to turn back to where they came from. On the other sides of the area, this problem was less severe, so that more such modifications were not necessary.

At this point, we want to address some points of criticism that have been raised in the literature concerning RWP (see also subsection 5.2.3). In [287], the authors point out that the average speed of nodes under RWP decreases over time. This is because nodes are usually allowed to choose arbitrarily low speeds: when a node chooses a speed value close to 0 in combination with a destination that is far away, this node might not choose a new speed till the end of the simulation, effectively bringing down the average speed among the

nodes of the network. Following a suggestion by the authors of that paper, we avoid this situation here by keeping the minimum speed for moving nodes at $1m/s$. Another criticism on RWP is that it gives rise to artificial variations in node density, with more nodes being situated in the middle of the simulation area than at the sides (see subsection 5.2.2 and [29]). However, in the scenario considered here, the middle of the area corresponds to the center of the town. Therefore, we do not think that such density differences are unrealistic in the given setup.

In all our experiments, we have chosen maximum node speeds that correspond to realistic inner city movements. In most experiments, we chose the network nodes to be pedestrians or cyclists, with a maximum speed of $3m/s$ ($10.8km/h$). Only for the set of experiments with increased mobility, we allow nodes to go up to $15m/s$ ($54km/h$), which is a reasonable maximum speed for cars in an urban environment. The pause time of our RWP model is always $30s$. Finally, in all experiments, we keep 20% of the nodes static, to represent immobile network users in the town. These can for example be wireless access points placed by shop or restaurant owners, or mesh infrastructure nodes provided by the town authorities. Due to the presence of these static nodes, the networks considered here have more similarities with WMNs than those studied in the experiments of chapter 5, which could be considered pure MANETs.

6.1.3 Radio propagation

Wireless communication in an urban environment is strongly conditioned by the way radio waves interact with the objects they encounter. The most basic effect is that waves produced at street level are blocked by buildings, so that connectivity in urban wireless networks is restricted compared to open space scenarios. Some urban simulation studies for AHWMNs in the literature only account for this effect, using open space propagation models along the line of sight (LoS) and blocking any non-LoS communication (see e.g. [188]). In our study, we use a more detailed approach, which incorporates also other propagation effects. The most important of these effects is reflection off buildings: as radio rays bounce off building walls, they can travel around corners into side streets. Also, reflection allows a signal to travel further along the LoS through a street than it would in open space, since multiple reflected rays are tunneled in the same direction. This means that crude approximation models that do not account for reflection are too restrictive. Another important effect is diffraction, which allows rays to bend around corners to a certain extent. This further improves connectivity into side streets. Other effects include scattering, which is the reflection off small objects and uneven surfaces, and signal variations over time due to changes in the environment, such as the passing of vehicles or people. Both of these last effects are hard to model correctly and greatly increase the computational complexity (see [241]), and are therefore not taken into account in this study.

Making detailed calculations of radio wave propagation in an environment with many obstacles is a computationally intensive task, especially when many

simultaneous transmitters and receivers are involved, as is the case in AHWMN experiments. Therefore, we decided to do all propagation calculations off-line in a preprocessing step. We started from the two-dimensional map of the center of Lugano, and assumed each building on the map to be of a height that is sufficient to block radio communication going over it (a height of 5 meters already makes diffraction over a building impossible [241]). Then, we defined a discretization of the space in which the nodes move, in order to allow an efficient preprocessing: we chose sample points at regular intervals of 5 meters along the streets of the town, resulting in 6070 different positions. The locations of the different sample points are shown in figure 6.3. In each of these points, we placed a transmitter sending at $2.4Ghz$ and with a power of $10mW$, and we calculated with which strength its signal was received in each of the other points. All radio propagation calculations were done using the WinProp tool [21]. This is a commercial software package to model ray propagation in urban environments. WinProp makes use of raytracing, a technique derived from the field of computer graphics in which the trajectories of all the waves going between a transmitter and a receiver are first calculated individually, and then combined in order to derive the resulting signal [238]. The results of the preprocessing calculations were stored in a matrix of 6070×6070 entries. Subsequently, we adapted the radio propagation module of QualNet. The precalculated signal strength values were read into memory. During the simulation, the signal strength between a transmitter a and a receiver b was approximated by the precalculated signal strength between a transmitter in the sample point closest to a and a receiver in the point closest to b. This allows very fast calculations, while only causing an inaccuracy of maximally 2.5 meters on each side.

Our proposed approach allows for an efficient and reasonably accurate simulation of the use of an AHWMN in an urban environment. Nevertheless, we would like to give a word of caution. It is clear that we have made a number of simplifications with respect to reality when we made the calculations of the radio propagation. These influence the correctness of the obtained results. A first simplification is the fact that we used discrete sample points which approximate the location of transmission and reception of radio signals. A second is that we did not model small objects in town such as trees, cars, etc.., and that we assumed all buildings to have smooth, flat walls. Finally, we did not account for altitude differences: while most of the area we considered is flat, the railway station and the streets near it on the west side of the town center (see the map in figure 6.1) are situated on a hill. Due to all of these simplifications, the results obtained in our simulations can be expected to differ to some extent from what could be experienced during a real deployment test in the center of Lugano. However, we want to stress the fact that the greater degree of detail used in these simulations does provide a better approximation of the behavior of an AHWMN in a realistic urban scenario, and that without giving us exact performance results for the Lugano scenario, they do allow us to learn more about the performance of our algorithm in such a setting. Moreover, realistically speaking, no AHWMN simulation can ever be expected to be 100% accurate and efforts to improve accuracy can therefore only be justified up to a

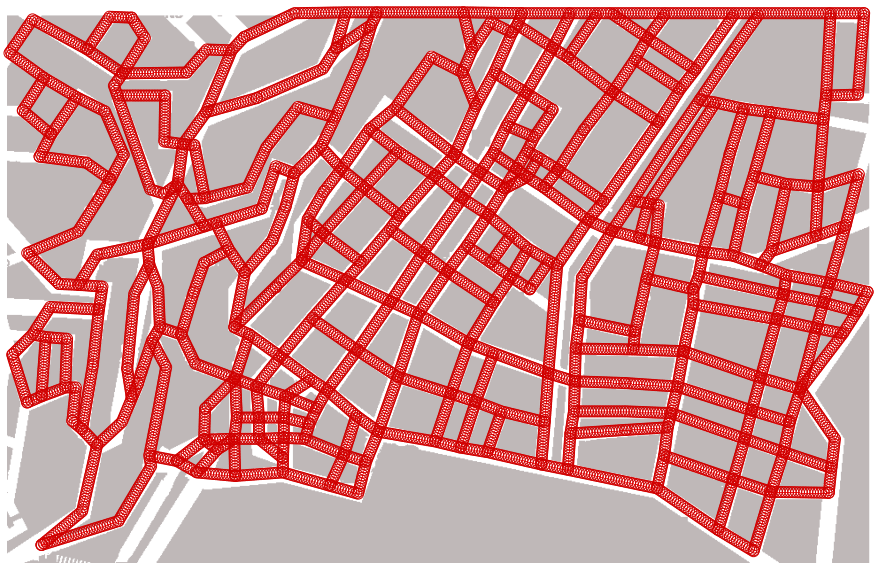


Figure 6.3: For the modeling of radio wave propagation, we make use of precalculated signal strength data. In order to calculate this data, we choose discrete sample points at regular intervals in the areas where the nodes move, and make ray propagation calculations for each pair of points. In the map, the locations of the sample points are indicated with red circles.

certain extent.

6.1.4 Data traffic

When modeling data traffic we use patterns that can reflect realistic applications of the network. We assume that the AHWMN will in the first place be used to support interactive communication between users. We model this type of applications using bidirectional point-to-point data communication sessions. The data packet size is 160 bytes in all tests. The data rate is varied, from 1 packet every 30 seconds, representing an interactive SMS conversation, up to 25 packets per second, which is sufficient to support good quality VoIP applications. In order to represent silent periods in the interactive communication, only 40% of all scheduled packets are sent. This corresponds to the typical proportion of send time in VoIP traffic [139]. Note that here we do not apply detailed models of VoIP traffic, but rather a rough approximation that reflects the data load that can be expected. Developing accurate models of VoIP is a topic of ongoing research in the networking community (see e.g. [79]), and is outside the scope of this thesis.

6.1.5 Related work on the simulation of AHWMNs in urban environments

While there exists a lot of work on the evaluation and comparison of different routing algorithms for AHWMNs (e.g., see [42, 65]), almost all of it is carried out using open space scenarios with random mobility and idealized signal propagation models such as the ones applied in chapter 5. Only recently a number of studies have appeared that investigate the use of AHWMNs in urban environments. Here we provide a short overview of this work.

One could make a distinction between simulation studies that make rather rough approximations of how an urban setting influences node mobility and radio wave propagations, and studies that specifically aim at high accuracy, thereby sacrificing efficiency. The work presented in [138] belongs to the first category. The authors propose a scenario with randomly placed building blocks. Node mobility is modeled in a way that is similar to what we propose in subsection 6.1.2, in the sense that they use an adaptation of RWP whereby node movements are limited to certain paths between the buildings. Radio wave propagation is implemented using a simple LoS approach: nodes can only communicate with each other when no building is situated between them. As mentioned in subsection 6.1.3, such a simple model is too restrictive compared to reality. The authors investigate how the considered setup influences the behavior of AODV in comparison to open space scenarios and notice a severe drop in performance. The authors of [188] follow a similar approach. In their urban environment, buildings are not placed randomly, but aligned according to a regular grid pattern. Nodes move through this grid according to a sort of random walk that is limited to the spaces between the buildings. Radio propagation is simulated with an LoS approach. The authors evaluate how the performance of DSR is influenced by the urban setting and find a drop in performance that is however not as strong as the one observed by the authors of [138] (however, this is partly due to the fact that they specifically set scenario parameters so that the performance of the routing algorithm is not affected too much). Finally, the authors of [132] use a similar grid town pattern, but with a different heuristic radio propagation model. Here, radio signals are weakened with a fixed amount for every corner they take. The aim of the paper is to investigate the feasibility of a commercial MANET application; in particular, the authors investigate whether a MANET could be used to support communication between a fleet of taxis. Node movement patterns in the grid world are based on data about the behavior of real taxis and can therefore be considered more realistic than the random patterns used in the previously described studies. The conclusions from the study are not entirely positive. Especially, the authors point out that a high density of users in the system is a critical factor for good performance.

Among the studies that apply a higher level of accuracy in their simulations, we find in the first place the work presented in [242] and other papers by the same authors. In this work, real and very detailed town maps are used, and radio propagation is modeled using a raytracing approach. The authors apply some discretization of the space, which allows to perform a preprocessing step,

and use a number of optimization techniques to speed up raytracing calculations. Nevertheless, the final simulations take very long (in the order of tens of processor days). Also in terms of the modeling of node mobility, the authors try to develop very accurate models. The movement patterns they use are based on the combination of a number of different models that are derived from different research areas: they use results from the fields of urban planning (which allows to derive how people typically navigate from one location to another), meeting analysis (which allows to define how people move around and cluster together), and time use (which allows to figure out when people typically start and end certain activities). The proposed simulation environment is used to simulate the use of a WMN running AODV in an area of central London. The conclusion is also here that high node density is a critical factor in the system's performance. Another study that uses a very detailed simulation model is the one described in [234]. The authors also use raytracing, and apply a preprocessing step that is based on using a discrete set of transmitter locations but an unlimited number of receiver locations. Their simulations are reasonably efficient: they are only about a factor 1.5 slower than comparable open space simulations when using the ns-2 simulator. The authors perform experiments both with an indoor and an outdoor scenario, using variations of RWP to derive node movement patterns. In both cases, they compare the performance of AODV in the urban scenario to that in an open space scenario, and observe a large drop in performance.

The work presented in the current chapter of this thesis is in approach and level of detail similar to the last described work. It allows to get a reasonable feel for what the effects can be of an urban environment, while making enough abstraction to get an efficient simulation. This allows us to run a number of different tests in order to see how the approach followed by AntHocNet deals with the challenges of the urban setting, and compare this to the different strategy adopted by AODV. None of the urban scenario simulations we found in the literature provides a performance comparison between different routing algorithms.

6.2 Test results

In this section we describe the results of a range of tests that we ran with the simulation setup described above. The aim is to investigate how AntHocNet performs in an urban environment, and to compare it to AODV. In what follows, we first describe in subsection 6.2.1 results of tests in which we investigate general properties of the AHWMN network in the urban scenario, as compared to an open space scenario. These tests are meant to aid in the understanding of the results observed further on. Then, we present results of comparative tests, in which we vary similar parameters as in the tests of subsection 5.2: we present results with varying data send rates in subsection 6.2.2, with varying numbers of data sessions in subsection 6.2.3, with varying node densities in subsection 6.2.4 and with varying maximum node speeds in subsection 6.2.5. Finally, in subsection 6.2.6, we also investigate whether it is possible to support

VoIP level data loads with the different routing algorithms in the given urban scenario.

6.2.1 General network properties

In this subsection, we study how the properties of the network formed between the MANET nodes are affected by the fact that we work in an urban environment. The data shown here were obtained by running simulations with an increasing number of nodes in both the urban scenario and an open space scenario of the same dimensions. In the open space scenario we use the two-ray radio propagation model (see subsection 5.1.3). During these simulations, no data traffic was sent, but all nodes were periodically broadcasting beacon messages. We recorded all receptions of these beacon messages, and constructed a network graph for each beaconing period, counting a link between a pair of nodes i and j if during the period of the beaconing interval i received a beacon from j and j received one from i. This way we got a picture of the network connectivity as experienced by the nodes in the network. In these experiments, all nodes were moving between 1 and $3m/s$. We report results for the average number of neighbors, the connectivity (i.e., the fraction of node pairs between which a path exists), the average length of the shortest path between each pair of nodes, and the average link duration. The results are shown in the graphs of figure 6.4.

The average number of neighbors per node are shown in figure 6.4a. We can see that this number is always a lot lower in the urban scenario than in the open space scenario. This is due to the limited radio propagation caused by the shadowing by the buildings of the town. Also, we can see that, while the number of neighbors grows linearly with the number of nodes in the network in both scenarios, the increase is steeper in the open space scenario. This means that for an equal number of nodes per square meter, nodes in the urban scenario locally experience a lower density. As a consequence, connectivity is worse in the urban scenario, but interference between nodes is also lower.

The lower connectivity in the urban scenario is confirmed by the results of figure 6.4b, where we report the fraction of node pairs between which a path exists in the AHWMN network topology. While the open space scenario is always fully connected, the urban scenario has limited connectivity when there are few nodes in the network. It is interesting to note that the connectivity in urban conditions seems to saturate at a value that is lower than 100%. So even with 400 nodes, where the number of neighbors per node is higher than in the open scenario with 100 nodes, some nodes manage to stay out of reach in dead areas caused by the irregular structure of the town.

In figure 6.4c, we report the average length of the shortest path (measured in number of hops) between connected node pairs. Also this value is very much affected by the environment: paths are about double as long in the urban scenario. The node density has some influence on path lengths in the urban scenario, but less in the open space scenario since there even 100 nodes are enough to provide almost straight line paths.

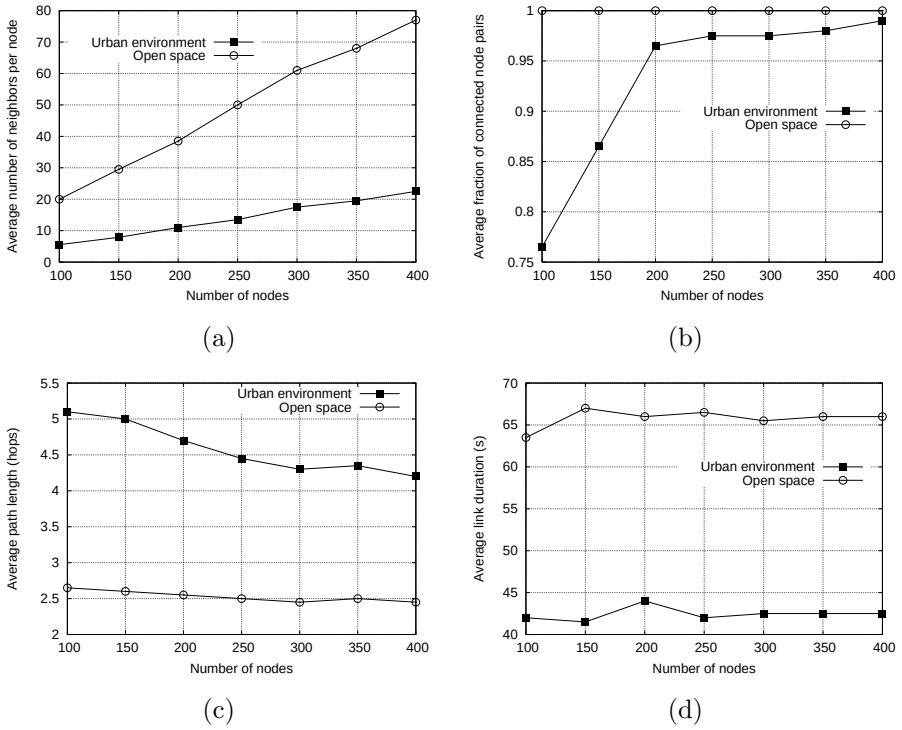


Figure 6.4: Graph properties of AHWMNs with increasing number of nodes in the urban setting versus in an open space environment. We report here (a) the average number of neighbors per node, (b) the average fraction of node pairs between which a path exists, (c) the average length of the shortest path between nodes (measured in number of hops), and (d) the average link duration.

Finally, in figure 6.4d, we report the average link duration. This is the average time that elapses between the appearance and disappearance of a link, and is a measure for how dynamic the network is (see also subsection 5.2.3). We can see that the average link duration is independent of the number of nodes. It is constant on about $65s$ in open space, and $43s$ in the urban scenario. This means that the change rate of the network is higher in the urban environment. We also did tests increasing the maximum node speed from $3m/s$ to $10m/s$ (not indicated in the figure), which gave an average link duration of $56s$ in open space, and $28s$ in the urban scenario.

6.2.2 Data send rate

In this subsection, we compare the performance of AntHocNet and AODV in the urban scenario with 300 nodes and with increasing data send rate. We use 10 parallel bidirectional data sessions of $0.033 packets/s$ (1 packet every 30

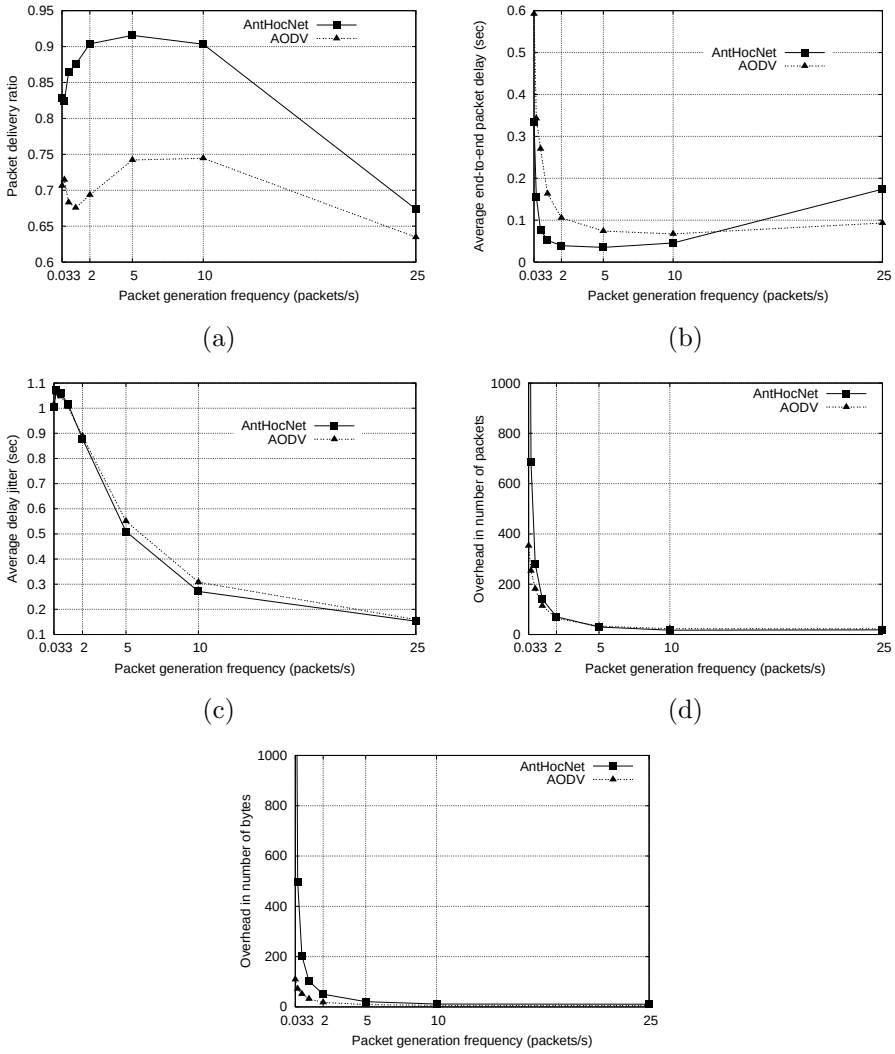


Figure 6.5: Results for AntHocNet and AODV with increasing data send rates in the urban scenario. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

seconds, corresponding to slow interactive SMS data load) up to $25 packets/s$ (corresponding to the data load needed to support good quality VoIP communication data load), using as intermediate values 0.2, 0.5, 1, 2, 5 and $10 packets/s$. Like in the comparative tests of section 5.2, we report the delivery ratio, the average end-to-end delay, the average delay jitter, the overhead ratio in number

of packets, and the overhead ratio in number of bytes. All results are shown in figure 6.5.

When considering delivery ratio and delay (figure 6.5a and 6.5b), we can see that both algorithms have bad performance for the lowest data rates, better performance at intermediate rates, and worse performance again for the highest rates. The bad performance at the lowest rates is due to the fact that both AntHocNet and AODV need to set up a route between source and destination prior to communication. When data packets are sent sporadically, previously constructed routes can rarely be reused, and new route setups are often necessary. As data rates increase, subsequent packets can profit from previous route setups. In AntHocNet, where routes are proactively maintained and therefore remain valid for longer, this effect is visible at lower rates than for AODV. For the highest rates, both algorithms have a decrease in performance, because the high load of data packets starts to interfere with the control packets. In general, we can see that AntHocNet outperforms AODV for most data rates (except for the delay at the highest data rate). Considering that the urban scenario has longer paths and less good connectivity than a comparable open space scenario, this result was to be expected, as it was shown in subsections 5.2.6 and 5.2.7 that AntHocNet performs well under such conditions. When we compare directly with the results for varying data send rates in open space scenarios presented in subsection 5.2.4, we can see that both algorithms are better able to deliver the highest data rates in the urban scenario (especially for AODV the difference is large). This is on the one hand because we use only 10 sessions here (see also further in subsection 6.2.3), and on the other hand because there is less interference due to the shadowing by the buildings. Nevertheless, it must be noted that neither of the algorithms is able to provide a delivery ratio that is sufficient to support a VoIP application (see also further in subsection 6.2.6).

When considering average delay jitter, we can see more or less monotonically decreasing values for both algorithms, with a slight advantage for AntHocNet. The fact that jitter improves with increasing data rates is due to the fact that as data packets are sent more frequently, subsequent packets travel closer after each other, and therefore encounter more similar conditions. Hence, the differences in their delays are smaller. A similar effect was visible in the comparative tests in the open space scenarios of subsection 5.2.4. However, in those tests, jitter values only decreased for the lowest values for the data rate, and then started to increase. It must be pointed out that in absolute terms, the jitter values in the city scenario are much higher for the lowest data rates (due to the worse connectivity and the longer path lengths), and that therefore more improvement is possible.

In terms of both overhead measures, we can observe monotonically improving performances for both algorithms. This is because of the earlier mentioned effect that at low data rates, routes practically need to be rebuilt for each data packet, whereas at higher rates, subsequent packets can profit from earlier route setups. Especially AntHocNet produces high overhead for the lowest data rates. This is because its mechanisms to proactively maintain routes create a lot of overhead compared to the number of packets that are delivered. For the higher data rates,

both algorithms have similar overhead ratios. AntHocNet slightly outperforms AODV in overhead in number of packets, but not in overhead in number of bytes.

6.2.3 Number of data sessions

Here, we report the results of test in which we vary the number of data sessions, from 5 up to 25 sessions, with as intermediate values 10, 15 and 20 sessions. All data sessions are bidirectional. We did tests with two different values for the data send rate: 5 and $10 packets/s$. In figure 6.6, we report delivery ratio, delay, jitter, overhead ratio in number of packets and overhead ratio in number of bytes.

When considering delivery ratio (figure 6.6a) we can see that AntHocNet outperforms AODV for all scenarios. Apart from that, we observe a common trend for both algorithms under both data rates: performance monotonically deteriorates when we increase the number of sessions. At the higher data rate of $10 packets/s$, this deterioration is more clear than at the lower rate of $5 packets/s$: at $10 packets/s$, AntHocNet's delivery ratio goes down from 93% for 5 sessions to 57% for 25 sessions, and AODV's delivery ratio from 77% for 5 sessions to 43% for 25 sessions. The larger difficulties experienced by both algorithms when more data sessions and high data rates are used are in line with the open space results of subsections 5.2.4 and 5.2.5. They confirm that the ability of both algorithms to deal better with high data rates in the urban scenario, as was observed in subsection 6.2.2, was at least in part due to the fact that we used less sessions.

When considering average end-to-end delay (figure 6.6b) and average delay jitter (figure 6.6c), we observe similar trends as for the delivery ratio: AntHocNet consistently outperforms AODV, performance for both algorithms deteriorates monotonically with increasing numbers of sessions, and the deterioration is more pronounced at the higher data rate of $10 packets/s$. The end-to-end delay of AODV shows a particulary bad increase for the highest numbers of sessions. This pattern is similar to the bad results we observed for AODV in the open space scenarios of subsections 5.2.4 and 5.2.5. Also in those tests, AODV had much more difficulties than AntHocNet when dealing with increasing numbers of sessions.

Finally, when looking at both overhead measures (figures 6.6d and 6.6e), we can see that the performance at the low data rate of $5 packets/s$ stays stable for increasing numbers of sessions, while for $10 packets/s$, it deteriorates. Especially AODV shows a huge increase for both overhead measures for the highest number of sessions. This extra overhead is responsible for AODV's increase in average end-to-end delay. It confirms that AODV's purely reactive approach, whereby new route setups are frequently needed, does not scale well with increasing numbers of sessions.

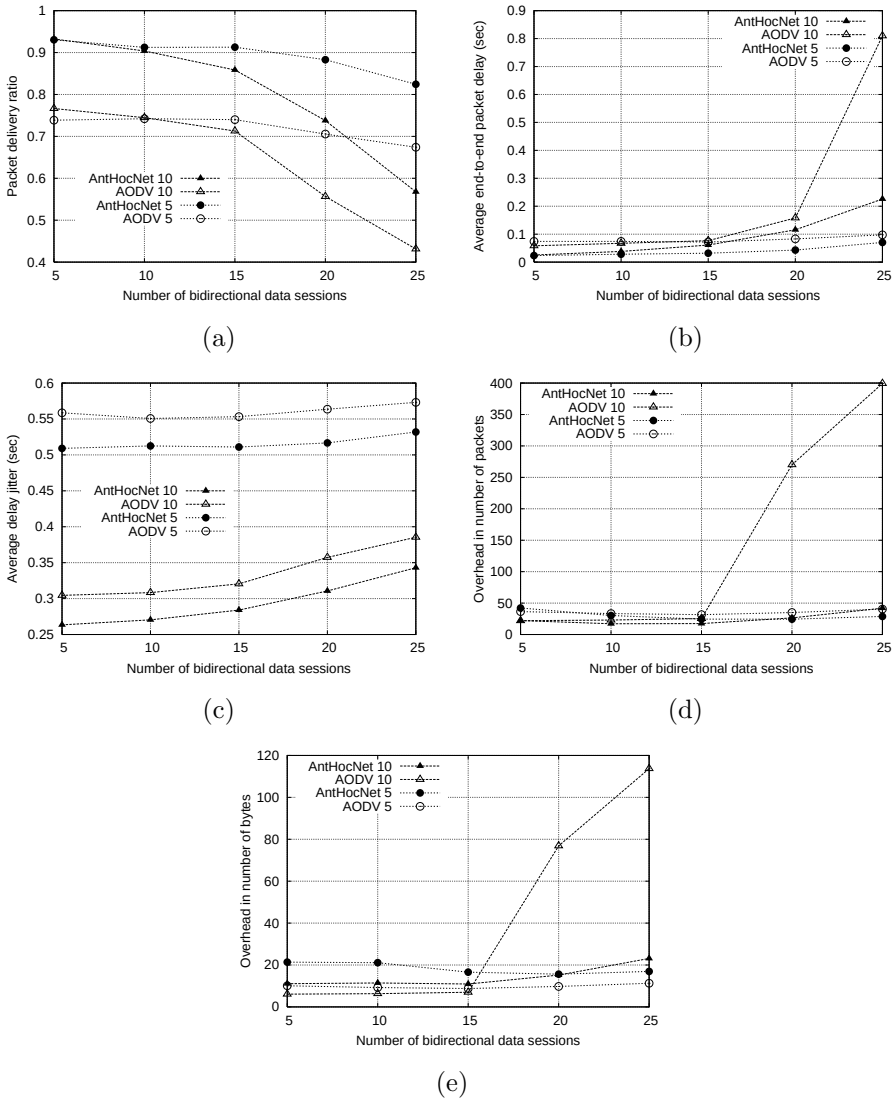


Figure 6.6: Results for AntHocNet and AODV with increasing number of data sessions in the urban scenario. We use data rates of 5 and 10 *packets/s*, indicated by different curves in the figure. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

6.2.4 Node density

In the tests reported on here, we increase the number of nodes from 100 to 400 with increments of 50. This is similar to the tests of subsection 6.2.1. Since

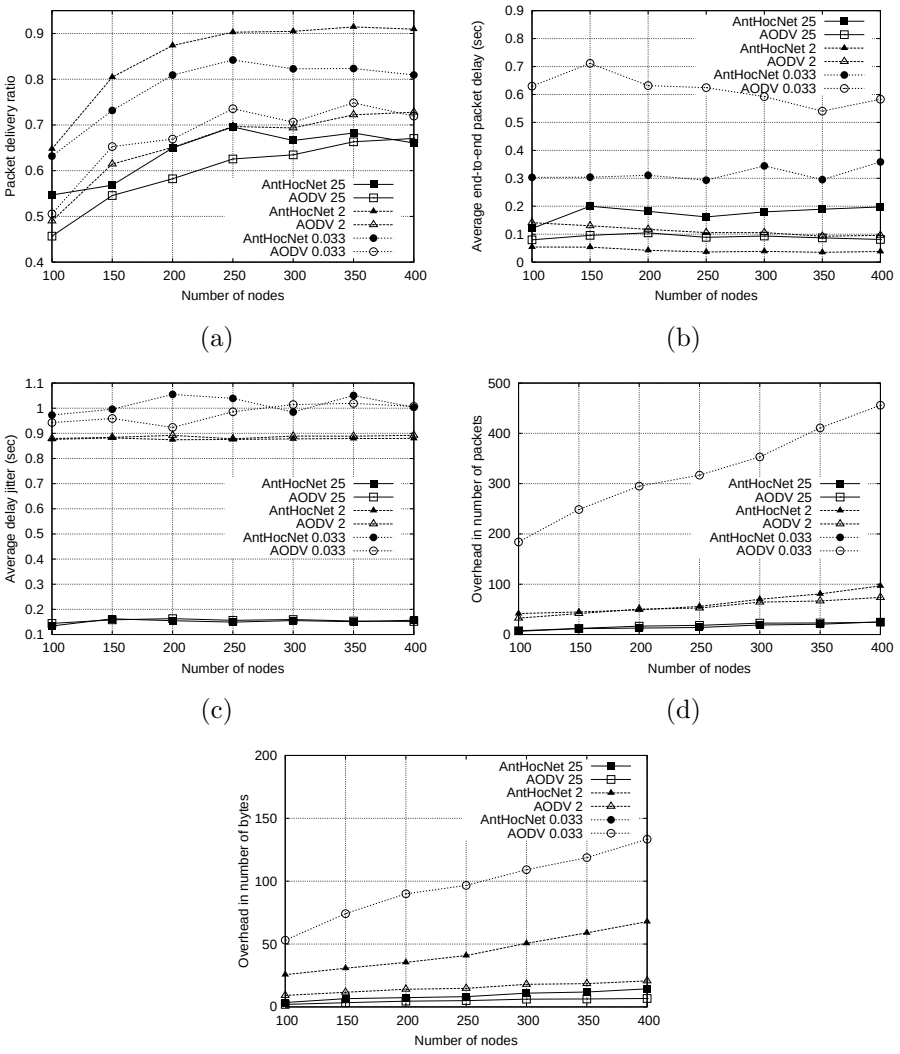


Figure 6.7: Results for AntHocNet and AODV with increasing number of nodes in the urban scenario. We use data rates of 0.033, 2 and 25$packets/s$, indicated by different curves in the figure. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

the network area size of our urban scenario is fixed, increasing the number of nodes comes down to increasing the node density. We used 10 bidirectional data sessions, and did tests with three types of data load: low (0.033$packets/s$),

medium ($2 packets/s$) and high ($25 packets/s$). We report delivery ratio, delay, jitter, overhead ratio in number of packets and overhead ratio in number of bytes. All results are shown in figure 6.7.

In terms of delivery ratio (figure 6.7a), we observe a fixed pattern for both algorithms and for each of the data rates: performance improves with increasing node density, up to 250 nodes, after which it stabilizes. The delivery ratio is best for medium and low data rates, and worse for the highest rate, confirming that it is difficult to support the data rates needed for VoIP traffic. AntHocNet outperforms AODV for all data rates and all node densities, except for the highest data rate at the highest node density. This confirms again that AntHocNet has problems when the network gets too dense. However, the density at which AntHocNet gets in trouble here is much higher than the one in the open space tests of subsection 5.2.6 ($1/7500$ $nodes/m^2$ in open space versus approximately $1/4000$ $nodes/m^2$ in the urban scenario). This is because of the shadowing of radio rays by buildings, which leads to a lower effective node density (see subsection 6.2.1). It is also interesting to see that the trend followed by the delivery ratio graphs is the same as that followed by the graph of the pairwise connectivity in the urban scenario, shown in figure 6.4b: first steeply increasing and then stabilizing. This shows that a relatively high number of nodes is needed in urban scenarios in order to provide the level of connectivity that is needed to deliver a good service.

In terms of end-to-end delay (figure 6.7b), we can see performances that are relatively stable with respect to the node density. Only for the medium data rate ($2 packets/s$), we could say that the delay decreases with increasing node density. AntHocNet outperforms AODV for low and medium data rates. However, for the highest data rate, AODV has better delay. This is different from open space scenarios (see subsection 5.2.4), where AODV had huge difficulties delivering high data rates. As pointed out before in subsection 6.2.3, at least part of the responsibility for AODV's improved performance is the fact that we have less data sessions here.

In terms of delay jitter (figure 6.7c), we can again observe a relatively stable behavior with respect to node density. Moreover, both algorithms have more or less the same performance. In general, we can see that jitter is highest for the lowest data rate, lower for the medium data rate, and lowest for the highest data rate. This is in line with the earlier results of subsection 6.2.2.

Finally, in terms of both overhead measures (figures 6.7d and 6.7e), we can see that performance is worse for the low data rate (the results for AntHocNet for the lowest data rate are off the chart), and better for the medium and high data rates. When looking at the overhead in number of bytes, AntHocNet does always worse than AODV. When looking at overhead in number of packets, AntHocNet does worse than AODV for the lowest data rate, but comparable or better for the medium and high data rates. The bad results for both algorithms at low data rates is due to the earlier mentioned fact that when data packets are sent sporadically, a route setup is almost always needed for each new data packet. AntHocNet's strategy to proactively maintain routes then leads to extra overhead without giving much advantage. At higher data rates, packets can

profit from earlier route setups, so that the overhead ratio can be reduced. AntHocNet's mechanisms to proactively maintain paths augments this effect, so that the algorithm can get a lower overhead ratio when measured in number of packets. However, the fact that AntHocNet uses larger hello messages as well as larger control messages (forward and backward ants carry full paths, whereas AODV's RREQ and RREP messages do not) causes a higher overhead when measured in number of bytes. Finally, we would like to point out that both algorithms experience increasing overhead for increasing numbers of nodes. This is because in networks with more nodes flooded control packets are forwarded more times, and more hello messages are produced.

6.2.5 Node speed

In the experiments presented here, we vary the maximum node speed. We increase it from the $3m/s$ used in the previous tests up to $15m/s$, which is a reasonable maximum speed for cars in an urban environment. As intermediate values we use 6, 10 and $12.5m/s$. We keep using 10 bidirectional data sessions and again do tests with different data rates: a low rate of 0.033 packets per second, a medium rate of 2 packets per second, and a high rate of 25 packets per second. In figure 6.8, we report delivery ratio, end-to-end delay, delay jitter, overhead in number of packets and overhead in number of bytes.

As can be expected, delivery ratios (figure 6.8a) go down with increasing node speeds under all data send rates for both algorithms. Apart from that, we get a confirmation of earlier results. Delivery ratios are highest for medium and low data rates, and lowest for the highest data rate. AntHocNet delivers more packets than AODV, except for the highest data rate, where delivery ratios are comparable.

In terms of delay (figure 6.8b), we get a similar picture. Delay goes up with increasing nodes speeds, although the effect is minimal. The best delays are obtained for the low and medium data rates, and the worst for the highest rates. AntHocNet outperforms AODV at low and medium data rates, while AODV is the best at the highest data rate.

In terms of jitter (figure 6.8c), the effect of the node speed is again minimal: both algorithms give more or less stable jitter for increasing node speeds at all data rates. As in previous tests, jitter is highest for the lowest data rates and lower for the medium and high data rates. The performances of both algorithms in terms of jitter are very comparable.

Finally, also for both overhead measures, we get familiar patterns. For both measures, performances get slightly worse with increasing node speeds. For overhead in number of bytes, AODV outperforms AntHocNet at all data rates (the curve for the lowest data rate for AntHocNet is outside the range of the graph). For overhead in number of packets, AODV outperforms AntHocNet for the low and medium data rates, but not for the highest data rate.

In general, it is surprising to see that the effect of increasing the node speed is rather limited. This is of course partly because we considered only limited ranges of speed. However, using higher speeds would be unrealistic in the given

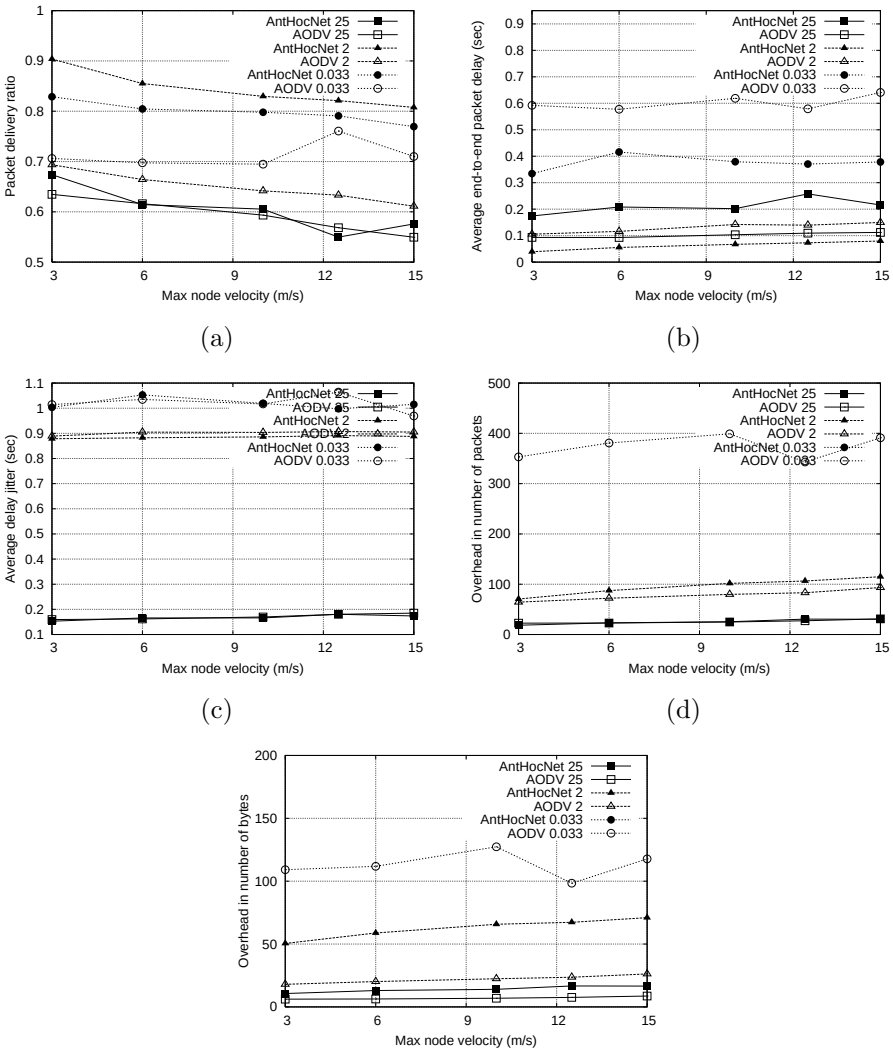


Figure 6.8: Results for AntHocNet and AODV with increasing maximum node speed in the urban scenario. We use data rates of 0.033, 2 and 25$packets/s$, indicated by different curves in the figure. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

urban setting. It appears therefore that in this kind of urban scenarios, the effect of the data send rate, the number of data sessions and the number of nodes in the network is much more important than the effect of the node speed.

6.2.6 Supporting VoIP traffic data loads

In a final set of experiments, we investigate the issue of delivering sessions with VoIP level data rates. In 6.2.2, we saw that both algorithms have trouble when data rates go up to VoIP levels: AntHocNet drops to a delivery ratio of 67% and a delay of $0.19s$, while AODV has a delivery of 63% and a delay of $0.1s$. Jitter is for both algorithms around $0.15s$. To support good quality VoIP, a delivery ratio of 90% is needed, and end-to-end delay should not exceed $0.15s$ (see [189]). Jitter is normally considered in conjunction with delay, as applications deal with jitter by keeping packets in buffer for a short time before delivering them; jitter can therefore be added to delay to count towards the delay limit [189]. Using this approach, we can see that both in terms of delivery ratio and delay, the standards needed to deliver good quality VoIP conversations are not met by either of the algorithms. Here, we investigate this in more detail.

We do tests varying the number of data sessions from 1 to 10. All sessions are bi-directional and send $25 packets/s$. The results are show in figure 6.9. When few sessions are started, we can see that AntHocNet manages to reach the formulated VoIP requirements both in terms of delivery ratio and delay (when delay and jitter are added together). AODV on the other hand falls short on both requirements. Then, when the number of sessions is increased, also AntHocNet fails to reach the VoIP requirements.

These reported results are averages over all the started communication sessions though. In order to get a more precise view, we investigate for each scenario how many of the individual sessions reach the cited requirements. In figure 6.10, we show this number (a) as a fraction of the total number of started sessions, and (b) in absolute terms. We can see that using AntHocNet, at least a few of the sessions can obtain VoIP quality. When only one session is started, it almost always gets the required quality (in 90% of the cases, as shown in figure 6.10a). Then, as more sessions are started, the fraction of them that receive VoIP quality decreases. However, in absolute terms, the total number of sessions that receive the required quality grows up to almost 2.5 when 4 sessions are started, and then remains more or less stable. Finally, when more than 7 sessions are started, the absolute number also decreases. When 10 sessions are started, on average about only 1 receives VoIP quality. This is because too many sessions are interfering with each other. For AODV, the number of sessions receiving the required service quality always remains low.

The results show that using AntHocNet it is in principle possible to support VoIP in the given urban scenario. However, not all sessions can get the required levels of service, and when too many sessions are started, all of them suffer. This indicates that it might be useful to refuse some sessions to start (e.g., sessions that would likely not receive the required quality anyway, because they need paths that go through highly congested or poorly connected parts of the network) in order to be able to deliver a good service to other sessions. This points to the importance of the use of admission control or some other system for Quality-of-Service support in resource limited urban MANETs: if only sessions that are likely to get the levels of service they need are allowed to start, less

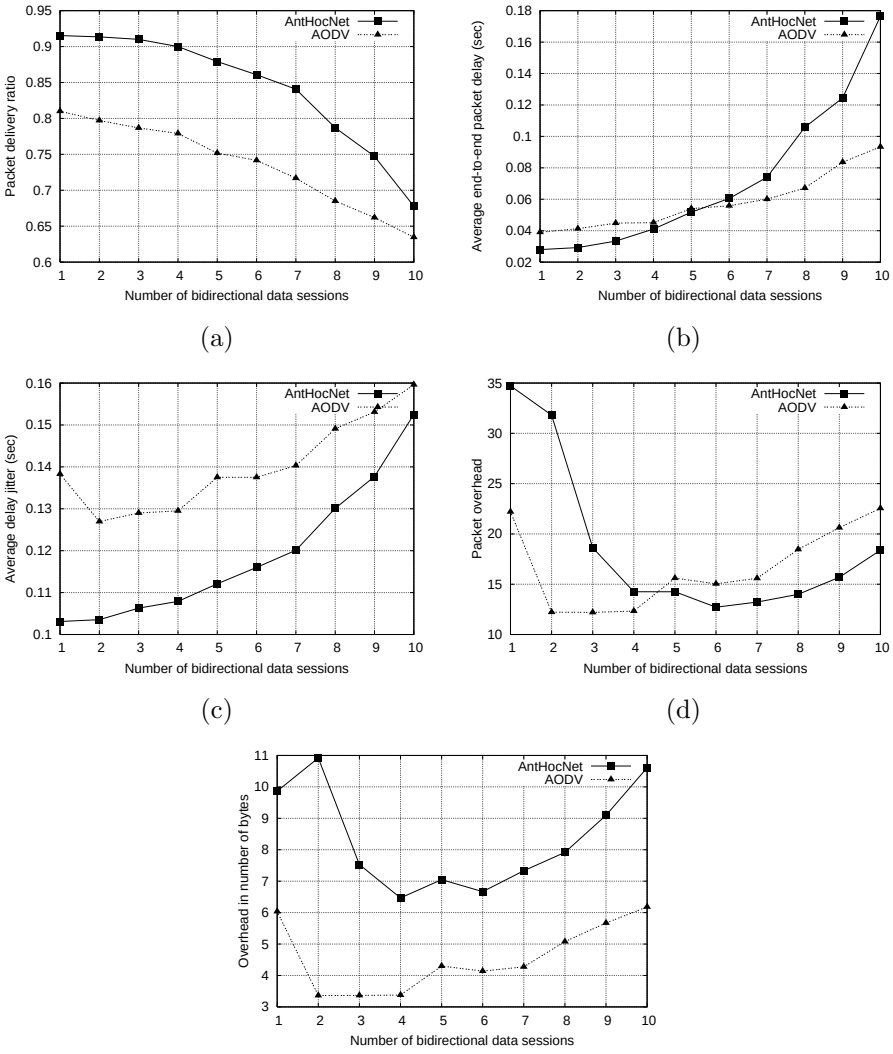
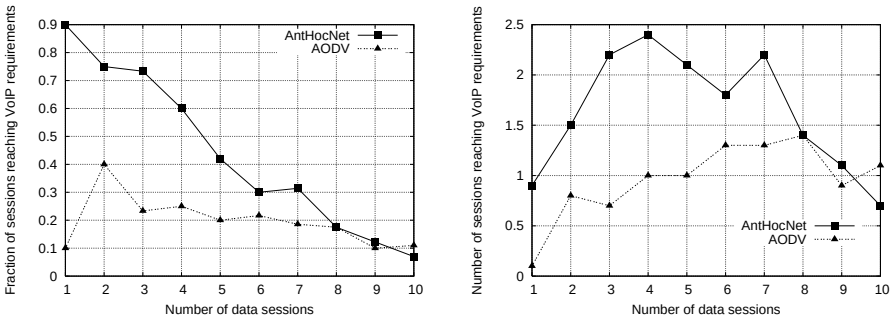


Figure 6.9: Results for AntHocNet and AODV with VoIP level data rates ($25 packets/s$) and varying numbers of data sessions in the urban scenario. We report (a) delivery ratio, (b) average end-to-end delay, (c) average delay jitter, (d) overhead in number of packets, and (e) overhead in number of bytes.

bandwidth is wasted. For existing work on admission control in AHWMNs, see e.g. [81].



(a) (b)

Figure 6.10: Sessions reaching VoIP quality requirements in terms of delivery ratio, delay and jitter: (a) as a fraction of the total number of started sessions and (b) in absolute numbers.

6.3 Conclusion

In this chapter, we have studied the behavior of AntHocNet in an urban scenario. First, we have given a detailed description of how we obtained a good and efficient simulation of the working of an AHWMN in an urban environment. Then, we have presented results of a set of experiments in which we compared the performance of AntHocNet and AODV in this setup.

As a setting for our urban scenario experiments, we used an area of $1561 \times 997m^2$ in the center of the Swiss town of Lugano. We modeled node mobility in this area using an adaptation of RWP in which node movements were limited to the streets and open spaces in the town. Urban radio propagation was modeled using a preprocessing step, in which the network surface area was discretized and signal strengths between each of the discrete points was calculated off-line using a raytracing approach. Finally, network data traffic was modeled using bidirectional point-to-point data sessions with varying data rates that could represent the typical data load generated by different types of interactive communication applications. We used the QualNet discrete event network simulator, and used the same network protocols as in the open space experiments of the previous chapter.

In the results section, we first investigated general network properties of the AHWMN in the urban environment. We observed that compared to open space scenarios, nodes typically have less neighbors, network connectivity is less good, paths are longer, and the network change rate is higher. Then, we discussed the results of a number of tests in which we compared the performance of AntHocNet to that of AODV. In the first tests, we studied the effect of increasing data rates. We observed that AntHocNet could outperform AODV for most data rates, but noticed that it had more difficulties with the highest data rate. Next, we investigated what happened when the number of data sessions was increased. We saw that AntHocNet could each time outperform AODV, and that it was

AODV that had large difficulties when the number of sessions became high. Then, we did tests with increasing number of nodes. Both algorithms had more difficulties in networks with more nodes. AntHocNet outperformed AODV when low and medium data rates were used, but had more difficulties at the highest data rate. After that, we also investigated what happened with increasing node speeds. We found that increasing the node speed had a relatively small negative effect on the performance of both algorithms. Finally, we considered whether it would at all be possible to deliver the level of service needed to support VoIP conversations in the given scenario. Our conclusion is that using AntHocNet, it is in principle possible, if at least a mechanism is applied to control which sessions are allowed to start.

Chapter 7

Towards the implementation of adaptive routing in AHWMNs

Most existing work on the development of routing algorithms for AHWMNs is carried out in simulation, rather than in real deployment studies. This is also the case for the work that has been presented in this thesis. An important factor that influences the choice for simulation as a research tool is that building an AHWMN testbed is expensive and requires a substantial effort, in which many technological issues need to be resolved. Simulation is often the only real option for researchers who want to investigate a specific aspect of AHWMNs such as routing. Moreover, simulation also gives other advantages, such as the ease with which different and large test scenarios can be investigated, and the possibility to easily repeat previous tests. Nevertheless, since a few years there is a growing awareness in the AHWMN research community that simulation experiments have their limitations [54,117,124,260]. The main concern is that the abstractions and simplifications used in simulation studies can have a significant impact on the network behavior, so that observed results can diverge from what can be expected in reality. Consequently, researchers are increasingly interested in the use of real world testbeds to support research in AHWMNs. Several such testbeds have been set up, such as the Roofnet experiment of MIT [30] and the Magnets project of DTL [148].

In the current chapter, we describe work that we have done regarding the implementation of adaptive routing in AHWMNs. We have developed a system, called MagAntA[1], that is aimed at supporting adaptive routing in Linux based AHWMNs. MagAntA runs as a daemon in user space. It is written in C, and possesses a modular structure, which makes it easy to adapt and extend with

[1]The name MagAntA is derived from the Magnets testbed set up at DTL, the color magenta of the DTL logo, and the ants that formed the initial inspiration for the adaptive routing algorithms we study.

new algorithms.

In what follows, we first describe related work on AHWMN testbeds and the implementation of AHWMN routing algorithms. Then, we describe our MagAntA routing system. Finally, we discuss how MagAntA integrates with the network protocol stack in the Linux kernel. The material described in this chapter was based on work carried out during an internship at DTL. It has been presented in [101].

7.1 On the deployment of AHWMNs

Here, we provide a general discussion on the deployment of AHWMNs. We first discuss existing AHWMN testbeds. In particular, we give details about the Magnets testbed of DTL, in order to give a concrete example of the kind of networks that MagAntA was developed for. Then, we discuss the implementation of AHWMN routing algorithms. We first describe in general the kind of issues that need to be addressed when implementing a routing algorithm for AHWMNs. Then, we discuss existing implementations that have appeared in the literature.

7.1.1 AHWMN testbeds

A number of AHWMN testbeds have been developed for research purposes and described in the literature. These include the Roofnet project of MIT [30], the Magnets project of DTL [148], the BerlinRoofNet project run by students of Humboldt University in Berlin [1], the APE testbed of Uppsala University in Sweden [181], etc.. Most of these testbeds are WMNs, although the APE testbed is meant for the deployment of MANETs. Apart from these research testbeds, a number of WMNs have also been deployed by local authorities in order to provide services to the general public. These include WMN projects in Taipei [10] and Philadelphia [12]. Finally, some WMNs are arising in truly ad hoc fashion from the voluntary efforts of computer enthusiasts. Examples of such networks are the olsr.freifunk.net experiment in Berlin [9], and the FeuerFunk experiment in Vienna and other Austrian cities [2]. Here, we discuss details of the Magnets research testbed of DTL. Magnets was the network that MagAntA was in the first place developed for. Its description will help the reader to get a concrete idea of the kind of environment we are working in.

The Magnets network is a WMN that is deployed in the center of Berlin. It contains a wireless backbone network of 5 nodes and a wireless mesh of 50 nodes. The structure of the backbone network is shown in figure 7.1. It consists of 5 mesh routers, with point-to-point links between them. The mesh routers are placed on top of tall buildings, so that they are in line of sight of each other. The links between them are realized using directional antennas and are based on 802.11 technology. Two of the links (the one between the T-Systems and ETF buildings, and the one between the T-Labs and TC buildings) work at a frequency of $5Ghz$, while the others work at a frequency of $2.4GHz$. Be-

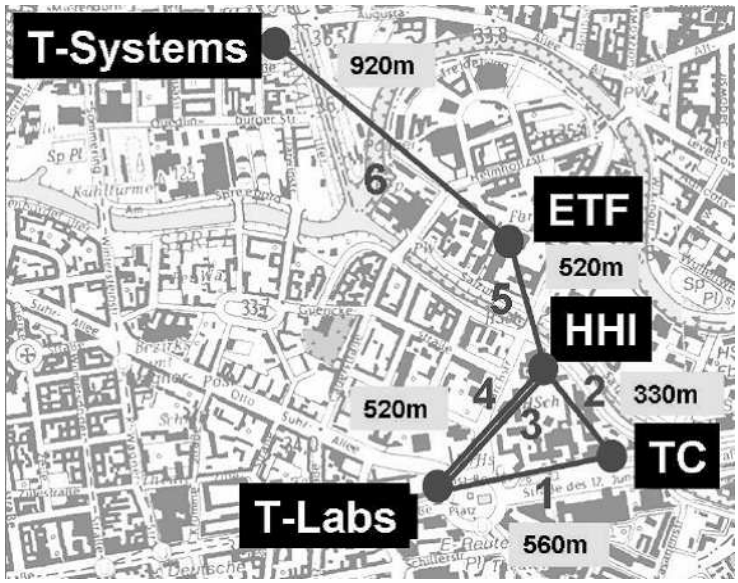


Figure 7.1: The layout of the Magnets backbone network in the center of Berlin. The figure shows the name and the location of the buildings on which the backbone nodes are placed, and indicates the wireless links that exist between them, with their lengths. Figure taken from [148].

tween two of the nodes (the HHI and T-Labs buildings), there are two parallel links. Thanks to the spatial reuse of the directional antennas, these links can be considered independent, so that they offer possibilities for data load balancing. The backbone network provides reliable data transport at high rates (in experiments, throughputs of up to $55Mbps$ have been measured). A detailed study of the properties and performance of the backbone network has been described in [147]. The mesh part of the Magnets network consists of 50 nodes that are routerboards with multiple Mini PCI cards. They use omnidirectional antennas and run 802.11a/g technology. While the organization of the backbone nodes is the result of careful planning and tuning, the mesh nodes are placed in ad hoc manner, and their number can easily be extended. The combined use of backbone and mesh nodes gives the magnets network a strongly heterogeneous character. This heterogeneity will be increased in a later stadium, when the multiple Mini PCI slots of the mesh nodes will be used to integrate different technologies, such as GPRS [112], UMTS [11], WiMax [5], Bluetooth [33] and Zigbee [13]. Moreover, the plan is to also investigate the integration with other WMNs that exist in the city of Berlin, namely olsr.freifunk.net and BerlinRoofNet. All this heterogeneity is the main difference between Magnets and other WMN testbeds, such as Roofnet and BerlinRoofNet, which are flat networks consisting of only one type of nodes.

7.1.2 Implementations of AHWMN routing algorithms

When implementing a routing algorithm for AHWMN deployment, one has to deal with a number of important issues [38]. A first of these is addressing. WMN nodes should be able to identify themselves even before IP addresses are assigned. This is not so much a problem for mesh routers such as the nodes in the Magnets backbone network, but it is for mesh nodes that are connected to the network in ad hoc fashion. For these nodes, connectivity to a server providing IP addresses cannot always be guaranteed. Moreover, some form of routing will need to be performed to reach such a server. Therefore, address auto-configuration is important. A second issue is the integrated use of the different wireless interfaces available on each node. A node that has for example a bluetooth interface and an 802.11 interface will have different neighbors on each of these two interfaces, so that it in fact takes part in two different networks. These different networks should be integrated to form one AHWMN. A third issue is the integration with the existing TCP/IP network protocol stack and the routing protocols therein. AHWMNs often need to connect to other networks (e.g., the Internet), running traditional routing protocols such as OSPF or RIP (see section 3.1). It is important that these protocols integrate smoothly with the AHWMN routing protocol. Finally, an important aspect for any system addressing these issues is that it should require minimal or no changes to existing operating systems and firmware, in order to provide easy installation and portability.

A number of approaches to implement AHWMN routing functionality have appeared in the literature. Here, we give a short description of the most interesting ones. We give special attention to how they deal with the different issues listed above.

The first implementation of an AHWMN routing protocol that was made publicly available is Mad-hoc AODV [174], an implementation of AODV for use under Linux. In this implementation, all code runs in user space, and no modifications whatsoever are made to the Linux kernel. This means that Mad-hoc AODV can get only limited feedback about what is going on inside the kernel and in particular inside the routing layer in the network protocol stack. It deals with this problem by making use of snooping: it monitors what kind of packets are being sent out by the kernel and derives from that which routing information is needed and which actions should be taken. For example, to find out that a new route setup is needed, Mad-hoc AODV checks for request packets sent out by the Address Resolution Protocol (ARP) [215]. ARP is a protocol used in broadcast type networks such as Ethernet in order to map IP addresses to MAC addresses: when a node does not know the MAC address that corresponds to a give IP address, it uses ARP to find this out. Therefore, when Mad-hoc AODV observes that the local node is sending out an ARP request message, it infers that the node does not know how to reach a certain destination, and reacts to this with the transmission of an AODV RREQ. Once it has obtained a route, it updates the routing tables in IP. In a similar way, all other AODV events are derived from packet snooping. The functions needed to perform snooping, as well

as the functions to update IP routing tables, exist in the kernel and are available to user space programs. The approach of making absolutely no modifications to the kernel gives Mad-hoc AODV the advantages that code development and installation are easy, and that it is portable across different systems. There are however several disadvantages. In the first place, the approach is quite inefficient. In the example of the snooping of ARP requests to define the need for a route setup, it is clear that the sending of the ARP request is not needed and leads to extra overhead. A second disadvantage is that the system is dependent on the working of ARP and IP and the synchronization between them. Apart from these disadvantages, we point out that also other issues, including address auto-configuration, the integration of different network views, and the integration with other networks, are not dealt with in this system. Moreover, Mad-hoc AODV was found to contain several severe bugs (see e.g. [199]). Now, it is no longer supported or distributed.

In [57], the authors present an different implementation of the AODV routing algorithm, called AODV-UCSB. This implementation makes use of Netfilter [144], a packet filtering framework that is available inside Linux. Netfilter consists of a number of hooks that are present at various points in the Linux network protocol stack. Data packets traversing any of these hooks are diverted to functions in user defined kernel modules, where they can be examined, modified, queued or dropped. In the case of AODV-UCSB, a kernel module is defined that stores incoming and outgoing data packets until a routing decision is taken for them. The actual routing is done by a daemon running in user space, which acts upon the stored data packets. This way, the use of other routing algorithms is circumvented. An advantage of this approach is that the routing code running in user space can easily be adapted and modified, while the necessary kernel code is limited to the definition of a kernel module, which is easier to install and port than a kernel modification [216]. A disadvantage of the approach is that it does not include solutions for integration with other networks, for address auto-configuration, and for the integrated use of different network interfaces. A different implementation of AODV that follows a similar approach to AODV-UCSB is AODV-UU [181]. In its newest versions, this system does include solutions for addressing and integration with other networks.

In [69], yet another implementation of AODV is described. Here, AODV is implemented as an extension inside ARP. As explained before, the ARP protocol allows nodes in broadcast networks to find out the MAC address that is associated with a certain IP address. In the normal mode of operation, a requesting node A broadcasts an ARP request message with the IP address of a node B. If B receives the message (i.e., a node with the requested IP address exists on the same broadcast network), it sends back an ARP reply message containing its MAC address, and A can start sending to B. A special case is when B exists in a different network that is connected to A's network via a gateway node G. G can then answer to A's ARP request, and A maps B's IP address to G's MAC address, indicating that all packets for B need to be sent to G. This technique is called Proxy ARP [48]. It allows communication with remote nodes in a way that is transparent to the routing algorithm at the IP

layer: to IP, it seems that the remote node B is reachable in one hop and that no routing is needed. The authors of [69] adapt the working of Proxy ARP to integrate AODV in it. ARP requests are replaced by AODV requests, and they can be forwarded over multiple hops. ARP replies are replaced by AODV replies. Routes are set up by associating the IP address of the destination node to the MAC addresses of the different next hops along the path between source and destination. This approach has as an advantage that it allows a smooth integration with the TCP/IP protocol stack and with traditional routing protocols. This is because the AHWMN routing is done completely inside ARP, transparent from the IP layer and its routing algorithms. An important disadvantage is that it requires a change in the ARP protocol, which is not straightforward to implement and not easily portable. Other disadvantages are that the system relies on the availability of IP addresses (and thus does not solve the address auto-configuration issue), and that it does not provide integration between the network views provided by the different network interfaces that are present in a node.

An interesting property that makes the latter implementation different from the former ones, is that it is not implemented at layer 3 (the IP layer), but below it: since ARP functions as some kind of glue between the IP layer and the MAC layer, it can be seen as residing at layer 2.5. The main advantage of working at layer 2.5 is the possibility to offer AHWMN functionality in a way that is transparent to the IP layer and the routing algorithms therein. This transparency makes it easier to integrate with other networks and to allow coexistence with traditional Internet routing protocols. A number of other AHWMN routing implementations follow a layer 2.5 approach. A first example is LUNAR [259], which also integrates AHWMN routing with ARP. Like the AODV implementation described above, it replaces ARP requests by route requests which can be forwarded to set up paths. LUNAR has as an interesting feature that it repeats route setups every three seconds in order to proactively deal with changes in the network (in this way, it can be considered a hybrid approach like AntHocNet; see also subsection 4.3.3). The system also provides solutions for address auto-configuration and integration with other networks. A different example of a layer 2.5 approach is Lilith [264]. Here, the routing algorithm is not implemented inside ARP, but inside Multiprotocol Label Switching (MPLS) [223]. MPLS is a protocol for constructing paths at layer 2.5, transparent to IP. In Lilith, MPLS path setups are replaced by route setups of a simple reactive routing protocol. Like LUNAR, also Lilith can deal with address auto-configuration and integration with different networks. Finally, two other ad hoc routing implementations that work at layer 2.5 are MCL [7], which runs under Windows, and Ana4 [37,38]. They both follow an approach that uses an ad hoc virtual interface. The idea is to define an extra, virtual interface next to the interfaces that are already present in the nodes. This virtual interface contains all the functionality related to ad hoc networking, including the routing algorithm. For details about this approach, we refer to subsection 7.3.1, where we describe Ana4.

7.2 The MagAntA routing system

In this section, we describe the MagAntA routing system. First, we present the general structure of the system. Then, we discuss each of its different components separately. These are the control module, the routing interface and the routing module. We conclude the section with a description of the adaptive routing algorithm that is currently implemented in MagAntA.

7.2.1 The program structure

A schematic representation of the MagAntA routing system is shown in figure 7.2. It consists of a daemon that runs in user space. The advantage of working in user space as opposed to kernel space is that coding and debugging is easier, while a disadvantage is that the system's working is less efficient. We opted for this approach because we see MagAntA mainly as a research tool, rather than a finished product for routing.

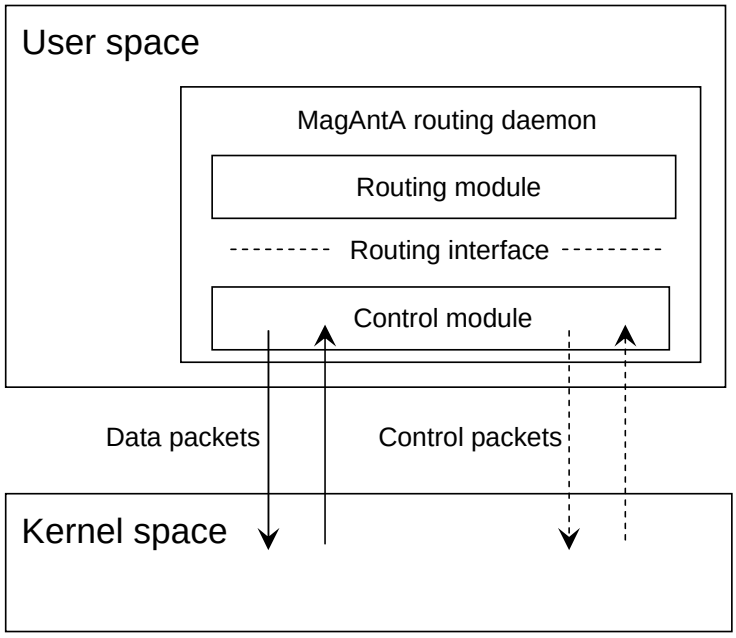


Figure 7.2: The structure of the MagAntA routing system.

The MagAntA routing daemon performs routing on a per packet basis, meaning that it provides a new routing decision for each data packet. This gives maximum flexibility to the system, and allows to implement adaptive routing strategies such as probabilistic data forwarding. In order to implement per packet routing, MagAntA relies on kernel modules that send all data packets up to user space. These kernel modules are developed independently by fellow

researchers at DTL and are therefore described separately in section 7.3; here we focus our attention on the user space routing daemon.

Internally, the program structure of MagAntA consists of two main modules: the control module, which provides basic functionality that is needed for any routing algorithm, and the routing module, which contains the actual routing logic. The two modules communicate through a simple interface, which we call the routing interface. The advantage of this modular structure is that the routing algorithm code can be developed independently from technical issues. Below, we describe each of the two modules and the interface. For a more detailed account of the system, we refer to the related technical report [100].

7.2.2 The control module

The function of the control module is to sustain the working of the daemon and to provide basic functionality to the routing module. The daemon is implemented using an infinite for-loop, in which the program waits for different events to take place. In particular, it waits for data packets coming from the kernel, for incoming control packets, for routing events timing out, and for user interrupts (which can be used to pause the daemon). When either of these happen, it takes the appropriate actions.

The functionality provided by the control module to the routing module includes four basic services. The first is to gather information about the local host, such as the available network interfaces and their MAC addresses, the local IP address (if it is available), etc.. It presents this information to the routing module, so that it can be used by the routing algorithm. The second service is the transfer of data packets to and from the kernel. For this part, it communicates with kernel modules that send packets up to user space. Details about these kernel modules are given in section 7.3. The control module deals with the technical issues involved in interacting with the kernel modules and hides these from the routing module. Upon reception of a data packet, it stores the packet in a static buffer, and provides the routing module with pointers to fields in the packet headers that are relevant for routing, such as the destination address, the next hop address, the outgoing interface, etc.. The routing algorithm can then work directly with these pointers without being aware of the structure of the data packets or their headers. The third service is the sending and receiving of control packets. For this service, the control module communicates directly with the device drivers of the different network interfaces in the kernel. This is done through the use of raw packet sockets that are standard available in the kernel [244]; no kernel modules or modifications are needed. To the routing module, several different possibilities to transmit control packets are provided. These include unicast and broadcast transmission, as well as delayed broadcast transmission, in which the control packet is broadcast after a short random jitter (this is useful for the implementation of flooding). The fourth service is event scheduling. The control module keeps an event list, to which the routing module can add any event that it needs. Whenever an event times out, the control module calls the event handling function of the routing module.

7.2.3 The routing interface

The function of the routing interface is to form a bridge between the control module, which provides basic functionality, and the routing module, which contains the routing protocol logic. Its most important component is the *routing_interface* structure, which defines all communication possibilities between the two modules. Apart from this structure, the routing interface code contains also a number of other structures and functions that provide useful functionalities for the routing module. These include functions to copy and compare addresses and time structures.

The contents of the *routing_interface* structure are shown in figure 7.3 (some details, such as the parameters taken by each of the functions, are left out here to improve readability). It contains a pointer to a *local_data* structure and a number of function pointers. The *local_data* structure contains information about the local host, and is filled in by the control module. It contains the name of the local host, its IP address, the IP, ad hoc and MAC broadcast addresses, and the list of devices available for ad hoc routing, with their names and MAC addresses. This is basic information that is needed by the routing module. The function pointers of *routing_interface* are used for the communication between the control module and the routing module. The first half of these pointers are filled in by the control module, in order to allow the routing module to make use of the functionality it provides. They include a function to send data packets to the kernel (either to be delivered locally or to be forwarded), different functions to send control packets over the network (see subsection 7.2.2), a function to schedule events, and one to remove previously scheduled events. The other half of the function pointers are filled in by the routing module. They allow the control module to call the routing module whenever needed. They include a function to initialize the routing algorithm and one to start it, a function to deliver received data packets to the routing algorithm, one to deliver received control packets to it, and one to pass it events that have timed out.

7.2.4 The routing module

The routing module contains the actual routing code. Since lower level technical issues are taken care of by the control module, this code can concentrate on the algorithm logic. The routing interface provides a clear definition for the interactions between the routing module and the control module, which simplifies the design and integration of new routing algorithms. On the one hand, the interface describes the functions that are provided by the control module and can be used by the routing module (the top half of the function pointers shown in figure 7.3). On the other hand, it describes the functions that the routing module needs to implement so that the control module can call it (the lower half of the function pointers shown in figure 7.3). To integrate a new algorithm into the system, it is sufficient to provide implementations for these latter functions and initialize the function pointers with them. Currently, we have implemented one adaptive routing algorithm in the routing module of MagAntA, which we

```
struct routing_interface {
      struct local_data *local_data;

      void (*send_data_packet)();
      void (*control_unicast)();
      void (*control_broadcast)();
      void (*control_broadcast_jitter)();
      void (*schedule_event)();
      void (*unschedule_event)();

      void (*initialize_routing_algo)();
      void (*start_routing_algo)();
      void (*deliver_data_packet)();
      void (*deliver_control_packet)();
      void (*handle_event)();
};
```

Figure 7.3: Declaration of the *routing_interface* structure. Some details were left out to improve readability; e.g., the parameters taken by each of the functions are left out here.

describe below in subsection 7.2.5. We would like to point out that the followed approach is quite general, and that also more traditional routing algorithms, such as AODV or OLSR, could be added to the system.

7.2.5 The MagAntA adaptive routing algorithm

The adaptive routing algorithm that is currently implemented in MagAntA can be considered a simplified version of AntHocNet. The algorithm is partly reactive, in the sense that nodes only gather routing information about destinations that they are communicating with. Therefore, when a source node starts a data session to an unknown destination, it executes a route setup process, in which it looks for an initial route to the destination. During the course of the session the algorithm works proactively: the source node periodically sends out ant agents towards the destination, in order to find new paths, and to update information about existing ones. Pheromone values are based on the round-trip-time experienced by the ants. Data are forwarded stochastically according to the pheromone tables. Link failures are detected using hello messages and are dealt with using warning messages. In what follows, we give details about each of these elements.

When a node in the network starts a data session to an unknown destination, it reactively starts a route setup process, in which it searches for initial routing information. This process consists of flooding a reactive forward ant over the network in search of the destination. When the ant arrives in an intermediate

node, this node's address is added to the ant, and the ant is forwarded. The forwarding at intermediate nodes is normally done by broadcasting, unless the ant is in a node where routing information to its destination is available. In that case, the ant is unicast to a stochastically chosen next hop. The probability of choosing a next hop n is given in formula 7.1.

$$P_{nd} = \frac{(\tau_{nd})^{\beta_1}}{\sum_{j \in N} (\tau_{jd})^{\beta_1}} \tag{7.1}$$

In this formula, τ_{nd} is the pheromone associated with neighbor n and destination d. By raising it to a power β_1, exploration can be encouraged ($\beta_1 < 1$) or limited ($\beta_1 > 1$). When a copy of the forward ant reaches the destination, a backward ant retraces the forward path and updates the routing tables of intermediate nodes. In this way an initial route is established between the source and destination of the session.

During the course of the session, the source node proactively tries to improve and extend existing routing information. To this end, it periodically sends out proactive forward ants to sample existing and new paths towards the destination. Proactive ants are not broadcast, but are unicast based on the pheromone values. Like during the unicasting of reactive forward ants, a next hop is chosen probabilistically at each node. The probability for a proactive forward ant with destination d to take next hop n is given in formula 7.2.

$$P_{nd} = (1 - q) \cdot \frac{(\tau_{nd})^{\beta_2}}{\sum_{j \in N} (\tau_{jd})^{\beta_2}} + q \cdot \frac{1}{|N|} \tag{7.2}$$

In this formula, $(1 - q)$ is the probability that the ant will be forwarded using the pheromone values. Again, we raise the pheromone value τ_{nd} to a power β_2, in order to increase exploration ($\beta_2 < 1$) or decrease it ($\beta_2 > 1$). q is the probability that uniform forwarding is applied. This is another way of enhancing exploration: each of the $|N|$ neighbors can be picked with probability $(1/|N|)$. While a low value for power β_2 can allow the exploration of paths with low pheromone, uniform forwarding allows to choose neighbors for which no routing information at all is available. Different from AntHocNet, we do not implement pheromone diffusion (see subsection 4.2.3) here, in order to keep the first algorithm implementation in MagAntA simple. Therefore, there is no virtual pheromone available to guide the exploration of unknown paths. The source nodes stop sending proactive ants to a destination if for a certain period no data have been sent to this destination.

Pheromone variables are calculated based on the round trip times (RTT) experienced by the ants. When a forward ant passes through a node, this node adds its address to the ant, and a timestamp. When the backward ant arrives back at the same node, the timestamp in the ant is compared to the current time in the node to get the RTT. This allows to get a good estimate without the need for synchronized clocks in the different nodes. Pheromone values are updated as indicated in equation 7.3. As pheromone represents a goodness value rather than a cost, we update with δ_{id}, which is the inverse of the RTT

from the current node through neighbor i to destination d. γ is a discounting factor. The use of RTT as a routing metric is different from AntHocNet, where the best results were obtained when paths were evaluated based on information about the quality of the wireless radio link (see subsections 4.2.6 and 5.3.2). Unfortunately, in many existing systems it is difficult to obtain such information from the physical or MAC layer protocols, and we decided therefore not to use it here. In a later implementation, we could implement local probing by the routing algorithm itself to find out about link quality. This is the approach followed in the Roofnet project of MIT [30].

$$\tau_{id} \leftarrow \gamma \cdot \tau_{id} + (1 - \gamma) \cdot \delta_{id} \tag{7.3}$$

Data packets are forwarded stochastically according to the pheromone values. However, different from proactive forward ants, there is no uniform routing for data packets, and the power exponent β_3 that the pheromone is raised by is higher than for ants, so that there is a stronger preference for the best paths. This is because stochastic routing for data packets is not meant for exploration, but rather to spread the data load over multiple available good paths. The probability for a data packet with destination d to take a next hop n is given in formula 7.4.

$$P_{nd} = \frac{(\tau_{nd})^{\beta_3}}{\sum_{j \in N} (\tau_{jd})^{\beta_3}} \tag{7.4}$$

Link failures are detected via hello messages. Hello messages are short beacon messages which are sent out periodically by each node. When a node hears a hello message from another node, it knows that this node is its neighbor. When for a multiple of the hello beacon period no beacon was received anymore from the other node, the connection to the neighbor is considered broken. This mechanism for failure detection is similar to the one used in AntHocNet (see subsection 4.2.5). However, in AntHocNet link failures could additionally be detected when unicast transmissions failed. This is not possible here, because feedback about transmission failures are normally not given by the Linux routing protocol stack (one possible solution is available if the Linux kernel contains the Linux Wireless Extension [258], which is at the moment however not supported in many systems and for many drivers).

When a connection failure is detected, the routing algorithm simply removes all the entries concerning this neighbor from its pheromone table. It does not immediately warn other nodes about the changed situation. However, it is possible that due to the lost connection, the node now no longer has a path to some destination. If this is the case, and the node is still receiving data packets for this destination, it unicasts a warning back to the node that is sending these data packets, stating that this route is no longer valid. This approach of dealing with link failures is different from AntHocNet, on the one hand to keep the first implementation in MagAntA simple, and on the other hand because we expect to use MagAntA on WMNs rather than MANETs, where network change rates are lower and fast reactions to link failures are therefore less crucial.

7.3 Integration with the Linux kernel

In this section we discuss how the MagAntA routing daemon running in user space integrates with the TCP/IP network protocol stack inside the Linux kernel. First, in subsection 7.3.1, we explain the approach using Ana4, a layer 2.5 architecture for the deployment of WMNs. A prototype of this system has been developed by fellow researchers at DTL [233]. However, a number of problems were also encountered. We discuss these problems and the current state of the system in subsection 7.3.2. Then, in subsection 7.3.3, we discuss a number of possible alternative approaches.

7.3.1 Ana4

The current implementation of MagAntA relies on Ana4 [37,38], an architecture for the deployment of AHWMNs using Linux. Ana4 provides solutions for AHWMN related issues such as address auto-configuration, combining the views provided by different network interfaces, and providing a smooth integration with the existing protocols in the TCP/IP network protocol stack (see also subsection 7.1.2). Its functionality is placed in an ad hoc protocol layer, which is located at layer 2.5, between the link layer (layer 2) and the IP layer (layer 3). By working at layer 2.5, Ana4 can hide AHWMN related complexity from protocols both at lower and higher layers, so that no changes are needed for those. For the interaction between the Ana4 system in the Linux kernel and the MagAntA routing daemon in user space, we make use of a new version of Ana4 [233], developed at DTL, in which each individual data packet is sent up to user space to be routed. An overview of the full system is given in figure 7.4. In what follows, we explain details of this setup. We discuss how Ana4 is implemented inside the TCP/IP network protocol stack in the Linux kernel, how it deals with issues of AHWMN deployment, and how it interacts with the MagAntA routing daemon.

The integration of Ana4 at layer 2.5 is based on the definition of a virtual interface: Ana4 presents itself as a new, virtual network interface, which exists in parallel to the real interfaces that are present in the node. We refer to this interface as the Ana4 ad hoc interface. For the IP protocol at layer 3, the Ana4 system appears to be an extra layer 2 interface. This situation is illustrated in figure 7.5a. IP considers this new interface as the single interface over which it sends and receives all data packets to and from the WMN. Inside the virtual interface, packets are in the layer 2.5 architecture, where they are treated by Ana4. If they are outgoing packets coming from IP or if they are packets in transit, they are assigned a next hop and outgoing (real) interface, and are passed on to this interface. If they are incoming packets arriving at their destination, they are passed on to IP. For the other (real) interfaces at layer 2, the Ana4 ad hoc layer appears as another layer 3 protocol that they exchange data packets with, more or less in parallel to IP. This is illustrated in figure 7.5b. The interfaces distinguish between packets from IP and those from the ad hoc layer based on the MAC type field in the packet headers.

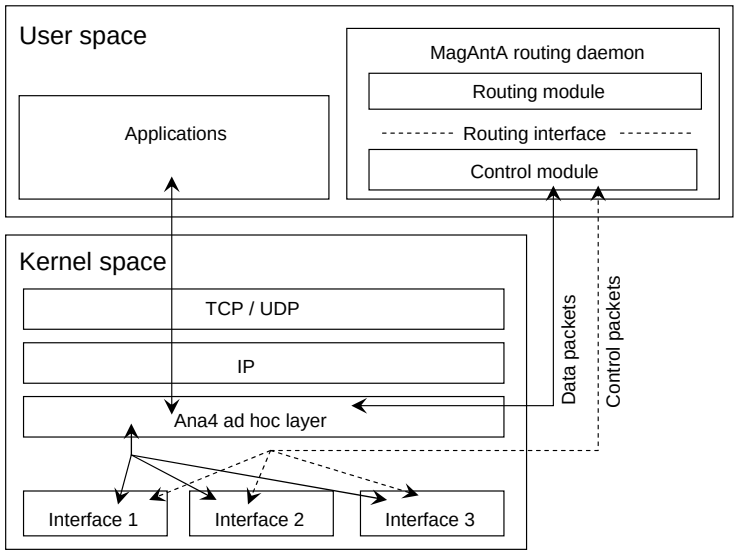


Figure 7.4: The integration of MagAntA with the Linux kernel using Ana4.

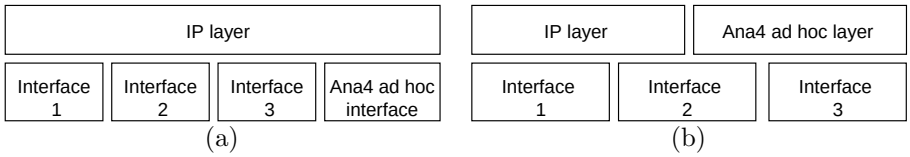


Figure 7.5: The Ana4 ad hoc layer, (a) as seen from layer 3, and (b) as seen from layer 2.

Let us now give some details about how Ana4 deals with the AHWMN deployment issues that we pointed out in subsection 7.1.2. The first of these is addressing. Ana4 maintains a separate ad hoc address space. Each node independently derives its own unique ad hoc address, based on one of its MAC addresses, and maintains a table with known mappings between IP addresses and ad hoc addresses for other nodes in the network. Data packets are given a small extra header in which the ad hoc addresses of their source and destination are stored, and packet forwarding inside the AHWMN is based on these addresses. The second issue is the integrated use of the different wireless interfaces. This is obtained through the virtual interface approach: while the ad hoc layer sends and receives packets over the different available interfaces, IP only communicates with the virtual interface, so that it perceives the AHWMN as a single interface network. This makes it easy to deploy heterogeneous AHWMNs such as the Magnets testbed described earlier in subsection 7.1.1, where nodes communicate through different wireless technologies. Even, there is no

need for the used interfaces to be wireless, and one could also include wired links in the network. The third issue is integration with other networks and existing IP routing protocols. Ana4 performs routing inside the ad hoc layer. It assigns next hops to data packets that are sent over the AHWMN based on an internally maintained routing table. By placing AHWMN routing inside the ad hoc layer, it is transparent to IP. This means that from IP's perspective, the AHWMN seems to need no routing, so that it appears like a flat, single-hop network, similar to an Ethernet bus. This way, compatibility with IP routing algorithms running in the Internet is straightforward. Finally, we point out that all the code for Ana4 is defined in kernel modules. This means that no kernel modifications are needed, allowing easy installation and good portability.

In the original Ana4 system, data forwarding is based on the use of three tables: the ARP table, which contains the mapping between IP addresses and ad hoc addresses for known destinations, the COM table, which is the actual routing table holding a next hop ad hoc address for each destination, and the ATP table, which is the table with lower layer information storing an outgoing interface name and MAC address for each neighbor (next hop). A simplified representation of the different tables is shown in figure 7.6. The tables are kept in the ad hoc layer in kernel space, and are updated by a routing daemon running in user space, through the use of IOCTL commands. This way, the user space routing algorithm defines the forwarding of data packets indirectly through the tables. In the current version of Ana4, the proactive OLSR protocol is implemented for this purpose. For the working with MagAntA, this table based approach is not satisfactory. Since we want to support per packet stochastic data load spreading, we need to be able to take a new routing decision for each individual packet. This is not possible if the routing daemon influences data routing indirectly through the routing tables that are present in Ana4. Therefore, we use a different version of the Ana4 system [233], in which the ad hoc layer sends the address part of each packet up to user space (sending the data part is avoided to improve efficiency), through a character device file. The MagAntA routing daemon then fills in the necessary address fields in the ad hoc and MAC headers, and sends the packet back down. After that, the ad hoc layer resumes its normal operation. This approach is illustrated in figure 7.4, where the solid arrows show the flow of data packets.

7.3.2 Current state of the system

The MagAntA system has been implemented to work with the Ana4 architecture described above. The adaptation of Ana4 that sends data packets up to user space was developed by colleagues at DTL [233]. Currently, their system is working sufficiently well so that preliminary testing and debugging of the MagAntA code could be performed. However, there are some problems with the adapted Ana4 system that make the performance unsatisfactory and limit the possibility for extensive testing of MagAntA.

The most important problem with the adapted Ana4 system is that it is unstable and fails after a number of packets have been sent. This has to do

(a)

IP address	Ad hoc address
192.168.1.1	000e:35b3:dedb:0001
192.168.1.3	0002:442a:35ed:0001
...	...

(b)

Destination	Next hop
000e:35b3:dedb:0001	0002:442a:35ed:0001
0002:442a:35ed:0001	0002:442a:35ed:0001
...	...

(c)

Neighbor ad hoc address	MAC address	Device
0002:442a:35ed:0001	00:02:44:2a:35:ed	eth1
...	...	...

Figure 7.6: The different tables used for routing in Ana4: (a) the ARP table maps IP addresses to ad hoc addresses, (b) the COM table maps destination ad hoc addresses to next hop ad hoc addresses, and (c) the ATP table maps next hop ad hoc addresses to MAC addresses and outgoing device names. Example adapted from [38].

with locking issues. The original Ana4 system uses a lock on the entire module for the period that a data packet is being held. However, when each data packet is sent up to user space, such a lock cannot be maintained. Efforts were made to solve this problem, by breaking the lock into smaller ones, but they lead to unstable behavior of the system. Another problem is the low performance. The followed approach to send the data packets from kernel to user space using a character device file involved additional copying of the address part of the packets, introducing extra delay in packet processing. Precise performance data for the adapted Ana4 system are not given in [233], but the system is stated to be "too inefficient to cope with even the relatively small packet rates that occur in WMNs". Finally, a third problem is that there is relatively little support for Ana4. The researchers that were involved in the development of the original system have moved on to other projects and the system is no longer actively supported.

Due to the above problems, we are currently looking for a different way to integrate MagAntA with the Linux kernel. We describe some candidate solutions below in subsection 7.3.3.

7.3.3 Other approaches

Here, we explore other possibilities to integrate the MagAntA routing daemon with the Linux kernel. We look for systems that allow as much as possible to stick to the original design architecture laid out for MagAntA. We first discuss an approach that builds on an existing implementation of the AODV routing protocol, and then one that uses Click router.

In subsection 7.1.2, we have described several existing implementations of the AODV routing protocol. Of these, the AODV-UCSB [57] implementation has a design that is similar to the one that we are following with MagAntA. It uses a routing daemon that runs in user space and that makes routing decisions on a per packet basis. The integration with the kernel is obtained through Netfilter and a kernel module. Each packet entering or leaving the node passes through the Netfilter hooks, from where it is redirected to a kernel module. This kernel module temporarily queues the packet. The user space daemon then takes a routing decision for each queued data packet. The main difference with the approach we have been following is that data packets do not go up to user space. Instead, the routing module manipulates the packets in the kernel space. Nevertheless, per packet routing is still possible, and all routing code remains in user space. This means that limited changes would be needed to MagAntA in order to work with this system; it would suffice to adapt the functions in the control module that deal with receiving and sending data packets. The changes could be made transparent from the routing module. Disadvantages of using the AODV-UCSB approach is that, different from Ana4, this system does not provide ready solutions for some important AHWMN related issues such as address auto-configuration, integration of heterogeneous interfaces, and a smooth integration with other networks running traditional IP routing algorithms.

A different approach, which is also proposed in [233] in response to the problems with Ana4, is to use a Click router. Click [155] is a flexible, modular software architecture for building configurable routers under Linux. The Click system defines a number of modules, called elements, which carry out basic functionality that is needed in a router. Examples of existing elements are *Queue*, which queues data packets, *LookupIPRoute*, which looks for the destination of a packet in a static routing table, *ToDevice*, which hands packets to a Linux device driver for transmission, etc.. To build a Click router, one needs to select a combination of these elements that together perform the required functionality, and define connections between them. If the functionality of existing elements is not sufficient, one can also program new elements: all elements are written in C++ and are subclasses of the class *Element*. Once a Click router has been put together, it can be placed in the Click system to function as a Linux router. One has the choice between running the router using a Linux in-kernel driver, or using a user-level driver. While the former gives better performance, the latter allows easier debugging and fast prototyping. For MagAntA, the advantage of working with Click would be that we can continue working in user space, and maintain a modular and flexible structure for the program. Another advantage is that Click is widely used and well documented and supported, which makes it an easy to use developing environment. A disadvantage is that we would need to make considerable changes to the structure of MagAntA, in order to make it adhere to the typical structure of a Click router in which the code is organized according to the flow of packets through a connected system of elements. Moreover, we would need to program a number of new Click elements to support functions that are needed in adaptive routing. Another disadvantage is that, unlike Ana4, Click does not support ready solutions for AHWMN related issues

such as address auto-configurations and the like. Since, however, Click allows flexible routing design, such solutions could be implemented.

When comparing the two possibilities described above, it seems that the approach using the AODV-UCSB implementation would be the easiest solution to continue with the work implemented so far. However, the approach using Click gives more flexibility and is better documented and supported, so that in the long run, it might be more interesting to invest the extra effort needed to migrate to Click.

7.4 Conclusion

In this chapter, we have described efforts that we have made towards the implementation of adaptive routing in AHWMNs. We have presented the MagAntA system, a user space routing daemon for adaptive routing. MagAntA has a modular structure, consisting of a control module, a routing module, and a routing interface between these two. This structure, together with the fact that the system runs entirely in user space, makes it easily extendible and adaptable, so that it is an easy to use research tool. For integration with the Linux kernel, the MagAntA system relies on Ana4, a layer 2.5 architecture for the deployment of AHWMNs. The interaction between MagAntA and Ana4 is established through packet forwarding: we use an adapted version of Ana4, developed by colleagues at DTL, in which each data packet is sent up to user space to be routed. The fact that each data packet can be routed individually by the MagAntA routing daemon provides the flexibility that is needed to perform adaptive routing.

The work presented in this chapter is still in progress. While MagAntA has been subjected to preliminary tests, these were only sufficient to do debugging and verify basic functionality. Thorough tests of the system to get performance data have not been carried out yet. This is due to programming problems that were encountered by the developers of the adapted Ana4 system. This system is currently still unstable and gives poor performance. We have described some possible alternatives for the integration of Ana4 with the kernel, using the existing AODV-UCSB system and using Click. Further developments will be needed to find out which approach is best suited for our work.

Chapter 8

Conclusions

In this thesis, we have addressed the problem of adaptive routing in AHWMNs using techniques from AI and ACO. We have first given an introduction to the fields of AHWMNs and adaptive routing. Then, we have described the AntHocNet routing algorithm, and next, we have presented a range of experimental tests in which we investigated the behavior and performance of this algorithm. The tests were carried out both in traditional open space scenarios and in an urban setting. Finally, we have also described a system for the implementation of ACO algorithms in real world testbeds.

In what follows, we first give an overview of the contributions and findings of this thesis, and then discuss possible future research directions.

8.1 Contributions and findings of this thesis

The main contribution of this thesis is the development of the AntHocNet routing algorithm. AntHocNet has a hybrid architecture, whereby a reactive route setup process is used at the start of each new communication session in order to obtain an initial path for data forwarding, and a proactive route maintenance process is run throughout the duration of the session with the objective to keep information about existing paths up-to-date and to explore new and possibly better paths, continuously adapting to the changing network environment. AntHocNet also possesses a number of reactive mechanisms to deal with link failures, such as the transmission of failure notification messages and the possibility to execute local route repairs.

A distinct feature of AntHocNet is that it is based on methods from AI. In particular, it uses elements from ACO and from dynamic programming. From ACO it inherits the use of continuous Monte Carlo sampling of full paths, and the use of stochastic decision making. From dynamic programming, it adopts the use of information bootstrapping. Both Monte Carlo sampling and information bootstrapping are important paradigms in the field of reinforcement learning. In AntHocNet, Monte Carlo sampling is applied extensively through-

out the different components of the algorithm, while information bootstrapping is the basis of the pheromone diffusion process that is part of the proactive route maintenance process. While the continuous sampling of full paths provides adaptivity and reliability, information bootstrapping gives a highly efficient but potentially unreliable way of spreading routing information. In the combination of both techniques, the sampling practices can be considered a way to confirm the information suggested earlier by the information bootstrapping process. This approach to combining Monte Carlo sampling and information bootstrapping is inherently different from other ways of integrating these two important learning paradigms in the field of reinforcement learning.

We have evaluated the AntHocNet routing algorithm in a wide range of different tests. A first set of tests was based on standard, open space scenarios. This is a common approach in the field AHWMN research, and was adopted here in order to allow fair comparisons. In these tests, we first compared AntHocNet to existing state-of-the-art AHWMN routing algorithms, including AODV, an important reactive routing algorithm, OLSR, a representative proactive routing algorithm, and ANSI, which is like AntHocNet based on ACO. Results showed that AntHocNet could outperform the other three algorithms over a wide range of different scenarios. Specifically, AntHocNet turned out to perform well in tests with increasing levels of mobility, to deal better than the other algorithms with different levels of data load, to cope well with sparse network situations, and to scale well to networks of increasing sizes. On the downside, AntHocNet's advantage was decreased when mobility got very high, and the algorithm was outperformed by AODV when node density got high. In the same test scenarios, we also investigated the internal working of AntHocNet. We alternatively enabled and disabled various components of the algorithm, and investigated their influence. We found that the proactive maintenance process always has a high contribution to the algorithm's performance, and that this process should work as fast as possible (i.e., producing large and frequent update messages), as long as no excessive levels of overhead are reached. The local route repair mechanism was found to give important advantages in large networks, but to have a negative impact in small networks. Among the different routing metrics, the SINR based measure gave best results. Finally, we found that, contrary to results in wired networks, stochastic forwarding of forward ants and data did not give advantages; it is better to exploit the best available information.

A second set of tests was based on an urban scenario. We proposed a simulation setup providing a detailed and at the same time computationally efficient modeling of outdoor radio propagation, node mobility, and user traffic. We applied this model using the street map of Lugano as a reference. We first investigated properties of the urban network graph and compared them to those of equivalent open space environments. We found that nodes have less neighbors, that network connectivity is worse, that paths are longer, and link durations shorter. Next, we compared the behavior of AntHocNet with that of AODV in this setup, applying tests in which we varied data send rate, number of data sessions, node density, and node speed. We found that AntHocNet profits from the lower local density in urban settings to let its proactive mechanism work

efficiently. However, at high rates, it suffers from interference. At very low rates, both algorithms have difficulties due to their approach to construct initial routing information reactively. We found that node density has a strong impact on the delivery ratio, while the node speed has relatively little impact. Finally, we also found that using AntHocNet, it is possible to deliver VoIP levels of service, but only for a limited number of sessions. This points to the need for call admission control mechanisms in urban MANETs.

Finally, we investigated the implementation of ACO routing in a Linux based testbed. We presented the MagAntA system, which is a user space routing daemon that implements ACO routing. MagAntA has a modular structure, consisting of a control module, a routing module, and a routing interface between these two. Thanks to this modular approach and the system's location in user space, MagAntA is easily extendible and adaptable. For the communication with the Linux kernel, the system relies on Ana4, a layer 2.5 architecture for the deployment of AHWMNs that is defined in a number of kernel modules. We use a version of Ana4 developed by colleagues at DTL, in which all packets going to and from the AHWMN are sent up to user space to be treated for routing by MagAntA. This version of Ana4 is in its current state unfortunately still unstable, and MagAntA can therefore not fully be used. We investigated some possible alternative deployment plans for our system.

8.2 Future research directions

Here we point out some future research directions that are relevant for the work presented in this thesis. These concern the deployment and testing of AntHocNet in hardware testbeds, the support of QoS issues in AntHocNet, the use of the ideas behind AntHocNet in other types of networks, the application of other ideas from artificial intelligence in the field of computer networking, and finally the support of autonomic networking.

In chapter 7, we have described a system for the implementation of AntHocNet in a Linux testbed. However, further work is needed in this direction. Implementation and testing in real testbeds is important in the field of networking and especially in the area of AHWMNs. Real deployment tests can bring up issues that did not come up in simulation. To work well in real implementations, changes might be needed to the AntHocNet algorithm. However, it is encouraging to see that the few other algorithms that take into account link quality and use it to proactively improve paths during the course of communication sessions, namely LQSR, Srcr and LUNAR, have come out of research on real testbeds.

In chapter 6, we have investigated the use of AntHocNet in a detailed simulation of an urban environment. One of the tests was concerned with the support of VoIP traffic. Our finding was that VoIP data traffic can in principle be supported by AntHocNet, if the number of data sessions with such demands is limited. This brings up the issue of admission control and QoS support. Such mechanisms should be integrated with AntHocNet. An interesting aspect is the

fact that AntHocNet uses extensive probing, which could provide the information needed to support QoS. Admission control could be carried out during the reactive route setup process.

Interesting future work would also be to apply the ideas behind AntHocNet in different kinds of networks. One could in the first place target packet switched wired networks. Existing ACO routing algorithms developed for such networks provide adaptivity with respect to variations in data load, and often ignore the effect of topology changes. While such changes are less frequent in wired networks compared to AHWMNs, they can still occur, e.g. due to link or node failures, and the mechanisms developed in AntHocNet could be used to deal with this. Other types of networks in which AntHocNet's mechanisms could be useful are application layer overlay networks. In such networks, nodes are connected at the application layer through virtual links, that are implemented as paths in the underlying physical network. Overlay networks can have very dynamic topologies. Examples of application layer overlay networks are peer-to-peer networks [243] and resilient overlay networks [20].

Another interesting direction for future research is to elaborate further on the relationship between the problem of routing and the field of reinforcement learning. In the reinforcement learning literature, one can find more approaches to learning that can be useful for network routing. One aspect that seems interesting is the possibility to perform off-policy Monte Carlo learning [252]. In off-policy learning the policy used to sample paths is different from the one that information is gathered for. A routing algorithm could then use a rather explorative policy for sampling, but gather information for a data packet routing policy that is more exploitative or even radically different (e.g. in the case of differentiated QoS routing [153]). Another interesting approach would be to use intermediate bootstrapping, as is done in temporal difference learning [251].

Finally, our work on AntHocNet could be extended in the direction of autonomic computing [152]. The main idea behind autonomic computing is that computer systems are getting too complex for human operators to manage, and that they should become more "self-organized". For the area of networking, this means that the network should be self-configuring, self-optimizing, self-protecting and self-healing. While these properties are to some extent already present in the adaptivity, robustness and scalability of AntHocNet, further developments would be needed to get to a fully autonomic routing algorithm. E.g., the algorithm should be able to tune its parameters, and to adapt its working automatically to different types of networks and different types of data traffic to be served. Nevertheless, AntHocNet's approach to constantly gather information about the network through sampling and pheromone diffusion gives a good basis to build such a system.

Bibliography

[1] Berlinroofnet. Available from: http://sarwiki.informatik.hu-berlin.de/BerlinRoofNet.

[2] Funkfeuer. Available from: http://www.funkfeuer.at/.

[3] IEEE 802.11 standard group website. Available from: http://www.ieee802.org/11/.

[4] IEEE 802.15 standard group website. Available from: http://www.ieee802.org/15/.

[5] IEEE 802.16 standard group website. Available from: http://www.ieee802.org/16/.

[6] The internet engineering task force mobile ad-hoc networking page (MANET). Available from: http://www.ietf.org/html.charters/manet-charter.html.

[7] Microsoft Mesh Networks. Available from: http://research.microsoft.com/mesh/.

[8] The NRL OLSR routing protocol implementation. Available from: http://pf.itd.nrl.navy.mil/olsr/.

[9] olsr.freifunk.net. Available at: http://olsrexperiment.de.

[10] Taipei-wlan. Available from: http://wlan.taipei-elife.net/english/main.html.

[11] Technical specifications and technical reports for a utran-based 3gpp system. Available from: http://www.3gpp.org/ftp/Specs/html-info/21101.htm.

[12] Wireless philadelphia. Available from: http://www.wirelessphiladelphia.org.

[13] The zigbee alliance. Available from: http://www.zigbee.org/en/index.asp.

[14] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2:1–22, 2004.

[15] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *SIGCOMM Computer Commununication Review*, 34(4):121–132, 2004.

[16] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks Journal*, 47:445–487, March 2005.

[17] I. F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–116, August 2002.

[18] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, December 2004.

[19] M. Ali, U. Saif, A. Dunkels, T. Voigt, K. Römer, K. Langendoen, J. Polastre, and Z. A. Uzmi. Medium access control issues in sensor networks. *ACM SIGCOMM Computer Communication Review*, 36(2), April 2006.

[20] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Banff, Canada, October 2001.

[21] AWE Communications. *WinProp software suite.*

[22] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proceedings of ACM SIGCOMM '96*, August 1996.

[23] B. Baran and R. Sosa. A new approach for AntNet routing. In *Proceedings of the 9th International Conference on Computer Communications Networks*, Las Vegas, USA, 2000.

[24] J. S. Baras and H. Mehta. A probabilistic emergent routing algorithm for mobile ad hoc networks. In *WiOpt03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[25] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man and Cybernetics*, SMC-13:834–846, 1983.

[26] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[27] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[28] B. Bensaou, Y. Wang, and C. C. Ko. Fair medium access in 802.11 based wireless ad hoc networks. In *Proceedings of Mobile Ad Hoc Networking and Computing (MobiHoc 2000)*, pages 99–106, 2000.

[29] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, 2003.

[30] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of Mobicom*, August 2005.

[31] L. Blazevic, L. Buttyan, S. Capkun, S. Giordano, J.-P. Hubaux, and J.-Y. Le Boudec. Self-organization in mobile ad-hoc networks: the approach of terminodes. *IEEE Communications Magazine*, 39(6), June 2001.

[32] L. Blazevic, S. Giordano, and J.-Y. Le Boudec. Anchored path discovery in terminode routing. In *Proceedings of The Second IFIP-TC6 Networking Conference (Networking 2002)*, May 2002.

[33] Bluetooth Special Interest Group. Specification of bluetooth system, February 2002. Version 1.1.

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

[35] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Theraulaz. Routing in telecommunication networks with "Smart" ant-like agents. In *Proceedings of IATA'98, Second Int. Workshop on Intelligent Agents for Telecommunication Applications*, volume 1437 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 1998.

[36] E. Borgia, M. Conti, F. Delmastro, and E. Gregori. Experimental comparison of routing and middleware solutions for mobile ad hoc networks: legacy vs cross-layer approach. In *Proceedings of the ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND)*, August 2005.

[37] N. Boulicault, G. Chelius, and E. Fleury. Experiments of ana4: An implementation of a 2.5 framework for deploying real multi-hop ad hoc and mesh networks. In *Proceedings of the IEEE ICPS Workshop on Multi-hop Ad hoc Networks: from theory to reality*, Santorini, Greece, July 2005.

[38] N. Boulicault, G. Chelius, and E. Fleury. Ana4: a 2.5 framework for deploying real multi-hop ad hoc and mesh networks. *Ad Hoc & Sensor Wireless Networks: an International Journal (AHSWN)*, 2006. To appear.

[39] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro,

and J. Alspector, editors, *Advances in Neural Information Processing Systems 6 (NIPS6)*, pages 671–678. Morgan Kaufmann, San Francisco, CA, USA, 1994.

[40] R. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: an overview. Request For Comments (RFC) 1633, Network Working Group, 1994.

[41] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proceedings of the First Workshop on Sensor Networks and Applications (WSNA)*, Atlanta, GA, USA, October 2002.

[42] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1998.

[43] T. Bu, L. Gao, and D. Towsley. On characterizing bgp routing table growth. In *Proceedings of the IEEE Global Internet Symposium*, Taipei, Taiwan, November 2002.

[44] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. Delay-tolerant networking: An approach to interplanetary internet. *IEEE Communications Magazine*, June 2003.

[45] D. Câmara and A.A.F. Loureiro. Gps/ant-like routing in ad hoc networks. *Telecommunication Systems*, 18(1–3):85–100, 2001.

[46] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2002.

[47] S. Capkun, M. Hamdi, and J.-P. Hubaux. Gps-free positioning in mobile ad-hoc networks. In *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS)*, January 2001.

[48] S. Carl-Mitchell and J. S. Quarterman. *Using ARP to Implement Transparent Subnet Gateways*. Network Working Group, October 1987. RFC 1027.

[49] G. Di Caro, F. Ducatelle, and L. M. Gambardella. Demonstrator for mobile ad hoc networks. *Internal Deliverable D12-D13 of FET Project BISON (IST-2001-38923)*, 2004.

[50] L. Carrillo, C. Guadall, J. L. Marzo, G. Di Caro, F. Ducatelle, and L. M. Gambardella. Differentiated quality of service scheme based on the use of multi-classes of ant-like mobile agents. In *CoNEXT'05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*, Toulouse, France, 2005.

[51] L. Carrillo, J.L. Marzo, D. Harle, and P. Vilà. A review of scalability and its application in the evaluation of the scalability measure of antnet routing. In C.E. Palau Salvador, editor, *Proceedings of the IASTED Conference on Communication Systems and Networks (CSN'03)*, pages 317–323. ACTA Press, 2003.

[52] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *Proceedings of IEEE Infocom*, Hong Kong, China, March 2004.

[53] R. Castaneda and S. R. Das. Query localization techniques for on-demand routing protocols in ad hoc networks. In *Proceedings of the ACM/IEEE MobiCom*, pages 186–194, Seattle, WA, USA, August 1999.

[54] D. Cavin, Y. Sasson, and A. Schiper. On the accuracy of manet simulators. In *Proceedings of the Workshop on Principles of Mobile Computing (POMC)*, 2002.

[55] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. In *Proceedings of 25th IEEE Conference on Computer Communications (INFOCOM)*, April 2006.

[56] I. D. Chakeres and E. M. Belding-Royer. The utility of hello messages for determining link connectivity. In *Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Honolulu, HI, USA, October 2002.

[57] I. D. Chakeres and E. M. Belding-Royer. Aodv implementation design and performance evaluation. 2/3, 2005.

[58] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves. A loop-free extended bellman-ford routing protocol without bouncing effect. In *Proceedings of the Symposium on Communications architectures & protocols (SIGCOMM)*, pages 224–236, 1989.

[59] C.-C. Chiang, H. Wu, W. Liu, and M. Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. In *Proceedings of IEEE SICON'97*, pages 197–211, April 1997.

[60] T. Clausen and P. Jacquet. *Optimized Link State Routing Protocol (OLSR)*. IETF, October 2003. RFC 3626.

[61] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proceedings of IEEE INMIC*, 2001.

[62] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations. Network Working Group, Request for Comments 2501, January 1999. Available from: `http://www.ietf.org/rfc/rfc2501.txt`.

[63] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305, 1997.

[64] M. Daneshtalab, A. A. Kusha, A. Sobhani, Z. Navabi, M. D. Mottaghi, and O. Fatemi. Ant colony based routing architecture for minimizing hot spots in NoCs. In *Proceedings of the 19th annual symposium on Integrated circuits and systems design (SBCCI)*, pages 56–61, Ouro Preto, MG, Brazil, 2006.

[65] S. R. Das, C. E. Perkins, and E. M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, March 2000.

[66] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th ACM International Conference on Mobile Computing and Networking (MobiCom)*, September 2003.

[67] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris. Performance of multihop wireless networks: Shortest path is not enough. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*. ACM SIGCOMM, 2002.

[68] Christian de Waal. Bonnmotion: A mobility scenario generation and analysis tool, 2002. Available from: `http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/`.

[69] S. Desilva and S. R. Das. Experimental evaluation of a wireless ad hoc network. In *Proceedings of the 9th International Conference on Computer Communications and Networks (IC3N)*, 2000.

[70] G. Di Caro. *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[71] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.

[72] G. Di Caro and M. Dorigo. Two ant colony algorithms for best-effort routing in datagram networks. In *Proceedings of the Tenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS'98)*, pages 541–546. IASTED/ACTA Press, 1998.

[73] G. Di Caro, F. Ducatelle, and L. M. Gambardella. AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. In *Proceedings of Parallel Problem Solving from Nature (PPSN) VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 461–470. Springer-Verlag, 2004.

[74] G. Di Caro, F. Ducatelle, and L. M. Gambardella. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications (ETT)*, 16(5), 2005.

[75] G. Di Caro, F. Ducatelle, and L. M. Gambardella. Swarm intelligence for routing in mobile ad hoc networks. In *Proceedings of the 2005 IEEE Swarm Intelligence Symposium (SIS)*, June 2005.

[76] G. Di Caro, F. Ducatelle, and L. M. Gambardella. *Reflecting Interfaces: The Complex Coevolution of Information Technology Ecosystems*, chapter Theory and practice of Ant Colony Optimization for routing in dynamic telecommunications networks. Idea Group Inc., 2008. To appear.

[77] G. Di Caro, F. Ducatelle, P. Heegarden, M. Jelasity, R. Montemanni, and A. Montresor. Evaluation of basic services in ad-hoc, peer-to-peer and grid networks. *Internal Deliverable D7 of FET Project BISON (IST-2001-38923)*, 2004.

[78] G. Di Caro and T. Vasilakos. Ant-SELA: Ant-agents and stochastic automata learn adaptive routing tables for QoS routing in ATM networks. *ANTS'2000 - From Ant Colonies to Artificial Ants: Second International Workshop on Ant Colony Optimization*, Brussels (Belgium), September 8-9, 2000.

[79] T. Dinh Dang, B. Sonkoly, and S. Molnar. Fractal analysis and modeling of voip traffic. In *Proceedings of the 11th International Telecommunications Network Strategy and Planning Symposium (NETWORKS 2004)*, June 2004.

[80] S. Doi and M. Yamamura. Bntnetl and its evaluation on a situation of congestion. *Electronics and Communications in Japan (Part I: Communications)*, 85(9):31–41, April 2002.

[81] Y. Dong, D. Makrakis, and T. Sullivan. Effective admission control in multihop mobile ad hoc networks. In *Proc. of the International Conference on Communication Technology (ICCT)*, 2003.

[82] M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano, IT, 1992.

[83] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[84] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[85] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 26(1):29–41, 1996.

[86] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*. Kluwer Academic Publishers, 2003.

[87] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[88] O. Dousse, F. Baccelli, and P. Thiran. Impact of interferences on connectivity in ad hoc networks. In *Proceedings of IEEE INFOCOM 2003*, San Francisco, April 2003.

[89] O. Dousse, P. Thiran, and M. Hasler. Connectivity in ad-hoc and hybrid networks. In *Proceedings of IEEE INFOCOM 2002*, pages 1079–1088, New York, June 2002.

[90] J. Dowling, E. Curran, R. Cunningham, and V. Cahill. Using feedback in collaborative reinforcement learning to adaptively optimize manet routing. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(3), May 2005.

[91] K. Doya, K. Samejima, K. Katagiri, and K. Kawato. Multiple model-based reinforcement learning. *Neural Computing*, 14(6):1347–1369, 2002.

[92] R. Draves, J. Padhye, and B. Zill. The architecture of the link quality source routing protocol. Technical report MSR-TR-2004-57, Microsoft Research, 2004.

[93] R. Draves, J. Padhye, and B. Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proceedings of ACM SIGCOMM*, pages 133–144, Portland, Oregon, USA, 2004.

[94] F. Ducatelle, G. Di Caro, and L. M. Gambardella. Ant agents for hybrid multipath routing in mobile ad hoc networks. In *Proceedings of the Second Annual Conference on Wireless On demand Network Systems and Services (WONS)*, St. Moritz, Switzerland, January 18–19 2005.

[95] F. Ducatelle, G. Di Caro, and L. M. Gambardella. Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications (IJCIA)*, 5(2), 2005.

[96] F. Ducatelle, G. Di Caro, and L. M. Gambardella. An analysis of the different components of the anthocnet routing algorithm. In *Proceedings of the Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS)*, September 2006.

[97] F. Ducatelle, G. Di Caro, and L. M. Gambardella. A study on the use of MANETs in urban environments. Technical Report IDSIA-01-07, Istituto Dalle Molle di Studi Sull'Intelligenza Artificiale (IDSIA), 2007.

[98] F. Ducatelle, L. M. Gambardella, M. Kurant, H. X. Nguyen, and P. Thiran. *Dependable Systems: Software, Computing, Networks*, volume 4028 of *Lecture Notes in Computer Science*, chapter Algorithms for failure detection in large IP-over-Fiber and Wireless Ad Hoc Networks. Springer, 2006.

[99] F. Ducatelle and J. Levine. Ant colony optimisation for bin packing and cutting stock problems. In *Proceedings of the UK Workshop on Computational Intelligence*, Edinburgh, UK, September 2001.

[100] F. Ducatelle and M. Roth. Documentation for the maganta routing package. Technical report, Deutsche Telekom Laboratories, July 2006.

[101] F. Ducatelle, M. Roth, and L. M. Gambardella. Design of a user space software suite for probabilistic routing in ad-hoc networks. In *Proceedings of the Fourth European Workshop on the application of Nature-inspired techniques to Telecommunication Networks and other Connected Systems (EvoCOMNET2007)*, Valencia, Spain, April 2007. Poster presentation.

[102] L. M. Feeney. A taxonomy for routing protocols in mobile ad hoc networks. Technical Report T99-07, Swedish Institute of Computer Science, 1 1999.

[103] L. Ford and D. Fulkerson. *Flows in Networks*. Prentice-Hall, 1962.

[104] K. Fujita, A. Saito, T. Matsui, and H. Matsuo. An adaptive ant-based routing algorithm used routing history in dynamic networks. In *Proc. of the 4th Asia-Pacific Conf. on Simulated Evolution and Learning*, 2002.

[105] L. M. Gambardella and M. Dorigo. Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of the Twelfth International Conference on Machine Learning, ML-95*, pages 252–260. Palo Alto, CA: Morgan Kaufmann, 1995.

[106] L. M. Gambardella, E. Taillard, and G. Agazzi. Ant colonies for vehicle routing problems. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. London,UK: McGraw-Hill, 1999.

[107] L. M. Gambardella, E. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society (JORS)*, 50(2):167–176, 1999. Also TR-IDSIA-4-97.

[108] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1(2), 2002.

[109] Y. Ganjali and A. Keshavarzian. Load balancing in ad hoc networks: Single-path routing vs. multi-path routing. In *Proceedings of IEEE INFOCOM*, March 2004.

[110] M. Gast. *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, Inc., 2 edition, April 2005.

[111] M. Gerla, K. Tang, and R. Bagrodia. TCP performance in wireless multihop networks. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 41–50, 1999.

[112] B. Ghribi and L. Logrippo. Understanding gprs: The gsm packet radio service. *Computer Networks*, 34:763–779, 2000.

[113] S. Giordano, I. Stojmenovic, and L. Blazevic. *Ad Hoc Wireless Networking*, chapter Position based routing algorithms for ad hoc networks: A taxonomy. Kluwer, 2003.

[114] I. Glauche, W. Krause, R. Sollacher, and M. Greiner. Continuum percolation of wireless ad hoc communication networks. *Physica A*, 325:577–600, 2003.

[115] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.

[116] P. P. Grassé. La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes sp.* La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[117] R. S. Gray, D. Kotz, C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, S. McGrath, and Y. Yuan. Outdoor experimental comparison of four ad hoc routing algorithms. In *Proceedings of the ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 220–229, October 2004.

[118] M. Grossglauser and D. N. C. Tse. Mobility increases the capacity of ad-hoc wireless networks. In *Proceedings of INFOCOM*, 2001.

[119] M. Günes, U. Sorges, and I. Bouazizi. ARA - The ant-colony based routing algorithm for MANETs. In *Proceedings of the ICPP International Workshop on Ad Hoc Networks (IWAHN)*, 2002.

[120] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, March 2000.

[121] Z. J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications*, 1997.

[122] J. F. Hayes and T. V. J. Ganesh Babu. *Modeling and Analysis of Telecommunications Networks*. John Wiley & Sons, Inc., 2004.

[123] C. Hedrick. *Routing Information Protocol*. IETF, 1988. RFC 1058.

[124] J. Heidemann, N. Bulusu, J. Elson, C. Intanagowiwat, K. Lan, Y. Xu, W. Ye, D. Estrin, and R. Govindan. Effects of detail in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*, pages 3–11, January 2001.

[125] W. Heinzelman, A. Chandrakassan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS)*, January 2000.

[126] M. Heissenbüttel and T. Braun. Ants-based routing in large scale mobile ad-hoc networks. In *Kommunikation in verteilten Systemen (KiVS03)*, March 2003.

[127] M. Heissenbüttel, T. Braun, D. Jörg, and T. Huber. A framework for routing in large ad-hoc networks with irregular topologies. *International Journal of Ad Hoc & Sensor Wireless Networks, Special Issue on the Fourth Annual Mediterranean Ad Hoc Networking Workshop*, 2(2), June 2006.

[128] M. Heusse, D. Snyers, S. Guérin, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. *Advances in Complex Systems*, 1(2):237–254, 1998.

[129] B. Hoffmann-Wellenhof, H. Lichtenegger, and J. Collins. *GPS: Theory and Practice*. Springer-Verlag, 2001.

[130] G. Holland and N. H. Vaidya. Analysis of TCP performance over mobile ad hoc networks. In *Proceedings of IEEE/ACM MOBICOM '99*, pages 219–230, August 1999.

[131] B. Hölldobler and E.O. Wilson. *The Ants*. Springer-Verlag, Berlin, Germany, 1990.

[132] E. Huang, W. Hu, J. Crowcroft, and I. Wassell. Towards commercial mobile ad hoc network applications: A radio dispatch system. In *Proceedings of the sixth ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, May 2005.

[133] P. Hui, A. Chaintreau, R. Gass, J. Scott, J. Crowcroft, and C. Diot. Pocket switched networking: Challenges, feasibility and implementation issues. In *Proceedings of the Second IFIP Workshop on Autonomic Communications*, October 2005.

[134] G. Huston. Analyzing the internet's BGP routing table. *Internet Protocol Journal*, 4(1), March 2001.

[135] IEEE 802.11 working group. ANSI/IEEE std. 802.11, 1999 edition: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Technical report, ANSI/IEEE, 1999.

[136] R. G. Ingalls. Introduction to simulation. In *Proceedings of the 2002 Winter Simulation Conference*, 2002.

[137] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networks (Mobicom)*, pages 56–67, Boston, MA, USA, 2000.

[138] A. Jardosh, E. M. Belding-Royer, K. C. Almeroth, and S. Suri. Towards realistic mobility models for mobile ad hoc networks. In *Proceedings of MobiCom*, 2003.

[139] W. Jiang and H. Schulzrinne. Analysis of on-off patterns in voip and their effect on voicetraffic aggregation. In *Proceedings of the Ninth International Conference on Computer Communications and Networks*, 2000.

[140] D. B. Johnson and D. A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks. Kluwer, 1996.

[141] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J. C. Chen. A survey of energy efficient network protocols for wireless networks. *Wireless Networks*, 7:343–358, 2001.

[142] J. Jubin and J. D. Tornow. The darpa packet radio network protocols. *Proceedings of the IEEE*, 1987.

[143] J. Jun, P. Peddabachagari, and M.L. Sichitiu. Theoretical maximum throughput of IEEE 802.11 and its applications. In *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA)*, April 2003.

[144] J. Kadlecsik, H. Welte, J. Morris, M. Boucher, and R. Russel. The netfilter/iptables project. `http://www.netfilter.org`.

[145] R. E. Kahn, S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman. Advances in packet radio technology. *Proceedings of the IEEE*, 1978.

[146] B. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00)*, 2000.

[147] R. Karrer, I. Matyasovszki, A. Botta, and A. Pescape. Experimental evaluation and characterization of the magnets wireless backbone. In *Proceedings of the 1st ACM International Workshop on Wireless Network Testbeds, Experimental evaluation and CHaracterization (WiNTECH)*, Los Angeles, CA, USA, September 2006.

[148] R. Karrer, P. Zerfos, and N. Piratla. Magnets - a next generation access network. In *Proceedings of IEEE INFOCOM*, April 2006.

[149] I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks II, P. Arabshahi, and A. A. Gray. Swarm intelligence for routing in communication networks. In *Proceedings of the IEEE World Congress on Computational Intelligence, Hawaii, May 12–17 2002*, 2002.

[150] V. Kawadia and P. R. Kumar. Power control and clustering in ad hoc networks. 2003.

[151] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgano Kaufmann, 2001.

[152] J. Kephart and D. Chess. The vision of autonomic computing. *IEEE Computer magazine*, pages 41–50, January 2003.

[153] K. Kilkki. *Differentiated Services for the Internet*. Macmillan Technology Series. Sams, 1999.

[154] Y. B. Ko and N. H. Vaidya. Flooding-based geocasting protocols for mobile ad hoc networks. *Mobile Networks and Applications*, 7(6):471–480, 2002.

[155] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3), 2000.

[156] B. Krishnamachari, D. Estrin, and S. Wicker. Modelling data-centric routing in wireless sensor networks. In *Proceedings of IEEE Infocom*, 2002.

[157] F. Kuhn, R. Wattenhofer, and A. Zollinger. Worst-case optimal and average-case efficient geometric ad-hoc routing. Technical report, Distributed Computing Group, ETH Zürich, December 2002.

[158] J. Kulik, W. Heinzelman, and H. Balakrishnan. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Seattle, WA, USA, August 1999.

[159] S. Kumar, V. S. Raghavan, and J. Deng. Medium access control protocols for ad-hoc wireless networks: A survey. *Elsevier Ad-Hoc Networks Journal*, 4(3):326–358, May 2006.

[160] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *Proceedings of ACM SIGCOMM*, pages 175–187, Stockholm, Sweden, 2000.

[161] K. Langendoen and G. Halkes. *Embedded Systems Handbook*, chapter Energy-Efficient Medium Access Control. CRC Press, August 2005.

[162] S.-J. Lee, E. M. Belding-Royer, and C. E. Perkins. Ad hoc on-demand distance-vector routing scalability. *ACM SIGMOBILE Mobile Computing and Communications Review*, July 2002.

[163] S.-J. Lee and M. Gerla. AODV-BR: Backup routing in ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2000.

[164] S.-J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of IEEE ICC*, 2001.

[165] S.-J. Lee, E. M. Royer, and C. E. Perkins. Scalability study of the ad hoc on-demand distance vector routing protocol. *ACM/Wiley International Journal of Network Management*, 13(2):97–114, 2003.

[166] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Proceedings of IEEE INFOCOM 2000*, March 2000.

[167] W. C. Y. Lee. *Mobile Communications Engineering: Theory and Applications*. McGraw Hill Professional, 1997.

[168] G. Leguizamón and Z. Michalewicz. A new version of Ant System for subset problems. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1459–1464. IEEE Press, Piscataway, NJ, USA, 1999.

[169] J. Levine and F. Ducatelle. Ant colony optimisation and local search for bin packing and cutting stock problems. *Journal of the Operational Research Society, Special Issue on Local Search*, 55(7), July 2004.

[170] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *Proceedings of the 7th ACM International Conference on Mobile Computing and Networking (Mobicom)*, pages 61–69, July 2001.

[171] J. Li, J. Jannotti, D. De Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad-hoc routing. In *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 120–130, August 2000.

[172] B. Liang and Z. J. Haas. Predictive distance-based mobility management for PCS networks. In *Proceedings of Annual IEEE Conference on Computer Communications (INFOCOM)*, 1999.

[173] S. Liang, A. N. Zincir-Heywood, and M. I. Heywood. Adding more intelligence to the network routing problem: Antnet and ga-agents. *Applied Soft Computing*, 2006.

[174] F. Lilieblad, O. Mattsson, P. Nylund, D. Ouchterlony, and A. Roxenhag. *Mad-Hoc AODV Implementation and Documentation*. `http://mad-hoc.flyinglinux.net`.

[175] J. Liu, L. F. Perrone, Y. Yuan, and D. Nicol. *The simulator for wireless ad hoc networks (SWAN)*. Bucknell University, 2005. Available from: `http://www.eg.bucknell.edu/swan`.

[176] L. Liu and G. Feng. A novel ant colony based QoS-aware routing algorithm for MANETs. In *Proceedings of the First International Conference on advances in Natural Computation (ICNC)*, pages 457–466, August 2005.

[177] Z. Liu, M. Kwiatkowska, and C. Constantinou. A self-organised emergent routing mechanism for mobile ad hoc networks. *European Transactions on Telecommunications (ETT)*, 16(5):457–470, 2005.

[178] K. Lougheed and Y. Rekhter. *A border gateway protocol*. IETF, June 1990. RFC 1163.

[179] A. Lozano, F. R. Farrokhi, and R. A. Valenzuela. Lifting the limits on high-speed wireless data access using antenna arrays. *IEEE Communications Magazine*, 39:156–162, 2001.

[180] Y. Lu, W. Wang, Y. Zhong, and B. Bhargava. Study of distance vector routing protocols for mobile ad hoc networks. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, 2003.

[181] H. Lundgren, D. Lundberg, J. Nielsen, E. Nordström, and C. Tschudin. A large-scale testbed for reproducible ad hoc protocol evaluations. In *3rd annual IEEE Wireless Communications and Networking Conference (WCNC 2002)*, pages 412–418. IEEE, March 2002.

[182] G. S. Malkin and M. E. Steenstrup. Distance-vector routing. In M. E. Steenstrup, editor, *Routing in Communications Networks*, chapter 3, pages 83–98. Prentice-Hall, 1995.

[183] D. A. Maltz, J. Broch, J. Jetcheva, and D. B. Johnson. The effects of on-demand behavior in routing protocols for multi-hop wireless ad hoc networks. *IEEE Journal on Selected Areas in Communications, special issue on mobile and wireless networks*, August 1999.

[184] V. Maniezzo, A. Colorni, and M. Dorigo. The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Université Libre de Bruxelles, Belgium, 1994.

[185] R.W. Mankin, R.T. Arbogast, P.E. Kendra, and D.K. Weaver. Active spaces of pheromone traps for *Plodia interpunctella* in enclosed environments. *Environmental Entomology*, 28(4):557–565, 1999.

[186] M. Marina and S. Das. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 14–23, 2001.

[187] M. Marina and S. Das. Routing performance in the presence of unidirectional links in multihop wireless networks. In *Proceedings of ACM MobiHoc*, pages 12–23, 2002.

[188] S. Marinoni and H. H. Kari. Ad hoc routing protocol performance in a realistic environment. In *Proceedings of IEEE ICN*, April 2006.

[189] Ian Marsh, Fengyi Li, and Gunnar Karlsson. Wide area measurements of voip quality. In *Proceedings of the 4th International Workshop on Quality of Future Internet Services*, 2003.

[190] S. Marwaha, C. K. Tham, and D. Srinivasan. Mobile agents based routing protocol for mobile ad hoc networks. In *Proc. of IEEE Globecom*, 2002.

[191] H. Matsuo and K. Mori. Accelerated ants routing in dynamic networks. In *2nd Int. Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2001.

[192] T. Michalareas and L. Sacks. Stigmergic techniques for solving multiconstraint routing for packet networks. In P. Lorenz, editor, *Networking - ICN 2001, Proceedings of the First International Conference on Networking, Part II, Colmar, France July 9-13, 2001*, volume 2094 of *Lecture Notes in Computer Science*, pages 687–697. Springer-Verlag, 2001.

[193] R. Michel and M. Middendorf. An ACO algorithm for the shortest supersequence problem. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 51–61. London,UK: McGraw-Hill, 1999.

[194] N. Minar, K. H. Kramer, and P. Maes. Cooperating mobile agents for dynamic network routing. In Alex Hayzelden and John Bigham, editors, *Software Agents for Future Communication Systems*, chapter 12. Springer-Verlag, 1999.

[195] J. P. Monks, V. Bhargavan, and W.-M. Hwu. A power controlled multiple access protocol for wireless packet networks. pages 219–228, 2001.

[196] R. Montemanni and L. M. Gambardella. Exact algorithms for the minimum power symmetric connectivity problem in wireless networks. *Computers and Operations Research*, 32(11):2891–2904, November 2005.

[197] J. Moy. OSPF version 2. Request For Comments (RFC) 1583, Network Working Group, 1994.

[198] S. Mueller, R. Tsang, and D. Ghosal. Multipath routing in mobile ad hoc networks: Issues and challenges. In *Performance Tools and Applications to Networked Systems*, volume 2965 of *LNCS*. Springer-Verlag, 2004.

[199] M. Musuvathi, D. Y. W. Park, A. Chou, D. R. Engler, and D. L. Dill. CMC: A pragmatic approach to model checking real code. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, 2002.

[200] C. Na, J. K. Chen, and T. S. Rappaport. Measured traffic statistics and throughput of IEEE 802.11b public WLAN hotspots with three different applications. *IEEE Transactions on Wireless Communications*, 5(11), November 2006.

[201] A. Nasipuri, R. Castaneda, and S. R. Das. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *Mobile Networks and Applications*, August 2001.

[202] G. Navarro Varela and M. C. Sinclair. Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1809–1816. IEEE Press, Piscataway, NJ, USA, 1999.

[203] J. C. Navas and T. Imielinski. Geocast geographic addressing and routing. In *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking (MobiCom)*, pages 66–76, Budapest, Hungary, 1997.

[204] S. Ngo, X. Jiang, V. Le, and S. Horiguchi. Ant-based survivable routing in dynamic WDM networks with shared backup paths. *Journal of Supercomputing*, 36(3):297–307, June 2006.

[205] K. Oida and A. Kataoka. Lock-free AntNet and its evaluation for adaptiveness. *Journal of IEICE B* (in Japanese), J82-B(7):1309–1319, 1999.

[206] K. Oida and M. Sekido. ARS: an efficient agent-based routing system for QoS guarantees. *Computer communications*, 23:1437–1447, 2000.

[207] OPNET Technologies, Inc. *OPNET Users' Manual*. Avilable from: `http://www.opnet.com`.

[208] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, New Jersey, 1982.

[209] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of INFOCOM'97*, April 1997.

[210] G. Pei, M. Gerla, and T.-W. Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *Proceedings of the IEEE International Conference on Communications*, pages 70–74, June 2000.

[211] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[212] C. E. Perkins. *Ad hoc On-Demand Distance Vector (AODV) Routing*. IETF, July 2003.

[213] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[214] P. Pham and S. Perreau. Performance analysis of reactive shortest path and multi-path routing mechanism with load balance. In *Proceedings of IEEE INFOCOM*, March 2003.

[215] D. C. Plummer. *An Ethernet Address Resolution Protocol, or: Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware.* Network Working Group, November 1982. RFC 826.

[216] O. Pomerantz. *The Linux Kernel Module Programming Guide.* iUniverse Inc, 2000.

[217] J. Postel. *User Datagram Protocol.* IETF, August 1980. RFC 768.

[218] J. Postel. *Transmission Control Protocol.* IETF, September 1981. RFC 793.

[219] R. Prakash. Unidirectional links prove costly in wireless ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999.

[220] S. Rajagopalan and C.-C. Shen. ANSI: A unicast routing protocol for mobile ad hoc networks using swarm intelligence. In *Proceedings of the International Conference on Artificial Intelligence (ICAI)*, June 2005.

[221] T.S. Rappaport. *Wireless communications, principles and practice.* Prentice Hall, 1999.

[222] S. M. Redl, M. K. Weber, and M. W. Oliphant. *An Introduction to GSM.* Artech House, March 1995.

[223] E. Rosen, A. Viswanathan, and R. Callon. *Multiprotocol Label Switching Architecture.* Network Working Group, January 2001. RFC 3031.

[224] M. Roth and S. Wicker. Termite: Ad-hoc networking with stigmergy. In *Proceedings of Globecom*, 2003.

[225] M. Roth and S. Wicker. *Swarm Intelligence and Data Mining*, chapter Termite: A Swarm Intelligence Routing Algorithm for Mobile Wireless Ad-Hoc Networks. Springer, 2005.

[226] L. Rothkrantz and R. van der Put. Routing in packet switched networks using agents. In *First International Workshop on Ant Colony Optimization (ANTS)*, 1998.

[227] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999.

[228] R. Y. Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In S. Uryasev and P.M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*. Kluwer Academic Publisher, 2000.

[229] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy. PATHS: analysis of PATH duration statistics and their impact on reactive MANET routing protocols. In *Proceedings of MobiHoc'03*, pages 245–256, 2003.

[230] H. Sandalidis, K. Mavromoustakis, and P. Stavroulakis. Ant-based probabilistic routing with pheromone and antipheromone mechanisms. *International Journal of Communication Systems (IJCS)*, 17:55–62, January 2004.

[231] C.A. Santivanez, B. McDonald, I. Stavrakakis, and R. Ramanathan. On the scalability of ad hoc routing protocols. In *Proceedings of the* 21^{st} *Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom)*, 2002.

[232] Scalable Network Technologies, Inc. *QualNet Simulator, Version 3.8*, 2005. Available from: `http://www.scalable-networks.com`.

[233] H. Schiöberg. A performance evaluation framework for wireless mesh routing protocols. Master's thesis, Technische Universität München, Fakultät für Informatik, December 2006.

[234] A. Schmitz and M. Wenig. The effect of the radio wave propagation model in mobile ad hoc networks. In *Proceedings of ACM MSWiM*, October 2006.

[235] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.

[236] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb. A case study of OSPF behavior in a large enterprise network. In *Proceedings of the 2nd ACM SIGCOMM Internet Measurement Workshop (IMW)*, pages 217–230, Marseille, France, 2002.

[237] R. E. Shannon. *Systems Simulation - The Art and Science*. Prentice-Hall, 1975.

[238] P. Shirley and R. Morley Keith. *Realistic Ray Tracing*. A.K. Peters, 2001.

[239] A. Shmygelska and H. H. Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6(30), February 2005.

[240] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Proceedings of Mobicom*, Dallas, TX, USA, October 1998.

[241] V. Sridhana and S. Bohacek. Realistic propagation simulation of urban mesh networks. Technical report, University of Delaware, Department of Electrical and Computer Engineering, 2006.

[242] V. Sridhara, J. Kim, and S. Bohacek. Performance of urban mesh networks. In *Proceedings of ACM MSWiM*, October 2005.

[243] R. Steinmetz and K. Wehrle. *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Scienc*. Springer Publishing, September 2005.

[244] W. R. Stevens, B. Fenner, and A. M. Rudoff. *Unix Network Programming, Vol. 1: The Sockets Networking API*. Addison-Wesley Professional, 3 edition, October 2003.

[245] T. Stützle and H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8), 2000.

[246] J. Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara. User mobility for opportunistic ad-hoc networking. In *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, pages 41–50, 2004.

[247] M. W. Subbarao. Dynamic power-conscious routing for MANET's: an initial approach. *Journal of Research of the National Institute of Standards and Technology*, 1999.

[248] D. Subramanian, P. Druschel, and J. Chen. Ants and reinforcement learning: A case study in routing in dynamic networks. In *Proceedings of IJCAI-97, International Joint Conference on Artificial Intelligence*, pages 832–838. Morgan Kaufmann, 1997.

[249] J. Sum, H. Shen, G. Young, and J. Wu. Analysis on extended ant routing algorithms for network routing and management. *Journal of Supercomputing*, 24(3), March 2003.

[250] D. Sun and H. Man. Performance comparison of transport control protocols over mobile ad hoc networks. In *The 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication (PIMRC)*, September 2001.

[251] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[252] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[253] S. Tadrus and L. Bai. A QoS network routing algorithm using multiple pheromone tables. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*, Halifax, Canada, October.

[254] A. Tanenbaum. *Computer Networks*. Prentice-Hall, 4th edition, 2002.

[255] B. Tatomir and L. Rothkrantz. Dynamic routing in mobile wireless networks using ABC-AdHoc. In *Proceedings of the fourth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS)*, September 2004.

[256] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life,* Special Issue on Stigmergy, 5:97–116, 1999.

[257] C.-K. Toh. Associativity-based routing for ad-hoc mobile networks. *Wireless Personal Communications*, pages 1–36, March 1997.

[258] J. Tourrilhes. Wireless tools for linux. `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html`, 1996.

[259] C. Tschudin, R. Gold, O. Rensfelt, and O. Wibling. LUNAR: a lightweight underlay network ad-hoc routing protocol and implementation. In *Proceedings of Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN)*, February 2004.

[260] C. Tschudin, P. Gunningberg, H. Lundgren, and E. Nordström. Lessons from experimental MANET research. *Elsevier Ad Hoc Networks Journal*, 3(2), 2005.

[261] A. Tsirigos and Z. J. Haas. Multipath routing in the presence of frequent topological changes. *IEEE Communications Magazine*, 39, November 2001.

[262] UC Berkeley, LBL, USC/ISI, and Xerox PARC. *The ns Manual*. Available from: `http://www.isi.edu/nsnam/ns/ns-documentation.html`.

[263] UCLA Parallel Computing Laboratory. *GloMoSim Manual*, 1999. Available from: `http://pcl.cs.ucla.edu/projects/glomosim/`.

[264] V. Untz, M. Heusse, F. Rousseau, and A. Duda. Lilith: an interconnection architecture based on label switching for spontaneous edge networks. In *Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS)*, pages 146–151, August 2004.

[265] A. Varga. *OMNeT++, Discrete event simulation system, Version 2.2*. Technical University of Budapest, Department of Telecommunications, 2002. Available from: `http://www.hit.bme.hu/phd/vargaa/omnetpp.htm`.

[266] A. V. Vasilakos and G. A. Papadimitriou. A new approach to the design of reinforcement scheme for learning automata: Stochastic Estimator Learning Algorithms. *Neurocomputing*, 7(275), 1995.

[267] P. Vrancx and A. Nowé. Using pheromone repulsion to find disjoint paths. In *Fifth international workshop on ant colony optimization and swarm intelligence (ANTS)*, 2006.

[268] F. Wang and Y. Zhang. *Ad Hoc and Sensor Networks*, chapter A Survey on TCP over Mobile Ad-Hoc Networks, pages 267–281. Nova Science Publishers, 2005.

[269] L. Wang, Y. T. Shu, O. W. W. Yang, M. Dong, and L. F. Zhang. Adaptive multipath source routing in wireless ad hoc networks. In *Proc. of the IEEE Int. Conf. on Communications*, 2001.

[270] L. Wang, Z. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S.F. Wu, and L. Zhang. Observation and analysis of bgp behavior under stress. In *Proceedings of the* 2^{nd} *ACM Workshop on Internet Measurement*, 2002.

[271] C. J. Watkins. *Learning with Delayed Rewards*. PhD thesis, Psychology Department, University of Cambridge, UK, 1989.

[272] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[273] H. F. Wedde, M. Farooq, and Y. Zhang. BeeHive: An efficient fault tolerant routing algorithm under high loads inspired by honey bee behavior. In M. Dorigo, M. Birattari, L. Gambardella, F. Mondada, and T. Stützle, editors, *Ants Algorithms - Proceedings of ANTS 2004, Fourth International Workshop on Ant Algorithms*, Lecture Notes in Computer Science. Springer-Verlag, 2004. To appear.

[274] T. White, B. Pagurek, and F. Oppacher. Connection management using adaptive mobile agents. In H.R. Arabnia, editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98)*, pages 802–809. CSREA Press, 1998.

[275] O. Wittner and B. E. Helvik. Cross entropy guided ant-like agents finding dependable primary/backup path patterns in networks. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, Honolulu, HI, USA, May 2002.

[276] O. Wittner and B. E. Helvik. Ce-ants: Ant-like agents for path management in the next-generation internet. *Ercim News*, (64):31–32, January 2006.

[277] O. Wittner, B. E. Helvik, and V. Nicola. Internet failure protection using hamiltonian p-cycles found by ant-like agents. In *Proceedings of The 5th International Workshop on Design of Reliable Communication Networks (DRCN)*, Island of Ischia (Naples), Italy, October 2005.

[278] K. Wu and J. Harms. On-demand multipath routing for mobile ad hoc networks. In *Proceedings of EPMCC*, 2001.

[279] J. Xie, R. R. Talpade, A. McAuley, and M. Liu. Amroute: Ad hoc multicast routing protocol. *Mobile Networks and Applications*, 7:429–439, 2002.

[280] S. Xu and T. Saadawi. Does the IEEE 802.11 mac protocol work well in multihop wireless ad hoc networks? *IEEE Communications Magazine*, 111:130–137, June 2001.

[281] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad-hoc routing. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, pages 74–80, 2001.

[282] L. Yang and G. B. Giannakis. Ultra-wideband communications: An idea whose time has come. *IEEE Signal Processing Magazine*, 21(6):26–54, November 2004.

[283] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. A framework for reliable routing in mobile ad hoc networks. In *Proceedings of IEEE INFOCOM*, 2003.

[284] S. Yi, Y. Pei, and S. Kalyanaraman. On the capacity improvement of ad hoc wireless networks using directional antennas. In *Proceedings of the fourth ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Annapolis, MD, USA, June 2003.

[285] Y. Yi, M. Gerla, and T.-J. Kwon. Efficient flooding in ad hoc networks: a comparative performance study. In *Proceedings of IEEE ICC*, 2003.

[286] L. Yong, Z. Guang-zhou, S. Fan-jun, and L. Xiao-run. Adaptive swarm-based routing in communication networks. *Journal of Zhejiang University SCIENCE*, 5(7):867–872, 2004.

[287] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *Proceedings of IEEE INFOCOM*, 2003.

[288] Y. Zhang, L. D. Kuhn, and M. P. J. Fromherz. Improvements on ant routing for sensor networks. In *Proceedings of the fourth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS)*, September 2004.

[289] X. Zheng, W. Guo, and R. Liu. An ant-based distributed routing algorithm for ad-hoc networks. In *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS)*, pages 412–417, June 2004.

[290] G. Zhou, T. He, and J. A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.