Protocol Design and Implementation for Bee-Inspired Routing in Mobile Ad Hoc Networks

Alexandros Giagkos

Department of Computer Science
Aberystwyth University
Aberystwyth

April
2012

This thesis is submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy of Aberystwyth University.

Supervised by:

Dr Myra S. Wilson
Research Supervisor
Lecturer, Department of Computer Science
Aberystwyth University
Wales, UK

Doctoral commitee (June 27, 2012):

Dr Bernie Tiddeman
Chair
Senior Lecturer, Department of Computer Science
Aberystwyth University
Wales, UK

Examined and approved by:

Dr Neal Snooke
Internal Examiner
Lecturer, Department of Computer Science
Aberystwyth University
Wales, UK

Prof. Alan FT Winfield
External Examiner
Hewlett-Packard Professor of Electronic Engineering
University of the West of England, Bristol
England, UK

To my father Γιάννης

Abstract

The characteristic of mobility and the ease of deployment make wireless ad hoc networks suitable for a variety of real life applications that cover a wide range from civilian to military purposes. The lack of a fixed infrastructure demands all participating nodes to function as end points of a communication session and also to have routing capabilities. The latter allows data packets to be forwarded to nodes in a multi-hop manner and tackles the routing problem when nodes are joining, leaving or moving around within the network topology unexpectedly. At any time nodes need to be able to provide adaptive, optimal and efficient routing solutions.

In order to solve the challenging problem of routing in wireless ad hoc networks, this thesis applies methods from nature and, in particular, from the world of honeybee colonies. A new routing protocol design and its implementation, BeeIP, are proposed and tested. Using honeybee foraging and dancing metaphors, the protocol utilizes special packets to discover paths between sources and destinations. Real honeybees constantly monitor the goodness of their findings based on a number of quality factors such as the distance from the hive, the sweetness of the sugar solution, etc. Then, they efficiently distribute the future flights following the most optimal path. Focusing on these key concepts, this work investigates the extent to which a range of low-level network parameters can be used to represent and constantly monitor the goodness of the paths. The design uses a new model to map the honeybee dances and to efficiently use multiple paths for future data transmissions.

This thesis makes a number of novel contributions. Firstly, an extended mapping of the quality factors from nature to networks and a model to utilize them in order to represent and measure the quality of the paths. Next, the use of statistical prediction by considering prior gathered knowledge to detect any possible improvement or deterioration of path quality over time. Finally, a comprehensive comparison with state-of-the-art protocols in the ns-2 network simulator, the results of which show that BeeIP is able to outperform the others under different conditions and, in particular, in networks of high density, rate of mobility and increased data traffic. Therefore, the proposed design is a viable solution for routing in wireless ad hoc networks.

Acknowledgements

First and foremost I would like to thank my supervisor Dr Myra S. Wilson for her constant support, both academic and psychological, as well as her solid guidance throughout this work. I appreciate all her contributions of time, ideas and inspirational talks to make my Ph.D. experience productive and stimulating. It has been an honour to be her student.

Then, I would like to thank Mr David E. Price, my second supervisor, for his ideas and the valuable debates that we had during our long lasting meetings. His joy and enthusiasm for this particular field in computing has always been contagious and motivational for me.

I am also thankful to a number of other people starting with the examiners Prof. Alan FT Winfield and Dr Neal Snooke for their valuable remarks, suggestions and approval of this thesis. Dr Fred Long and Dr Edel M. Sherratt for the time they spent to discuss ideas regarding an accurate mathematical representation of the proposed model. My fellow Ph.D. students and the two mentors Prof. Mark Lee and Dr Frédéric Labrosse for their patience and feedback to my presentations and talks. Also, many thanks to Mr Huw Davies for spending time proofreading! I am also grateful to the group of people in the research office C57, both past and present members, for the friendly atmosphere, scientific input and great fun even during tough times in the Ph.D. pursuit. I also appreciate the financial support of the Computer Science department with all those job opportunities during the last many years and the Aberystwyth University's APRS funding to this research project.

Lastly, I would like to thank both my parents for all their love, encouragement, concern and strength all these years. My father, to whom this thesis is dedicated, who never stopped believing in me. My mother, who never stopped supporting and giving hope. Thank you.

Alexandros I. Giagkos
August 2012

Contents

List of Figures

List of Tables

List of Equations

List of Algorithms

List of Abbreviations

ABR	Associativity-Based Routing
ACK	Acknowledgement (packet)
ACO	Ant Colony Optimization
ACPI	Advanced Configuration and Power Interface
ADV	Adaptive Distance Vector
ANN	Artificial Neural Network
AODV	Ad hoc On-demand Distance-Vector
AOMDV	Ad hoc On-demand Multi-path Distance-Vector
ARA	Ant-colony based Routing Algorithm
BCO	Bee Colony Optimization
BeeIP	Bee Inspired Protocol
BGP	Border Gateway Protocol
CBR	Constant Bit Rate
CGSR	Cluster Head Gateway Switch Routing
CI95	95% Confidence Interval
CSI	Channel State Information
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CTS	Clear To Send
DARPA	Defence Advanced Research Projects Agency
DCF	Distributed Coordination Function

DDL	The data link layer
DSDV	Destination Sequenced Distance-Vector
DSR	Dynamic Source Routing
DYMO	Dynamic MANET On-Demand
ESR	Eyes Source Routing
FANN	Fast Artificial Neural Network (library)
FC	Fuzzy Curves
FSR	Fisheye State Routing
FTP	File Transfer Protocol
GloMoSim	Global Mobile Information Systems Simulation Library
GPS	Global Positioning System
GSR	Global State Routing
IARP	IntrA-zone Routing Protocol
ID	IDentification
IEEE	Institute of Electrical and Electronics Engineers
IERP	IntEr-zone Routing Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
LAR	Location-Aided Routing
LLC	Logical Link Control (sub-layer)
LRP	Label Routing Protocol
MAC	Media Access Control (sub-layer)
MACA	Multiple Access Collision Avoidance
MACAW	Multiple Access with Collision Avoidance for Wireless
MANET	Mobile Ad Hoc Network
MIRACLE	Multi-Interface Cross-Layer Extension

ML	Machine Learning
MLP	Multi-Layer Perceptron
MRP	Multipoint Relay
MSE	Mean Square Error
NeST	Network STatus
NPDU	Network Protocol Data Unit
NSF	National Science Foundation
OLSR	Optimized Link State Routing
OMNeT++	Objective Modular Network Testbed in C++
OPNET	Optimized Network Engineering Tools
OSI	Open Systems Interconnection (model)
PDR	Packet Delivery Ratio (performance metric)
PHY	The PHYsical layer
PRNet	Packet Radio Network
PSR	Power-aware Source Routing
PUMA	Protocol for Unified Multicasting Through Announcements
QoS	Quality Of Service
RERR	Route Error (message)
RIP	Routing Internet Protocol
RMASE	Routing Modelling Application Simulation Environment
RREP	Route Reply (message)
RREQ	Route Request (message)
RTS	Request To Send
RTT	Round Trip Time
RWP	Random WayPoint
SA	Sensitivity Analysis

SI	Swarm Intelligence
TC	Topology Control (message)
TCCA	Time-Control Clustering Algorithm
TCP	Transmission Control Protocol
TDR	Trigger-based Distributed QoS Routing
TI	Tactical Internet
TORA	Temporally-Ordered Routing Algorithm
TTL	Time To Live
UDP	User Datagram Protocol
VINT	Virtual InterNet-work Testbed
VoIP	Voice over IP
WSN	Wireless Sensor Network
WSYS	Weighting System (protocol)
ZRP	Zone Routing Protocol

Chapter 1

Introduction

The constant improvement of the technologies related to telecommunication and computer networks is one of the fastest growing aspects of people's needs. The Internet has revolutionized many aspects of daily life. In fact, it has created the user need and demand to be connected any time and anywhere. Wireless communication networks have played a critical role in fulfilling those telecommunication needs. Consequently, they have captured the attention of both academia and industry towards an endless extent of possible application domains. Making video calls from a smart phone, downloading from websites and sending e-mails while waiting in front of a flight gate, sharing important documents or purchasing goods on the Internet while travelling, are just a small set of possible everyday activities that depend on wireless communication.

Apart from everyday needs, wireless networks play a significant role in the development of highly sophisticated applications in research and industry, for both civilian and military purposes. Computer devices communicating with each other within large and heterogeneous networks (in terms of communication technologies, protocols and services) is seen everywhere in automated industrial monitoring, environmental sensing, wildlife monitoring, search and rescue operations, etc.

Wireless routing research has always been part of the evolution in wireless systems in order to effectively and efficiently absorb new features introduced by novel communication technologies. Routing can be defined as the construction and maintenance of a working path or paths between two nodes in a network, which wish to communicate with each other by exchanging data packets. In this work, attention is given to the routing problem of wireless networking and in particular those networks which have no fixed, predefined infrastructure (ad hoc). The rest of this chapter is as follows. First, the background and motivations behind this research are discussed, along with the problem definition and a brief overview of the proposed solution. The research hypothesis and question are then presented, followed by a discussion of the aims and objectives. A summary of the results as well as the scientific contributions are given next, to finally conclude with an outline of the thesis structure.

1.1 Background and motivations

Wireless ad hoc networks are those networks which have no fixed infrastructure. As opposed to the infrastructure counterparts, where wireless access points and routers are present, in an ad hoc environment there is neither an organized hierarchy nor central administration to orchestrate the data traffic between the participating devices (nodes). Rather, the routing problem is meant to be solved by the nodes themselves, in a decentralized and distributed manner.

The principle behind routing in wireless ad hoc networking is multi-hop relaying. This feature is provided by the routing protocol and offers the ability to allow data packets to be forwarded by intermediate nodes (relays), until they reach their final destination. The need for such a feature is clear: if a direct point-to-point connection between two nodes is achieved only when they are within transmission range, then intermediate nodes have to form a path (chain of point-to-point links) in order to provide a connection between remote nodes. These paths have to be formed on the fly and in an optimal manner.

Additionally, wireless ad hoc networks often experience the aspect of node mobility. Depending on the application, nodes are allowed to change their position at varying speeds and dramatically affect the network topology. This defines an important obstacle to routing, as it can suddenly cancel any prior topological configuration and planning.

There are two main wireless ad hoc networks defined in this thesis. The mobile ad hoc networks (MANETs) (Ram Murthy & Manoj, 2004) and the wireless sensor networks (WSNs) (Dargie & Poellabauer, 2010). MANETs consist of mobile nodes which communicate with each other in a decentralized and multi-hop way, exclusively using wireless links. Nodes in MANETs are equal (i.e., there is no hierarchy) and are able to provide routing solutions when asked by the application, in order to forward data packets in the correct direction within the topology. WSNs differ from MANETs in the nature of the nodes. A WSN consists of a set of sensor nodes, which are typically small and have limited resources and processing power. They may be mobile and are deployed to collect information using their sensory capabilities. When scheduled, the sensor nodes send the collected information to the source nodes in a multi-hop fashion, similar to that of MANETs. Source nodes are usually less mobile and constitute the processing units of the application.

Both MANETs and WSNs are heavily used today. Due to their flexibility, scalability and ease of deployment (in fact, they require minimal prior planning and set-up), they have great potential and offer endless possibilities to different application domains. Unsurprisingly, the current literature in routing research is very extensive. Several routing strategies, mostly inspired by the wired Internet, have been proposed and compared (Taneja & Kush, 2010; Lie & Kaiser, 2005; Layuan *et al.*, 2007; Singh *et al.*, 2011). Although the mainstream approaches met in these protocols are widely adapted as the state-of-the-art solutions, they require the routing protocol to forward data packets to the next hop in the path based on topological information collected by non-intelligent mechanisms. While the user needs change and increase, the cost of such approaches in terms of resource and processing power is also increased. Due to this fact, the research community has turned its attention to a different approach, proposing agent-based networking systems (Hayzelden & Bigham, 1999; Marwaha *et al.*, 2002). The principle difference between the traditional and agent-based routing is that in the latter, computation moves from one hop to the other instead of just the attendant routing information (Minar *et al.*, 1999). A traditional approach wants routing data to be exchanged between all nodes in the topology, which is then used to solve the problem individually. Agent-based routing defines a set of principle rules that all nodes are bound to follow, which allows routing computation to become a product of collaboration between the nodes. In that way, the complex problem of routing can be solved more easily, reducing the computational and resource costs.

Ways of providing solutions to complex problems using agent-based systems are commonly found in nature (Decastro *et al.*, 2004). In fact, examples of solving metaphors of the routing problem are met in insect societies. Ants and honeybees are not only able to discover the shortest path between a productive source of food and their nests or hives, but also they can exchange certain information about the maintained working path with each other.

Studying nature and mapping insect behaviour to networks is currently a subject of active research, offering a number of interesting results (Karaboga & Akay, 2009; Schoonderwoerd *et al.*, 1996). Noticeable examples include AntHoc (Di Caro, 2004), where the author proposes a way of solving the routing problem in wired networks by applying principles of the ants, based on Dorigo's work in ant colony optimization (Dorigo & Stutzle, 2004). The primary objective of this protocol is to maximize the performance of the complete network by distributing the load over multiple paths, using ant-like collaborative agents. In Dorigo et al. (2005), the authors extend Di Caro's and Dorigo's work by applying the ant-inspired routing in MANETs, named AntHocNet. Ducatelle's work keeps most of the features of the original protocol and shows promising results when compared with traditional protocols (Ducatelle *et al.*, 2005).

A similar time line of events has also been followed for bee-inspired routing. In Wedde et al. (2004a), Wedde, Farooq and Zhang propose BeeHive, the first bee-inspired protocol to provide routing in wired networks. Their approach is inspired by the foraging behaviour of honeybees to explore the surroundings for interesting sources of food and to advertise their discoveries by the means of their special communication. In Wedde et al. (2005a), the authors propose the first bee-inspired routing protocol for MANETs, which keeps the principle ideas of BeeHive, but also takes into consideration the limited energy resources of the wireless ad hoc network. The result is an energy-efficient routing protocol which, compared to traditional approaches, is shown to be a strong candidate in the field, in terms of saving energy (Wedde *et al.*, 2005a).

1.1.1 Research problem

The work presented in this thesis can be considered nature-inspired, and in particular bee-inspired as it proposes a new way of designing routing protocols for MANETs based on the foraging and communicational behaviours of honeybees. Before expanding the methodology of the new proposal, it is important to briefly discuss the issues found in mobile wireless ad hoc networks.

Wireless is an unreliable medium of communication, and is of broadcast nature. Unlike wired communications where signals are physically conducted through different wires, wireless transmissions share the same medium, the air. It is due to this fact, that wireless communications have lower bandwidth, increased number of collisions (especially when there are multiple transmissions involved), and are prone to signal propagation problems such as interference. Routing in wireless networks has to be fast and efficient, and occupy the medium as little as possible.

Wireless ad hoc networks are extremely dynamic, especially when mobile nodes are present. The connections between them cannot be initially planned. The topology is subject to frequent and unexpected changes, because nodes change position, existing ones go off-line or new ones connect to the network. Therefore, the routing protocol needs to be adaptive to rapid topological changes, and provide decentralized and distributed solutions, while organizing the topology dynamically.

Finally, mobile nodes in a wireless ad hoc network tend to be small and therefore have limited resources, not only in terms of energy (carrying batteries), but also in terms of processing power and memory. As previously mentioned, this problem becomes more important in the case of WSNs, where the sensor nodes tend to be small and cheap. The computation taking place in each node needs to be minimal, and preferably agent-based. Hence, the routing protocol needs be able to provide routing solutions in an efficient way, by using the available and remaining resources in a conservative way.

To summarize, the investigated problem of this research is that: *the new design must be able to provide adaptive, optimal and efficient routing solutions, in a decentralized, distributed and self-organized manner.*

1.1.2 Nature-inspired approach

In order to solve the above problem, this work proposes a new way of designing routing protocols inspired by the collaborative foraging behaviours of honeybees.

Honeybees have been studied and their remarkable behaviours have been deciphered by Nobel laureate[1] Prof. Karl von Frisch (1886–1982). Von Frisch's studies have revealed a number of interesting aspects of the insect's lives, including the foraging procedure followed by the honeybee foragers in order to discover, evaluate, and carry food back to their hives. What has been discovered is that when something interesting is found during the discovery phase, a honeybee will collect an amount of food and fly back to the hive where she will communicate with fellow honeybees to share her information. The communication takes place in a special place within the combs, and it is achieved by a procedure termed "the bee dance".

[1]Nobel Prize in Physiology or Medicine in 1973.

Depending on the quality of the food, a forager may dance vigorously and attract more recruits. In fact, the more zealous the performance, the more attraction the site receives. Interestingly, if the quality of the finding is not satisfactory, the performing forager may decide to stop sharing information about it.

In general, a honeybee society (colony) can be seen as a natural network of agents, which shares a number of common characteristics with wireless mobile ad hoc networks and in particular, adaptive routing. To start with, a colony constitutes a large number of individual agents that move freely and communicate with each other when required. Although these agents are of the same structure (female honeybees workers), they can switch roles depending on the current needs of their hive. For instance, a honeybee worker can switch to a forager and fly off the hive in order to find food, can be a guard and protect the hive in case of an emergency, or can be a larvae caregiver. For each role, honeybees are bound to follow a set of simple yet efficient rules, and success is the product of their collaboration. Therefore, there is no central administration, and there is no need for a global knowledge of their system.

An important characteristic shared by the honeybees is their ability to find the optimal path between their hive and a productive source of food. The impact of this feature is twofold. Initially, foragers are able to discover, evaluate and constantly monitor the quality of the path to the food they have discovered, as well as the quality of the food itself. Secondly, they are efficient communicators, as they share with precision the distance, direction and quality of the discovery.

Optimality is another characteristic of their foraging. Honeybees are trying to optimize the overall energy costs of the hive, by prioritizing those findings that have a good potential by adjusting their dances. Von Frisch has shown that honeybees constantly judge the quality of that whey find, based on a variety of factors, such as the distance, the sweetness of the food, the quantity, the environmental conditions, etc.

Finally, another characteristic of the honeybee colony that matches a favourable requirement in routing protocols is the finding and using of multiple paths. In real honeybees, not all foragers of the same hive work with the same path. Rather, they are distributed to different paths which lead to different sources of food simultaneously. Distribution is achieved by trying to fulfil the need for a specific missing commodity. When several findings can satisfy the need, honeybees are distributed according to each finding quality. The better the quality, the more recruits will work on it. Multi-path discovery and usage in routing protocols is an approach which allows fewer packets to be lost or dropped due to load congestion.

Understanding and interpreting the common characteristics found in honeybee colonies and MANETs is what defines the driving force of this research work. The thesis investigates how concepts of foraging and communication between honeybees can be efficiently mapped and modelled in wireless mobile ad hoc networks to solve the routing problem. Emphasis is placed on the ability to constantly evaluate the quality of the findings, i.e., working paths between sources and destinations, and the adjustments made to the artificial bee dance. Inspired by nature, the proposed approach uses important low-level information such as the wireless signal strength, moving speed of nodes, and remaining energy level as the quality factors to judge paths and control their usage in future transmissions. It is a novel approach which brings together features from artificial intelligence (Russell & Norvig, 2003) and traditional routing, in order to provide a new way of developing bee-inspired protocols that are able to make decisions utilizing intelligent and adaptive techniques.

1.2 Towards research

In this section both the research hypothesis and question to be answered are given.

1.2.1 Hypothesis and research question

The key hypothesis that stems from this research work is expressed as,

Hypothesis: "*If the method used by a honeybee colony in order to evaluate the quality of a food source is a corporate consideration of a number of factors, which indicate different attributes of the food source as well as the foraging process in general (environmental conditions), then an artificial colony based on similar principles should be able to obtain analogous information, and use it to evaluate the quality of the paths between sources and destinations and control the number of transmissions that will use the paths in the future.*"

This allows the formulation of the following research question:

Research Question: "*To what extent will a bee-inspired model that allows agents to discover and constantly measure and monitor the quality of paths in a MANET, based on low-level information obtained by the network, be able to adjust the future transmissions over those paths, in order to provide adaptive, decentralized and robust routing in comparison to other existing state-of-the-art approaches?*"

Satisfying the above hypothesis and answering the research question, will offer a valuable insight into the effectiveness of modelling a rich number of low-level network information, in order to measure and evaluate the quality of routing solutions in a MANET. It will also illustrate an effective way of distributing the traffic load over the optimal routing solution, based on a recruitment system which mimics real honeybee communication (bee dance). The result will ultimately provide a novel way of mapping nature to networking and, in turn, a new design of routing protocols in highly dynamic wireless networks.

1.2.2 Aims and objectives

In order to conduct a successful course of research towards the above problem and proposed solution, the following aims and objectives are derived.

- Concepts from the honeybee colony in terms of foraging need to be carefully mapped to the wireless network environment. The importance of the natural quality factors need to be understood and then substituted by network counterparts, to support quality measuring and monitoring.
- The honeybee dance as a medium of communication also needs to be carefully modelled, in such a way that the incoming input from the artificial foragers will allow adjustments to the artificial recruitment, thus, the number of transmissions over each path.
- A system of bee-inspired agents needs to be designed. The agents will be able to discover new routing paths and maintain the results as long as the network is in use. As they will move in a hop by hop fashion, they must be able to access the appropriate information and utilize this upon their arrival at the artificial hives (source nodes). There, the system needs to model a decision making mechanism to drive the artificial recruitment.

- The above design of the system needs to be implemented in a network simulator to allow evaluation of and improvements to the system.
- Finally, an extensive comparison needs to be made between the new proposed design and protocol with existing state-of-the-art approaches, and to quantify the success of it through a rich number of experiments studying difference quality metrics.

1.3 BeeIP: The proposed routing design

In this thesis, a new bee-inspired routing design is proposed, in order to solve the routing problem and answer the research question of section 1.2.1. The work emphasizes the ability of honeybees to perform foraging and to communicate with each other within the hive in order to archive efficient and productive recruitment. It is an extended mapping of concepts and principle behaviours between nature and networks which allows routing to be achieved between mobile wireless nodes in ad hoc telecommunication scenarios.

Following this design the resulting routing protocol, BeeIP, is able to reactively discover multiple paths between sources and destinations, and distribute traffic across them in a scalable, robust and efficient way. The novelty of the system is seen by the way agent-like control packets, which emulate the real scouts and foragers, constantly monitor and evaluate the performance of the previously discovered paths. By the means of an artificial honeybee dance, and the use of statistical tools, the artificial honeybees are able to perform recruitment based on the path quality feedback as well as their past knowledge.

In addition, deciding which is the most appropriate path to follow not only depends on what options are available from the dancing honeybees, but also on the particular need for a specific commodity (nectar, pollen or water). Following the proposed design, BeeIP has a flexible way of utilizing a selection metric for the next hop at the source nodes, which is related to the particular behaviour the protocol is required to achieve. The latter is designed to be defined by the implementation of the protocol and depends on the application needs.

Adaptation is achieved as the protocol is able to make different routing decisions every time routing is required within highly dynamic networks. The nodes follow a decentralized and distributed approach in order to discover paths and forward packets, and routing becomes self-organized as no prior planning is required.

1.4 Results overview

The proposed design and its implementation have been extensively compared with four representative, state-of-the-art routing protocols in the area. The comparison has been made against seven performance metrics. Namely, control overhead, packet delivery ratio, average end-to-end delay, throughput, path duration, network life, and number of route requests. In addition, the comparison has been separated into five different sets of experiments, each one investigating a different wireless network environment by varying one of the network configuration aspects, such as terrain area size, number of nodes in the topology, network traffic, node speed, and the time nodes will remain paused before they start moving again (pause time). These changes allow different network environments and conditions to be simulated and provide a complete way of testing the proposed design.

The results obtained show that the bee-inspired routing protocol generally outperforms the other protocols in most of the performance metrics. The biggest strength of the protocol is seen when observing the average end to end delay and packet delivery ratio. The protocol is able to deliver packets faster, due to its packet switching mechanism and ability to discover and use multiple paths. These two characteristics allow the protocol to outperform the others, especially within highly dynamic networks, e.g., increased node moving speeds and varying pause times. Additionally, the protocol is less sensitive to the terrain size changes, and maintains a balanced control overhead during mobility changes. In terms of network life, no clear assumption can be made, since the protocol shows no significant improvement compared to the other techniques.

1.5 Contributions

This thesis makes the following distinct contributions, not only in the field of wireless telecommunications, but also in artificial intelligence and in turn, swarm intelligence (Fleischer, 2005), as it combines principles of these areas in order to solve the routing problem. In particular, this thesis offers:

- A new way of providing routing in wireless mobile ad hoc networks, based on bee-inspired behaviours. The novelty in the system is seen in how measuring and monitoring of the quality of paths is achieved, as well as the decision mechanism which affects the artificial recruitment.

- An extended mapping of behaviours and key concepts from the world of honeybees to networking.
- An approach of using a rich set of quality factors to evaluate paths, and a way of combining them by the means of artificial intelligence and in particular artificial neural networks (Gurney, 1997). The result of this study offers an insight into the general importance of each selected parameter and their potential in affecting the overall performance of the path. The resulting model is designed to mimic nature where real honeybees judge the quality of their findings by considering a rich number of factors.
- A study which shows that adaptive routing can be achieved with the aid of making statistical predictions, based on the quality of the paths over time. This behaviour mimics the one met in nature, where evidence shows that honeybees can sense improvement or deterioration of the quality over time. By being able to consider prior knowledge, the artificial honeybees are able to detect changes on paths and affect their decisions in terms or recruitment in an adaptable manner.

1.6 Structure of the thesis

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

Chapter 2: Wireless ad hoc networks. This chapter reviews the existing related work in the field of wireless telecommunication, and in particular wireless ad hoc networks. Important approaches and techniques of providing routing are highlighted and discussed, with particular emphasis given to well-known state-of-the-art protocols. Additionally, the cross-layer technique is presented, a useful tool which is heavily used by this research work in order to access and obtain low-level information from other protocols of the network stack.

Chapter 3: The world of honeybees. This chapter gives an overview of the most important concepts in the world of honeybees, and constructs a solid background for understanding the reasons behind the internal structure and modelling of the proposed design.

Chapter 4: BeeIP: The bee inspired protocol. In this chapter, a full description of the proposed design is given. BeeIP is a bee-inspired routing protocol which takes into consideration a variety of low-level information of the network, to measure and monitor the quality of paths between sources and destinations. Routing is provided by selecting the most appropriate path (in terms of user needs), from a list of available paths for which artificial foragers have been recruited. Modelling of honeybee behaviours in terms of foraging and communication within the hive are described in this chapter.

Chapter 5: Methodology of experiments. This chapter is dedicated to the description of the methodology of the comparison experiments. Important decisions, such as the selection of the appropriate simulator, protocols that the proposed design is compared with, as well as the configuration and simulation scenarios of the experiments are thoroughly discussed.

Chapter 6: Evaluation study. Here, the results of an extensive set of simulations are presented and discussed, highlighting the strengths and weaknesses of the proposed design of routing.

Chapter 7: Conclusion and future work. This final chapter concludes the thesis, by discussing the initial hypothesis, question and contributions of this research work, its potential in being used in real applications, and makes suggestions for future improvements.

Chapter 2

Wireless ad hoc networks

2.1 Introduction to wireless ad hoc networks

The research work presented in this thesis, focuses on routing in wireless ad hoc networks. This chapter provides information regarding this particular type of networking, as well as an overview of the existing work which is done by the research community. The two important types of wireless ad hoc networks (Mobile Ad Hoc Networks and Wireless Sensor Networks), that dominate the research interest are defined. Coming from the wired Internet, wireless ad hoc networks expand the way people see computer networking and develop network applications. However, the nature of the wireless communication is such that it introduces several issues, which affect both routing and packet forwarding. These issues and ways to tackle them are summarized in this chapter. Routing protocols are mainly categorized as proactive, reactive and hybrid. Section 2.5 gives a definition of each category and overviews several representative routing protocols that are found in the field. Since the work presented in this thesis is nature-inspired, section 2.6 is used to provide an overview of other nature-inspired routing protocols, that have been recently proposed. Finally, alternative ways of routing as well as an introduction to cross-layering (an important approach in network protocol design that is heavily used in this work), are given in section 2.7.

2.2 Types of wireless ad hoc networks

In the last few years, there has been an increased interest in wireless ad hoc networks as they have a tremendous potential for use in commercial, research and military applications. Nowadays, an increased number of devices are equipped with wireless network capabilities, which allow them to connect to existing networks or the Internet, as well as to each other, in order to form an ad hoc network. Wi-Fi, Bluetooth, Cellular data service, etc., are some of the technologies in wireless communication which are used in everyday life.

A wireless ad hoc network is comprised of mobile computing devices which use wireless transmissions for sending and receiving data. In addition, wireless ad hoc networks have no fixed infrastructure, which allows them to deploy quickly, eliminating the complexity of the infrastructure's set-up. As all devices are allowed to move around, no single device is expected to act as central administration (such as a cell site for cellular data services, or an access point device for Wi-Fi). Thus, due to the limited wireless transmission ranges, all devices need to provide routing and must also be able to forward data packets to the correct destinations.

Studying wireless ad hoc networks is a very active research field. This is because such networks find applications in a variety of areas. Related to militia or natural catastrophes such as fires and tornadoes, tactical networks (Aschenbruck *et al.*, 2008) are deployed by forming wireless ad hoc communication between mobile devices, which operate within automated battlefields or places where the communication infrastructure is destroyed. Emergency services (Chlamtac *et al.*, 2003) are also applications where wireless communication cannot depend on fixed infrastructure. Search and rescue operations, disaster recovery, policing and fire fighting are all applications which belong to this area (Kostoulas *et al.*, 2008). Other applications are found in education such as ad hoc communication in university campuses (Stuedi & Alonso, 2007), virtual classrooms or communication during learning and teaching (Vasiliou & Economides, 2007), in entertainment such as multi-user games, outdoor Internet access, etc., in research such as multi-robot systems for terrain exploration (Yoshida *et al.*, 1995; Alena & Lee, 2005), and many others.

Current literature focuses on two major types of wireless ad hoc networks, the Mobile Ad hoc Networks (MANETs) and the Wireless Sensor Networks (WSNs). This is due to the fact that those two types are heavily used in the above application areas. In the next sections, 2.2.1 and 2.2.2, MANETs and WSNs are described and compared, providing the background to understanding the potential problems in data communication and routing within such networks.

2.2.1 Mobile ad hoc networks

Mobile ad hoc networks (MANETs) (Ram Murthy & Manoj, 2004; Marwaha *et al.*, 2002), have all the wireless ad hoc network characteristics. In a MANET, all the participants (nodes) are mobile and use only wireless communication links to send and receive data packets. There is not a fixed infrastructure or hierarchy between the participants. Being equal, they need to act as end hosts of a transmitting session, as well as routers. Thus, MANETs are highly dynamic and decentralized networks. Since each node is able to provide routing and forward data packets, MANETs are also multi-hop networks. That means that during a communication session, packets can be forwarded multiple times, hop by hop, in order to reach their final destination. Although MANETs' deployment is easy and cost-effective, they require complex distributed routing algorithms to operate. In conjunction to this, self-organization, configuration and maintenance are built into the network, in the form of network protocols. In terms of implementation, this increases the complexity and requires a careful design of a node's architecture, which involves all network layers of the network stack from the physical to the application layer.

The first MANETs have primarily been used for tactical networks for mili-

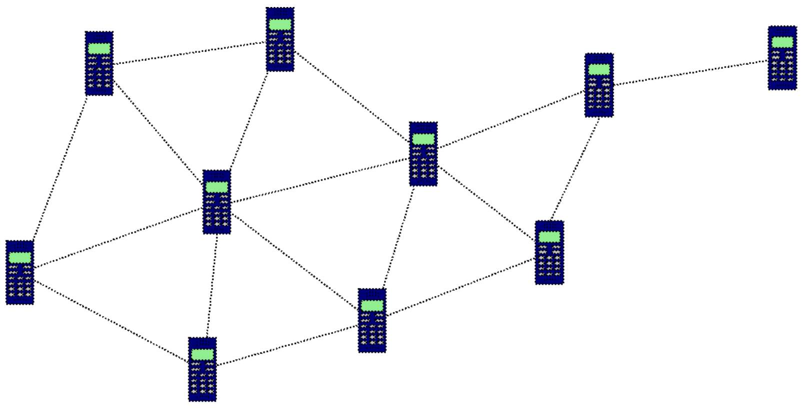


Figure 2.1: An example of a mobile ad hoc network (MANET), consisting of 10 nodes which communicate by wireless links. The dashed lines illustrate the wireless links, forming several possible routing solutions between potential sources and destinations.

tia (Taneja & Patel, 1972), in order to improve battlefield communication and survivability. Their highly dynamic nature offers an excellent communication between military devices, which cannot rely on access to a fixed placed infrastructure, in a battlefield or any other type of harsh terrain. Since then, MANETs have concentrated the interest of the research community and they have evolved and improved to satisfy a variety of needs. One of the earliest mobile ad hoc networking application is the DARPA Packet Radio Network (PRNet) project in 1972 (Jubin & Tornow, 1987). PRNet applied the packet switching technology in a mobile wireless environment, by utilizing a distributed architecture of multi-hop routing which offered an ad hoc communication between users within large geographic areas.

Another example of implementing a MANET, this time by using a large-scale implementation, is the Tactical Internet (TI) maintained by the US army in 1997 (Sass, 1999). The key feature of TI is the ability to exchange messages using the commercially-based Internet Protocol (IP), which is common across all nodes of the various TI segments.

MANETs do not rely on a specific type of device. Instead, a group of different computing devices, such as laptops, mobile phones, tables, etc., can form a mobile ad hoc network as long as they follow the same communication technology, for instance, they are all Wi-Fi enabled. An example of such a network is shown in figure 2.1. Nonetheless, MANETs are not perfect. The challenges of scalability, mobility, bandwidth limitations and power constraints of these networks, have not been completely alleviated to date. Network protocols for MANETs need to be highly dynamic and operate in a fully distributed way. They also need to be adaptive to any topological changes and robust in order to face unreliable links between transmitting nodes. Finally, due to the network limited resources, protocols need to provide as cheap as possible solutions in terms of energy consumption, bandwidth and computing power.

2.2.2 Wireless sensor networks

Wireless Sensor Networks (WSN) (Akyildiz *et al.*, 2002; Xu, 2002) are ad hoc networks which consist of a large number of nodes equipped with some sort of a sensor and are generally deployed in order to measure physical parameters of a certain phenomenon. They are the key to gathering the information required within a variety of environments, from buildings in an urban scenario, to deep forest, banks of river, surface of a lake or bottom of the ocean.

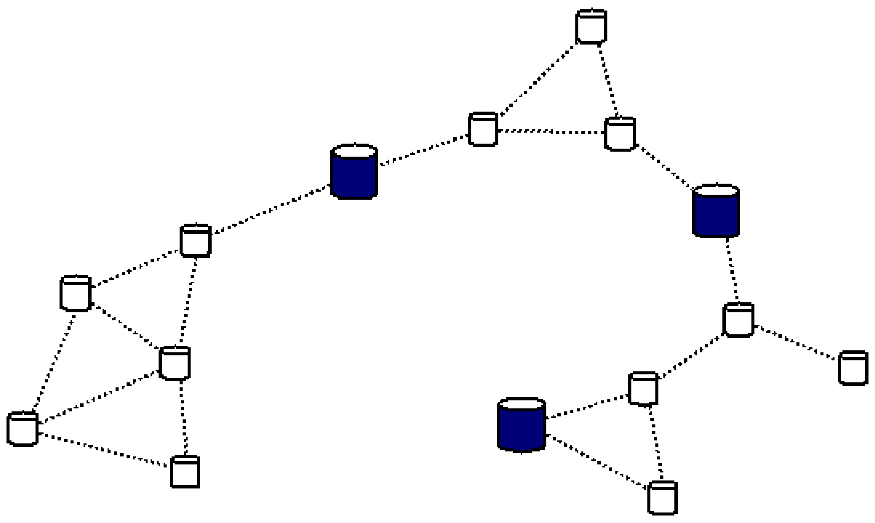


Figure 2.2: An example of a wireless sensor network (WSN). Here, the small cylinders illustrate the sensor nodes and the large coloured cylinders the sink nodes. Both sensor and sink nodes communicate with each other using wireless links, illustrated as dashed lines. Sensor nodes are responsible for collecting sensory data and send it to sinks, where the processing is being done.

Sensor nodes are small and cheap devices with very low data processing capability. Their purpose is to measure certain environmental values, gather and transmit the data to the monitoring station nodes, where it can be processed. The activity of sensing can be periodic or sporadic and the transmission of the data to the monitoring stations can also follow a pattern (e.g., collect information at frequent intervals). The fact that they are ad hoc networks expands their ability to deploy easily within difficult environments. For instance, in a geographical project, one can measure several characteristics of a lake by throwing sensor nodes into it and by placing sink nodes at positions where physical access can be gained, like the bank of the lake, or a platform on its surface. Figure 2.2 shows a simple WSN deployment, of 12 sensor nodes and 3 sinks.

Similarly to MANETs, WSNs need to be fast and adaptive to dynamic environments. As it is very difficult to reach sensor nodes in some cases (e.g., sensor nodes are thrown into a lake), network protocols designed for WSNs need to be robust and self-configured. Due to the lightweight and cheap nature of the sensor nodes, protocols need to be able to operate using as little resources as possible for data processing, storage and transmission power. These resource constraints also suggest that WSN protocols should be designed to be aware of data fusion (Chen *et al.*, 2009), a collection of techniques which refers to aggregating several small packets into one before relaying it. This not only reduces the number of headers bits being used, but also speeds up the transmission of multiple packets by eliminating the media access delay involved.

In most cases, mobility in WSNs is not a mandatory requirement. However, this depends on the application. The sensor nodes which periodically transmit environmental values such as the level of humidity or heat in a forest during a fire detection activity are not required to be mobile. On the contrary, the sensor nodes which are fitted on animal bodies in order to measure their everyday activities may be designed to support limited or partial mobility. Although mobility is not always an issue for WSNs, they still need to be adaptive to topological changes. This is due to the highly dynamic terrains caused by physical changes, e.g., weather conditions affect the position of sensor nodes, or the existence of non-replaceable broken sensor nodes that break the communication.

In terms of network size, deploying WSNs often requires hundreds or thousands of sensor nodes to cover the required terrain. This implies that protocols designed for WSNs need to scale up easily without losing efficiency.

2.3 Issues in wireless ad hoc networks

Wireless systems operate by transmission through free space rather than through wired connections. Hence, many factors influence the data transmission, like the form of the terrain, atmospheric conditions, buildings and other obstacles. Although unknown to the wired world, in a wireless environment these problems affect both data transmission and routing. In this section, a discussion related to the impact these problems have on routing is given, by looking at the wireless channel in general as well as the three involved layers of the network stack; the physical, data link and the transport layer. Furthermore, separate problems connected to node mobility and limited resources are also discussed.

2.3.1 Wireless channel and signal propagation

Not only an issue for routing, but rather packet transmission in general, a wireless channel is prone to a variety of transmission impediments such as path loss, fading and interference. Depending on the environmental conditions, these factors restrict the range, data rate and reliability of the wireless links between the nodes.

Path loss and shadowing. Being a function of the propagation distance, the path loss is expressed as the ratio of the power of the transmitted signal to the power of the same signal received by the receiver on a given path. It is the reduction of power density the electromagnetic wave experiences as it propagates through the free space. Its estimation (called prediction) is very important for designing and deploying wireless networks. Path loss directly depends on the nature of the terrain. Therefore, a single model is not enough to cover all wireless transmission environments. Several path loss models have been proposed, such as the Free Space and Two-Ray path loss model (Ram Murthy & Manoj, 2004; Takai *et al.*, 2001). Moreover, signals also suffer loss in power due to obstacles along the way, an effect called shadowing.

Fading. This refers to the fluctuations in the signal strength of a transmission due to environmental surrounding reflectors when it is received at the receiver, and is classified as slow or fast fading. Fast fading occurs due to the interference between multiple copies of the same transmitted signal arriving at the receiver at slightly different times. Reflection, diffraction and scattering (Sarkar *et al.*, 2003) are the main sources of fast fading. Slow fading occurs when there is an object which partially absorbs the wireless transmission positioned between the transmitter and the receiver. For instance, the latter may occur when the transmitter is within a building where the receiver is outside and the propagation has to go through a thick wall.

Interference. Since the wireless channel is a shared medium, multiple sources transmitting signals at the same time can result in the signals being corrupted and undecodable at the corresponding receivers. This phenomenon is encountered from a variety of sources. One main type of interference occurs when signals in nearby frequencies have components outside their allocated ranges. For instance, multiple Wi-Fi networks which operate in different but adjacent channels are likely to suffer from this form of interference. Similarly, when nearby systems use the same transmission frequency, they are subject to co-channel interference. Another frequent form of interference is the inter-symbol interference (Ram Murthy & Manoj, 2004). During inter-symbol interference, distortion in the received signal is caused by symbols (pulse or tones of an electromagnetic wave) overlapping with subsequent symbols, causing noise and making the communication less reliable.

Bandwidth. Fiber optics and the exploitation of wavelength division multiplexing offer high bandwidth in wired networks. Unfortunately, this is not the case in wireless communication. The radio band is limited and the data rates that it is able to offer are much less than what a wired network can offer. Limited bandwidth in wireless communication encumbers the routing protocols which often need to keep partial or full routing information for all the topology. This is because a lot of messages are required to be exchanged between nodes, which in turn results in more bandwidth being wasted.

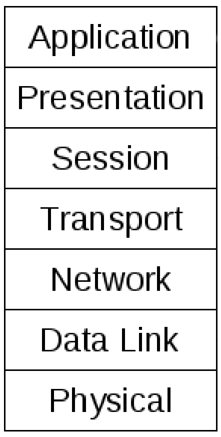


Figure 2.3: Open Systems Interconnection (OSI) network stack, illustrating different layers.

2.3.2 Physical layer

The principle task of the physical layer (PHY) (figure 2.3), is to transmit streams of bits from the transmitter to the receiver with minimal bit errors. Here, the two major issues which affect routing protocols are discussed.

Unidirectional and bidirectional links. Bidirectional wireless links are those where if node A can receive packets from node B, then node B can also receive packets from node A. Although the majority of protocols found in the literature assume that all links between the nodes are bidirectional, in reality an ad hoc wireless network may contain unidirectional links. They can occur for various reasons. One reason can be the difference in the transmission ranges of the nodes. If node A has higher range than B, then it is possible for B to receive packets from A, while A is not able to pick up packets from B. When this happens, a possible explanation is the difference in the remaining battery level of each node. Another related reason for the occurrence of unidirectional links, is the irregular patterns which are formed instead of a perfect circular transmission range, due to the difference in radio wave propagation to different directions (Zhou *et al.*, 2004). Moreover, unidirectional links may also occur as a result of the interference levels at each transmitter and receiver. It is possible that the level of interference at node A is different to that of node B, so that one of the two can temporarily be unable to pick up packets sent by the other.

Symmetric and asymmetric links. In a wired world, communication links are mostly designed to be symmetric. Symmetry means that bidirectional communication over the same link is of the same quality. The control of possible noise in the wired medium allows symmetric communication to become possible. However, wireless communication bidirectional link asymmetry can be observed due to a number of reasons, such as interference, multi-path effects, weather conditions, etc. (Aguayo *et al.*, 2004; Kotz *et al.*, 2004; Haykin, 2005). Especially, in MANETs and WSNs, due to their usual deployment in large and heterogeneous areas, wireless link quality fluctuates significantly. Hence, most of the network routing protocols which are designed to provide shortest-path and geographical routing are built under the assumption that if node A can hear from node B, no matter what the quality is, the connection is good enough to provide a routing solution (Kim & Shin, 2006).

2.3.3 Data link layer

The data link layer (DLL) is responsible for preparing and transmitting data between nodes. It often provides the functionality of correcting possible errors that may have occurred in the PHY layer. As defined by the IEEE 802 standards, the data like layer consists of two parts, namely the media access control (MAC) and logical link control (LLC) sub-layers. Wireless IEEE 802.11 (adopted in 1997) specifically emphasises the MAC sub-layer, for it is responsible for three important aspects of networking; reliable data delivery, access control and security. Thus, in this part of the chapter the discussion is focused on the MAC layer and, in particular, the most common MAC protocol for wireless networks that lack a fixed infrastructure, the IEEE 802.11 Distributed Coordination Function (DCF).

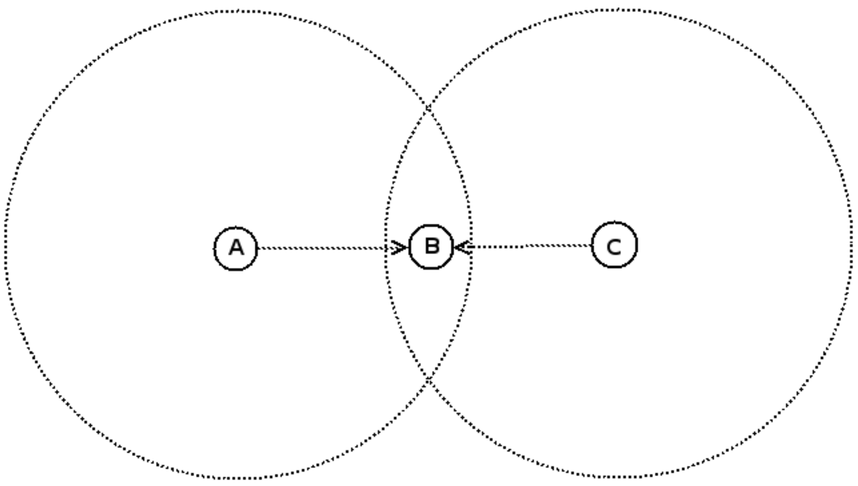


Figure 2.4: An example of the hidden terminal problem. Here, if node A is sending data to node B, node C is totally unaware of the ongoing transmission, resulting in severe packet loss at node B.

The hidden and exposed terminal problems. Routing in wireless networks is affected by hidden and exposed terminal problems, for which solutions are meant to be provided by the MAC protocols. The hidden terminal problem is a common phenomenon that is met due to the multi-hop nature of wireless ad hoc networks. For instance, in figure 2.4, when node A is sending data to node B, node C is completely unaware of the ongoing transmission. If C attempts to send data to B, then the hidden terminal problem will occur, causing collision at node B, and packet loss.

The exposed terminal problem, occurs when a node attempts to transmit data while a neighbouring node is already transmitting. This is due to two transmitters within the same transmission range, trying to send data simultaneously. In figure 2.5, a collision happens when node C requires transmitting data to node D, while

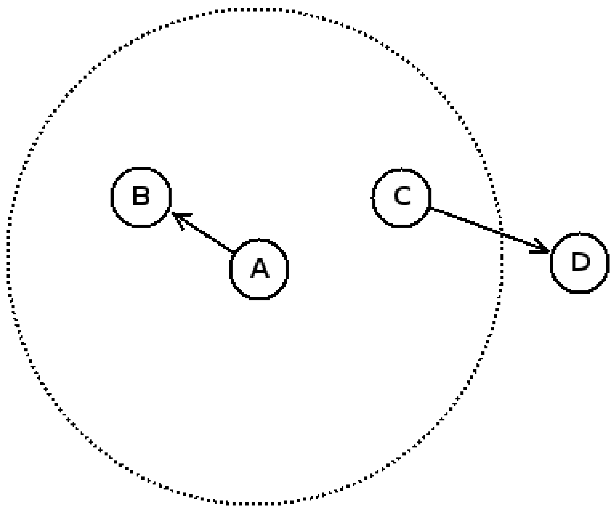


Figure 2.5: An example of the exposed terminal problem. Here, if node A is already transmitting data packets to node B, then node C will be unable communicate with node D, because of the neighbouring transmitter A.

node A is sending data to node B. This leads to a blocked communication where no node will be able to transmit or receive packets.

Sharing of the wireless channel is a difficult problem, and it is the MAC protocols' responsibility to tackle. In an ad hoc environment where no access point is present, the de facto mechanism of solving problems such as the ones described above, is the carrier sense multiple access with collision avoidance (CSMA/CA) (Brenner, 1997). In CSMA (Kleinrock & Tobagi, 1975), a transmitter may transmit if and only if the medium is sensed to be idle. A protocol which is based on this mechanism and is implemented to all Wi-Fi transmitters is the IEEE 802.11 DCF protocol (Mueller, 2007), discussed below.

IEEE 802.11 Distributed Coordination Function. This media access control mechanism is historically influenced by the multiple access with collision avoidance for wireless (MACAW) (Bharghavan *et al.*, 1994) and the multiple access collision avoidance (MACA) (Karn, 1990) protocols, which introduced the request-to-send/clear-to-send handshaking scheme in order to effectively use the wireless medium over different nodes, reducing the possibility of collision.

The CSMA of the IEEE 802.11 DCF works as follows. When a node is required to send data to a destination, it must first sense the channel to determine whether it is already being used for another transmission. If the channel is busy, the node goes to a back-off phase, when it has to wait for a fixed interval before it attempts to retransmit. The interval is switched to a random back-off value if the channel is assessed busy for a second time. The basic CSMA mechanism requires each frame that is sent by the MAC layer to be acknowledged. Therefore, after a successful transmission of a frame, the receiving node prepares an ACK frame and sends it back to the sender. If no ACK frame is received for a particular frame, the transmitter schedules a retransmission.

Apart from the CSMA, IEEE 802.11 DCF also employs a collision avoidance mechanism, which involves request-to-send (RTS) and clear-to-send (CTS) control frames to be sent between the transmitter and the receiver prior to any unicast data frame transmission. These control frames are heard by all nodes within the transmission range, which block their own transmission for the duration of the transmission[1] to prevent any collision.

The RTS/CTS handshake scheme addresses the problems of hidden and exposed terminals rather satisfactorily, but problems may still persist in some cases. This is due to the fact that they are "transmission-range" related and therefore are subject to physical conditions and energy consumption. For instance, Fu et al. (2003) and Xu et al. (2003) have shown that the power which is required for interrupting a packet reception is much lower than that of delivering a packet successfully, meaning that the transmission range of a node is smaller than its sensing range. Studies have also discovered that the problem of hidden terminal occurs less frequently when the transmission range is increased (Hanbali *et al.*, 2005). In a similar manner, IEEE 802.11 RTS/CTS mechanism helps to solve the exposed terminal problem only under certain conditions (clock synchronization between nodes, shared packet sizes and data rates). Additionally, another auxiliary functionality provided by the MAC layer is the clocks synchronization of all the wireless nodes, a feature which is exploited by a number of power-aware protocols.

The efficiency as well as the limitations of the legacy IEEE 802.11 DCF is still under investigation by the research community. In Chetoui et al. (2007), the problem of fairness in terms of the medium occupancy is illustrated. The authors mention that although the CSMA/CA mechanism ensures equal access to the shared medium, taking into consideration the destination between the transmitter and the receiver, fair medium occupancy cannot be always achieved. The time a node captures the channel to transmit a data frame increases as its nominal

[1]The duration is found in the control frame's header as the Duration/ID field.

bit rate decreases, due to the greater distance between the two antennas. In this regard, the overall maximum throughput of the network is negatively affected. Another drawback of the DCF protocol which affects the maximum throughput is the transmission of the extra control frames, i.e., RTS, CTS, and ACKs. The control overhead caused by the handshake scheme reduces the theoretically expected throughput (Jun *et al.*, 2003). Similarly to the throughput limitations, packet delay bottlenecks do exist in IEEE 802.11 DCF, mainly because of the randomness in the back-off interval window, especially within busy networks. Jun et al. (2003) present a detailed analysis for theoretical delay limits in different IEEE 802.11 specifications, pointing out that the threshold when packets arrive faster than the packet service rate, and vice versa, depends on the packet arrival pattern (e.g., general data or VoIP data traffic).

2.3.4 Transport layer

Moving to the fourth layer of the OSI model (one layer above the network layer), the transport layer is responsible for providing end-to-end communication services for applications. Amongst these services are connection-oriented or connectionless communications, data sending reliability, congestion avoidance and flow control (Forouzan, 2005). When a source is required to send data to a destination, a transport layer protocol organizes the data into segments and starts a communication session by using the routing solution provided by the network layer's protocol. The Internet today uses two de facto transport protocols for both connection-oriented and connectionless data streams. Both these protocols are briefly discussed in the following paragraphs, with respect to the impact they have on wireless communication and, in particular, routing.

Transmission Control Protocol (TCP). This state-of-the-art transport layer protocol is able to provide secure data transport between two nodes. At the beginning of a communication session between two nodes, the sender starts a TCP connection with the receiver, exchanging control packets by following a triple handshake scheme. TCP offers a reliable data transport service by acknowledging all data segments, while it also makes sure that they arrive in the correct order. TCP was designed to work in wired networks and because of that, its performance is quite poor when applied to highly dynamic networks such as MANETs or WSNs.

Firstly, TCP does by itself produce excess control overhead while applying its acknowledgement and data integrity mechanism (Gerla *et al.*, 1999). This contradicts one of the fundamental needs when designing wireless protocols: provide as little control overhead as possible to allow real data to occupy the medium. Another reason is the frequent link failures in wireless ad hoc networks, which may lead to increased packet loss. In TCP, this triggers the protocol's congestion control, which in turn, reduces the sending rate (by shortening the congestion window). Thus, TCP's reaction to false congestion in the network decreases the protocol's effectiveness (Xiao *et al.*, 2010). Next, unidirectional and asymmetric paths between sources and destinations also cause difficulties to TCP performance. Asymmetry may manifest in several forms like loss rate asymmetry, route asymmetry, etc. Especially when multi-path routing protocols provide more than one available routing solution for a given communication session, ACK streams may be disrupted and lead to false estimations of the available bandwidth, degrading the performance (El-Sayed *et al.*, 2005). However, despite these problems, TCP remains the state-of-the-art transport protocol when reliable data transportation is required in wireless environments.

User Datagram Protocol (UDP). This is second the state-of-the-art transport protocol, which is heavily used in the literature for both real-life and simulated experiments. The service provided by UDP is connectionless, meaning that there is no implicit handshake dialogue required in order to establish a connection and start transferring data. This also implies that the protocol offers no guarantee that the data will arrive at their destination correctly. UDP just sends datagrams from the source to the destination. The simplicity of UDP (adds no extra control overhead) as well as the need to factor out any transport layer difficulties are what make UDP the favourite choice when experiments need to be conducted towards the improvement of other low-level protocols such as for the network, or MAC layers. However, a valid counterargument is that real time applications do not exclusively make use of UDP and protocols should be able to deal with both connection-oriented and connectionless transport protocol.

2.3.5 Node mobility

From all the previous sections, it is understood that the mobility of nodes plays a significant role in the performance of the wireless network. MANETs as well as WSNs, depending on the deployment, consist of a number of nodes placed in a terrain where the network topology is neither considered nor planned previously. The impact node mobility has on a routing protocol's performance is an active research area (Radha & Shanmugavel, 2001; Dousse *et al.*, 2002), and can mainly be seen as the product of frequent link breaks, energy consumption, and node changes to the density within the topology.

Firstly, link breaks are caused when the two participating nodes of a data transmission move further away from each other, so that the receiver goes outside the transmitter's range. This is a typical situation in MANETs that routing protocols need to cope with. In some cases (see section 2.5), link breaks are detected by an expired timer which represents a packet that although it was sent, was never acknowledged. This causes extra delays and affects the overall throughput of the network. In addition, the recovery process after a link is found broken, requires extra control overhead to occupy the medium.

Secondly, route discovery, routing maintenance, and topology self-organization require control packets to be exchanged between neighbouring nodes. The more frequently they occur, the more extra control packets need to be sent, the more energy needs to be spent. This counteracts the limited resources in wireless ad hoc networks, which is discussed in section 2.3.6.

Finally, the node density is a major factor that has much influence on the performance. Work has been done to highlight this fact (Dousse *et al.*, 2003; Deepa & Nawaz, 2010). Although a sparse density increases the possibility of link failures due to nodes exiting the transmission ranges, studies have shown that high density wireless networks also suffer in connectivity. In Blake and Pullin (2007) as well as in Ververidis and Polyzos (2005), the authors have studied the impact of density on the performance of routing protocols in MANETs and show that the affect is of various forms. They show that the higher the density of the network, the greater the load within it. Throughput is affected as more control overhead is broadcast to the network. Coupled with this, they show that the greater the node density of the network, the greater the packet loss rate. To conclude, clustering is also a frequent problem when nodes move around the network topology. Depending on the various moving patterns the nodes follow, they can form small internally connected groups which are connected to each other by a few nodes sitting at the edge of the group's transmission range. Furthermore, when several communication sessions are required between nodes of different groups, traffic is eventually going through the nodes at the edges, causing bottlenecks and load congestion.

2.3.6 Limited energy resources

Nodes which participate in a wireless ad hoc network are usually limited by energy resources. This problem gets bigger in WSNs, where sensor nodes tend to be very small in size, so their batteries tend to be smaller. Energy management is itself a separated research objective (Tantubay *et al.*, 2011).

The impact energy consumption has on mobile wireless networks is large, due to the following facts. Nodes consume energy while sending, receiving and even discarding packets. In addition, studies have shown that memory allocation at the mobile node is one of the most important reasons behind high energy consumption (Amiri, 2010). Hence, consumption does not only take place at the PHY or MAC layers, where the packets take their final steps in order to be transmitted, but also at other layers of the network stack where changes to the packet or decisions (such as routing) are made (Chauhan & Chopra, 2010). Furthermore, power consumption in wireless ad hoc networks is directly proportional to route length, that is, if the route length (number of hops) is increased, then the power consumption is also increased.

2.4 Routing problem in wireless ad hoc networks

There are several reasons why wired routing protocols are not appropriate for wireless communication and, in particular, mobile wireless ad hoc networks. These reasons are listed in this section followed by a discussion of the most important features one must consider, when designing a new routing protocol.

- Wired protocols are designed for fixed topologies with immobile nodes. The rate of link breaks is much less than in wireless ad hoc networks.
- Due to abundant bandwidth, wired routing protocols are not taking a minimalistic approach in exchanging routing information.

- Collision detection cannot be achieved in wireless (where collision avoidance is used instead). This enhances the routing problem as it adds extra delays to the packet transmission.

- Since the topology in a wired network changes infrequently, routing loops rarely occur. On the contrary, moving nodes alter the topology fast and unexpectedly, making adaptation and self-organization far more difficult.

To overcome the issues of wireless ad hoc networks, routing protocols need to be designed with certain characteristics. Firstly, a wireless routing protocol needs to be fully distributed. Distributed routing is more tolerant to faults than centralized, and produces less control overhead. In addition, distributed routing algorithms scale up easier as the network becomes larger. Next, the protocol needs to be adaptive to frequent topological changes caused by the mobility of the nodes and the changes of the terrain. In terms of topology knowledge being used by each node, the protocol has to be localized. In other words, fewer nodes should be involved while computing and maintaining a routing solution between a source and a destination. Next, optimal routing solutions need to be generated fast, with minimum set-up time. Stale routes need to be deleted as quickly as possible. Finally, a routing protocol should be able to optimally use the limited resources of the mobile node, such as computing power, memory and battery energy, as well as the available bandwidth.

Routing protocols for wireless ad hoc networks are classified into several types, based on their routing approach and mechanisms (Lie & Kaiser, 2005; Boukerche, 2009). The most common classification that is met in the literature is based on the routing information update mechanism. Other classifications consider the utilization of specific resources such as power-aware and location aided, and others (see section 2.7). Recent research on designing adaptive routing protocols has brought into light another way of classifying protocols to those inspired by the traditional wired Internet protocols (Internet-inspired), and those inspired by nature (nature-inspired). A literature review on several important routing protocols is found in the following sections 2.5, 2.6, and 2.7.

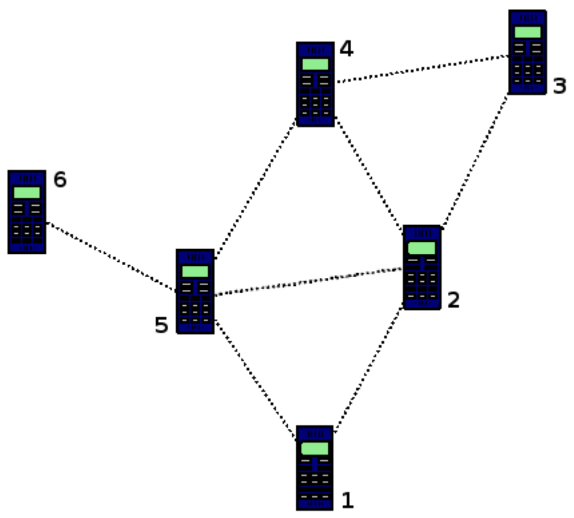
2.5 Internet-inspired protocols

Here, routing protocols that are inspired by the wired Internet but are designed and implemented for wireless environments are presented. Based on how routing information updates are achieved, including the routing discovery process, these protocols are either proactive, reactive or hybrid.

2.5.1 Proactive

In proactive or table-driven routing protocols, the network topology information is maintained within each node in the form of routing tables, which are periodically exchanged in order for the information to be kept up-to-date. Generally, routing information is flooded through the whole network allowing hearing nodes to update their topological knowledge. Whenever a route is required at the sender, a searching algorithm is applied on the routing tables to find the optimal solution. The advantage of proactive routing is that routes to all destinations are readily available at every node at all times. Although this method reduces the delay of the route discovery dramatically, it adds a large amount of control overhead to the network traffic. Examples of proactive protocols are the Destination Sequenced Distance-Vector (DSDV) (Perins & Bhagwat, 1994) and Optimized Link State Routing (OLSR) (Clausen & Jacquet, 2003), discussed below.

Destination Sequenced Distance-Vector. Historically, this is one of the first routing protocols proposed for wireless ad hoc networks (1994). It is a pure table-driven protocol and an enhanced version of the distributed Bellman-Ford algorithm (Bellman, 1958). In fact, DSDV (He, 2002) is inspired by the two legend routing protocols for the wired Internet, Routing Information Protocol (RIP) (Hedrick, 1988), and Border Gateway Protocol (BGP) (Lougheed & Rekhter, 1990). Routing in DSDV is achieved by exchanging information tables, whose contents include the shortest distance and the first node on the shortest path to every other node of the network. Distance in DSDV is measured by the number of hops. The tables are exchanged and updated at regular intervals, and also when a node observes a significant change in the local topology. An example of DSDV routing table is shown in figure 2.6.



(a) An ad hoc topology of 6 wireless nodes operating with DSDV.

Dest	NextHop	Dist	Seq No
1	2	2	33
2	2	1	28
4	4	1	42
5	4	2	57
6	2	3	81

(b) DSDV routing table for node 3.

Figure 2.6: Left (a) is an example of DSDV topology of 6 nodes, and right (b) is the routing table of node 3.

Once a routing information table is passed to a neighbouring node, an incremental update or full dump may occur. The former is used when a node does not observe significant changes in the local topology, whereas the latter is done either when the local topology changes dramatically or when an incremental update requires more than a single network protocol data unit (NPDU). When a node receives new routing information, that information is compared to the information already available from previous routing information packets. The comparison is made using sequence numbers which are assigned to each shortest path. Applying sequence numbers allows DSDV to tackle routing loops, one of the most common issues in distance-vector protocols. Any incoming route with a more recent sequence number is immediately adopted, while routes with older sequence numbers are discarded from the table. In the case where two routes have equal sequence numbers, the one with the best metric score is selected. As mentioned previously, the best metric may be the lowest number of hops.

The advantage of DSDV is its ability to provide routing solutions to all destinations at all times, ensuring less delay compared to other methods. However, such behaviour increases the control overhead, especially under conditions with high mobility. This also puts a negative effect on the scalability of the protocol. The larger the network size becomes, the more control overhead will be produced. The problem appears to both small networks with high mobility, and large networks with low mobility.

Optimized Link State Routing (OLSR). The protocol is an optimization of the classical link state routing approach (Forouzan, 2005; Adjih *et al.*, 2003a; Xu *et al.*, 2011) tailored to the requirements of a wireless ad hoc networks. It is similar to DSDV in a sense that they are both proactive and require routing information messages to be exchanged between nodes. However, in OLSR nodes are not required to possess knowledge for the whole network topology. What they know is partial information (neighbourhood only), which is collected by periodical beaconing from neighbouring nodes.

Furthermore, each node in the network selects a set of nodes in its one-hop neighbourhood which may retransmit its messages. These selected nodes are called multipoint relays (MPRs), and constitute the key concept of OLSR's multipoint relay mechanism. That is, when a routing information update is required, only MPRs are allowed to forward broadcast messages, reducing the control overhead of flooding. All nodes must select a MPR within their neighbourhood. The selection criterion of a MPR is that a message sent by any node should be repeated by the MPR and received by all nodes in the neighbourhood two hops away.

OLSR consists of two main components: the HELLO messages and the topology control (TC) messages. A HELLO message is periodically sent by all nodes, and contains information about their neighbours, the nodes they have chosen as MPRs, and a list of neighbours whom bidirectional links have not yet been confirmed. Exchanging HELLO messages not only allows the finding of possible pairs of connection, but also provides the required information in order to select MPRs in the first place. TC messages are used to share routing information about the current topology. They are periodically sent by nodes using the multipoint relaying mechanism. The payload of a TC message is a set of bidirectional links between a node and its neighbours. In terms of route calculation, the shortest path algorithm is used.

The advantage of OLSR over other proactive protocols such as DSDV is that it not only reduces the routing overhead associated with table-driven routing, but it also reduces the number of broadcasts done. On the other hand, OLSR remains prone to the common proactive routing vulnerabilities, such as incorrect control traffic generation and relaying (Adjih *et al.*, 2003b). Also, even though routing information updates are smaller than DSDV, they still need to be propagated over the whole network by the MPRs, which affect OLSR's scalability (Ge *et al.*, 2004; Nguyen & Minet, 2007).

Cluster head gateway switch routing (CGSR). This protocol (Chiang, 1997) uses a proactive routing scheme based on DSDV and is specifically designed for clustering in wireless ad hoc networks. Clustering allows the wireless channels to be effectively allocated among different groups of nodes (clusters). Each cluster is defined by the transmission range of the cluster head, that is, an elected node which can communicate directly with all the other nodes[2]. Although the complexity and the overhead of clustering rests in the selection of the cluster head, a controlled token scheme can be used within each cluster to give priority to cluster heads, in order to maximize channel utilization and minimize delay.

CGSR protocol also uses the idea of gateway nodes. A gateway is a node that belongs to more than one cluster. Using a table-driven algorithm CGSR improves efficiency by routing packets alternatively between cluster heads and gateways, before they reach their destination cluster head (and eventually the destination node). This method improves routing efficiency since cluster heads have more changes to transmit and gateways are the only nodes that cluster heads can forward packets to. Clustering provides a mechanism to allocate bandwidth, which is a limited resource in wireless ad hoc networks. Thereby, it improves bandwidth utilization. Additionally, CGSR improves its performance by applying token scheduling and code scheduling[3]. This allows the protocol to improve routing performance by routing packets through the cluster heads and gateways.

Since CGSR is a hierarchical routing scheme which enables partial coordination between nodes by electing cluster heads, it utilizes the bandwidth more efficiently. Although implementing a priority scheduling scheme with token and code scheduling is fairly easy, CGSR requires more resources in order to avoid gateway conflicts. Moreover, the power consumption at the cluster heads is also increased due to higher traffic rates of these nodes.

Global state routing (GSR). GSR (Chen & Gerla, 1998) protocol for wireless ad hoc networks wants the nodes to exchange vectors of link states among their

[2]Notice that this does not mean that nodes cannot communicate directly with each other in the same cluster.

[3]Assigning appropriate spreading codes to two different clusters.

neighbours during routing information exchange. In a pure link state fashion, once a node realises that a link has changed between itself and a neighbouring node, it floods the link state information into the whole network (global flooding). The link state information includes the delay to each neighbouring node. Unlike the traditional link state method, GSR does not flood the link state packets. Instead, every node maintains the link state table based on up-to-date link state information received from neighbouring nodes, and periodically exchanges its link state information with its neighbouring nodes only (no global flooding). Information is disseminated as the link state with larger sequence numbers replaces the one with smaller sequence numbers.

An advantage of GSR is that the time required to detect a link change is shorter than in other link state protocols. Furthermore, since the global topology is maintained in every node, preventing routing loops is simple and easy. A disadvantage is that GSR uses large size packets as update messages, which consume a considerable amount of bandwidth and lead to high latency rates of link state information propagation. In addition, large sized packets also consume more energy.

Fisheye state routing (FSR). Having its origins in GSR, the FSR (Pei *et al.*, 2000) routing protocol expands the idea of exchanging vectors of link states among adjacent nodes only to a more sophisticated exchanging scheme. Analytically, FSR uses the fisheye technique proposed by Kleinrock and Stevens (1971). This technique was developed to reduce the size of information required to represent graphical data. The eye of a fish captures with high detail the pixels near the focal point. The detail decreases as the distance from the focal point increases. In network routing and FSR, fisheye allows accurate distance and path quality information about the immediate neighbourhood of a node to be maintained, with progressively less detail as the distance increases. In comparison to GSR, FSR scales well to a large network size, while keeping the overhead low.

2.5.2 Reactive

Unlike proactive protocols, reactive (also called on-demand) protocols do not maintain routes between all the nodes in the wireless ad hoc network. Rather, routes are established when needed through a route discovery process in which a route request message is broadcast. A route reply is returned either by the destination or by an intermediate node with an available route.

Ad hoc On-Demand Distance-Vector (AODV). One of the most widely used and well-known examples of reactive routing is the AODV protocol (Perkins *et al.*, 1999). Although it is based on distance vector routing (Forouzan, 2005), AODV is designed to request for a route only when the route is needed. Additionally, AODV does not require nodes to maintain routes to destinations that are not actively used, reducing in that way both control messages overhead and power consumption.

The way this protocol provides routing solutions relies on the following concept. A node broadcasts a route request (RREQ) message when it determines that it needs a route to a destination and does not have one available. Such a situation can occur when the destination node is unknown or a previously valid route has expired. Hence, expiry time of each route in the routing table of a node plays a significant role to the routing mechanism. When a node receives a RREQ message it may send a route reply (RREP) back, if it is either the destination or it has a route to the destination with corresponding sequence number greater than or equal to that contained in the RREQ. Otherwise, it rebroadcasts the RREQ to others. Every time a route is used to forward a packet the expiry time of that route gets updated. In case of a broken link, nodes are notified with a route error (RERR) message, which gets propagated to all nodes in order to invalidate the broken route.

The main advantage of this protocol is that routes are established on demand, which offers less delay during the connections' set-up. However, the protocol's dependence on sequence numbers can drive the system to instability. Such a situation may happen when an out-of-date intermediate node has a higher but not latest destination sequence number for a path.

AODV is a single-path routing protocol. In single-path (or unipath) routing, only a single routing solution is selected in order to traffic packets between a source and a destination. This strategy allows the protocols to be susceptible to high network load, frequent route failures due to mobility and thus higher packet loss, and traffic congestions (Parissidis *et al.*, 2006). In order to have a more robust behaviour, in particular under networks with high mobility, multi-path routing is used. Unlike single-path, multi-path routing protocols are able to discover more than one routes during the route discovery process, and then utilize them in order to distribute the traffic. One multi-path routing protocol, based on AODV is discussed below.

Ad hoc On-Demand Multi-path Distance-Vector(AOMDV). This is an extension to AODV, in order to support multi-path routing proposed in Marina and Das (2001). Unlike AODV, where a route discovery process is able to find only one routing solution between a source and a destination, AOMDV is able to discover multiple alternatives. In order to achieve that, the following changes have been proposed for AODV.

One major change is that in AOMDV, nodes accept duplicate copies of RREQ messages and examine them in order to find alternative reverse paths. In order to guarantee loop freedom, AOMDV nodes only accept those RREQs which result in node or link disjoint paths. Node disjoint paths are those where no intermediate nodes participate to different paths, whereas link disjoint paths contain only unique links (Upadhayaya & Gandhi, 2011). Like AODV, AOMDV uses sequence numbers. The idea is that for each destination, each intermediate node can hold a cache of alternative paths which will be used as next hop in order to reach the appropriate destination. In terms of packet forwarding, AOMDV uses a simple approach. A link is used until it breaks, at which point an alternative is found from the cache. Additionally, AOMDV uses an optimization called packet salvation which is a mechanism borrowed by DSR (see below). By packet salvation, when a link fails the intermediate node may decide to forward the packet to an alternative path, instead of dropping it. Being a multi-path routing protocol, AOMDV is able to reduce both end-to-end delay and packet loss, and utilize the network topology more efficiently as the load is distributed across multiple routes.

Dynamic Source Routing (DSR). Closer to AODV, DSR (Johnson & Maltz, 1996) is another example of on-demand routing. This protocol is designed to eliminate the periodic update messages between nodes, thus the bandwidth consumed for this control overhead. It follows the idea of source routing (Forouzan, 2005; Postel, 1981). A routing entry in DSR contains all intermediate nodes to be visited by a packet, rather than just the next hop information maintained by DSDV or AODV. If the source knows the exact path to the destination, it puts it in the data packet's header and sends it off. In the opposite case, the source performs a route discovery. Route discovery is achieved by flooding the network with a RREQ message. The RREQ message gets updated as it collects new routing information from the nodes it visits on its way to the destination. Any intermediate receiving node which has an answer to the RREQ may reply with a RREP message. If not, it propagates the request further. Receiving nodes also use incoming RREQ messages to update their own routing tables. Also, the destination's reply is sent using the route recorded in the received RREQ.

To reduce the cost of route discovery in terms of control overhead, each node in the network maintains a cache of source routes it has learnt or overheard by the previously incoming RREQ messages (promiscuous mode). The cache is then used aggressively in order to limit the message propagation. DSR also uses RERR messages in order to inform the source node that an intermediate link is found to be broken. DSR also uses packet salvation, a mechanism of forwarding packets to alternative paths if they exist, instead of discarding them. However, the major disadvantage of this protocol is that its aggressive use of caches, as well as its inability to locally repair broken links, leads to stale routing information and cache pollution. As a consequence, data packets can be sent onto erroneous routes due to inconsistencies during the route discovery, adding extra delays and bandwidth wastage.

Associativity-based routing (ABR). This is a reactive approach which is combined with a beacon-based algorithm (Toh, 2002). In ABR, a source node floods RREQ messages throughout the network when a route is not available in its local routing cache. The RREQ messages are propagated by all the intermediate nodes, carrying the path they have traversed and the sequence number of the initial beacon. When the first RREQ reaches the destination, the destination does not immediately reply. Rather, it waits for a time period in order to receive multiple RREQs through different paths. The reason why ABR is an associativity-based algorithm is that after some time duration, the destination selects the path that has the maximum proportion of stable links. Special rules are applied in the case where two paths have the same proportion of stable links: the shorter of them is selected. In case of having more than one shortest path, a random path is selected as the path between the source and the destination.

One major advantage of ABR protocol is that stable routes have a higher preference compared to shorter routes. This results in fewer path breaks which, in turn, reduces the extent of flooding due to reconfiguration of paths in the network. On the other hand, the chosen path may be longer than the shortest path.

Dynamic MANET On-Demand (DYMO). This recently proposed (Chakeres & Perkins, 2012) reactive protocol is the successor of AODV, currently defined in an IETF Internet Draft. The route discovery in DYMO is similar to the previous routing protocols. The source node sends a RREQ message throughout the network to find a way to the destination. During the discovery process, each intermediate node records a route to the originator from the RREQ. When the destination receives the RREQ, it responds with a RREP sent hop by hop toward the RREQ originator. Once the RREP reaches the originating node, a route has been established between the two in both directions. The maintenance of the network routing in DYMO uses characteristics of AODV, when the lifetime of a route entry depends on an expiry time value.

In DYMO, the route lifetime is extended upon a successful forwarding of a packet. Moreover, RERR messages are used to notify the source when a route is unknown or broken. When a node receives an RERR message, it deletes the invalid route from its routing table and the route discovery is triggered.

2.5.3 Hybrid

A hybrid protocol in wireless ad hoc networks is an adaptive routing protocol that combines the best features from both reactive and proactive techniques. Generally, hybrid protocols separate the network topology in zones. Routing is determined proactively within each zone, and reactively outside it. The advantage of such a combination is the increased overall scalability and optimization within the zones.

Zone routing protocol (ZRP). This was the first hybrid protocol to be proposed (Beijar, 2002). As in most of the hybrid protocols, the key concept employed in ZRP is to use a proactive routing method within a limited zone of a predefined sized neighbourhood (intra-zone routing protocol or IARP), and a reactive method for nodes beyond this zone (inter-zone routing protocol or IERP). Each node maintains information about routes to all nodes within the same zone by exchanging periodic route update messages. Hence, the larger the zone, the higher the update control message overhead. On the other hand, the reactive method (IERP) is responsible for finding paths to the nodes which are not within the same routing zone. If the destination node of a transmission is outside the source node's zone, the reactive part of the protocol is triggered, broadcasting a RREQ message to its peripheral nodes. RREP messages are also used to answer the RREQs. RREP messages contain information about the nodes that forwarded the RREQ message, thus, they carry limited knowledge of each intermediate zone's topology. This allows the paths that are smaller to be stored and can, in principle, be more robust since they are not affected by the internal routing within each zone.

By combining the best features of proactive and reactive routing methods, ZRP reduces the control overhead compared to the RREQ flooding mechanism of the pure on-demand, and the periodic flooding of the pure table-driven protocols. Also, the decision on the zone radius has a significant impact on the performance of the protocol.

Adaptive distance vector (ADV). Another protocol which combines both proactive and reactive techniques is the ADV (Boppana & Konduru, 2001). It is a distance vector routing algorithm that exhibits some on-demand characteristics by varying the frequency and the size of the routing updates, in response to the network load and mobility conditions. Rather than advertising and maintaining routes for all receivers of any node of the network, ADV advertises and maintains routes for nodes that participate in any currently active connection. This not only reduces the size of the routing updates, but also the size of the routing tables maintained.

Furthermore, ADV uses the concepts of trigger meter and trigger threshold. A trigger meter is a special variable that is incremented whenever one of the following events occurs; a) a node has some data in its buffer due to lack of routes, b) one or more nodes in a neighbourhood make a request for fresh routes, or c) a node needs to advertise a valid or invalid route to the destination, in order to keep a current route up-to-date. Once one of the above triggers occurs, the trigger meter changes and is compared to a trigger threshold. The latter is used to decide when an update needs to be triggered. It is a dynamically changed variable, based on the recent history of trigger meter values at the time of previous updates.

Experiments have shown that ADV outperforms on-demand protocols like AODV and DSR in many instances (Boppana & Konduru, 2001). The authors show that the improvement is significant when the node mobility is high.

2.5.4 Choosing an appropriate approach

Obviously, there is no fixed and straight-forward decision of which of the three approaches can be considered the best. The fitness of each one depends on many factors, such as the size of the network, the mobility of the nodes, the data traffic, the applications running on the top of the network stack, and so on.

The proactive approach tries to maintain routes to all possible destinations at all times. Routing information is constantly propagated and maintained by all nodes in the topology, meaning that in principle the topological knowledge that nodes have includes all recent changes. Also, utilizing the whole topology in order to find available paths between a source and a destination implies that multiple solutions may be found. This is a positive aspect, since alternative paths can be found easily in case of failure. Nevertheless, in very dynamic networks with high density and mobility, proactive protocols can become inefficient or even break down completely. For this reason, proactive routing protocols tend to work better for wireless sensor networks where sensor nodes have fixed positions.

In contrast, reactive protocols initiate route discovery on the demand of a data transferring session. This routing approach can dramatically reduce the control overhead. However, the source node has to wait until a route to the destination is discovered, increasing the response delays. Reactive protocols adapt better to topological and network size changes, which allows them to be more scalable than proactive ones. In MANETs, where all nodes are mobile and the links between them break frequently, reactive protocols are generally preferred (Broch *et al.*, 1998a).

Combining proactive and reactive routing together, hybrid routing protocols can be quite scalable and provide fast routing solutions within zones, while at the same time they balance the amount of control overhead required to maintain routing, especially for low mobility networks (e.g., some WSN deployments).

2.6 Nature-inspired protocols

The main contribution of this section is a survey of the existing routing protocols whose source of inspiration is driven by nature (Kumar & Kumar, 2011). Today's communication networks have become enormously complex systems. New technologies and applications require a large number of communicating and interacting entities. MANETs and WSNs provide the network infrastructure for such applications, where entities must work together in a decentralized and collective system. Also, due to resource constraints discussed in section 2.3.6, the complexity required within each entity needs to be minimal. Similar examples of such complex and decentralized systems are found in nature and, in particular, in insect societies.

2.6.1 Swarm Intelligence - ACO & BCO

The term Swarm Intelligence (SI) (Fleischer, 2005) has come to represent the idea that it is possible to control and manage complex systems of agent-like entities, which based on their multiple interactions with the environment and each other, are able to provide solutions to difficult problems. The underlying features of SI are based on observations of social insects. For instance, ant colonies and bee hives are populated by small creatures which, despite their size and simple capabilities as single units, are known for their surprisingly interesting achievements. Finding and storing food, producing goods such as honey, wax or propolis, mating and protecting their young, and guarding the nest are some of their daily activities. The way insects conduct their affairs is organized collectively. The fundamental question asked in the area of SI is how such swarms of creatures with relatively low brain power and communication capabilities can build such emergent behaviour and collective intelligence that seem to exhibit a more global purpose?

Under the umbrella term of SI there are two main meta-heuristics of research interest in connection to designing network routing protocols.

Ant Colony Optimization

The Ant Colony Optimization (ACO) (Dorigo & Stutzle, 2004) meta-heuristic has been inspired by the operating principles of ants, which allow them to perform complex tasks such as foraging and nest building. In particular, ACO focuses on the ability of ants to find the shortest path between their nest and an interesting source of food, and constitutes of two components: stigmergy (Theraulaz & Bonabeau, 1999) and pheromone control (Schoonderwoerd *et al.*, 1996).

Finding food is a complex problem and cannot be solved by a single insect. Foraging ants need to communicate with each other and share information about possible sites of interest. Communication is achieved by stigmergy, where ants exchange information indirectly through the environment. Trails of a volatile chemical substance called pheromone are laid down by foraging insects on their way between the nest and the source of food. As time passes, fellow ants sense the pheromone and take the path increasing the pheromone concentration on it. As a result, the foraging power for the particular path is increased, and it becomes in favour of the foragers. An example is illustrated in figure 2.7.

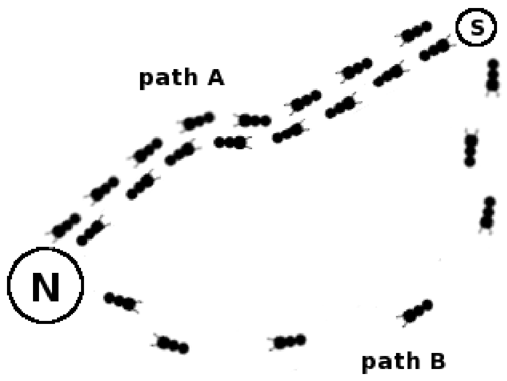


Figure 2.7: Stigmergy plays an important role in ant foraging. Here path A is selected as the optimal path between the nest (N) and the source of food (S). As more ants use the specific path, its pheromone concentration increases affecting the recruitment process.

In Bonabeau et al. (2000), the authors pointed out that using stigmergy does not imply that the shortest path will be found. If ants initially choose a non-optimal path, laying pheromone trails will enforce them to use it, as long as the food is available. Therefore, ACO-related routing algorithms propose a number of pheromone control strategies that tackle this problem. The two most common ones are pheromone evaporation and ageing. Evaporating means that the pheromone value of a path is decreased over time by a constant factor. In case of a broken path, ants will no longer be able to use it and therefore the concentration of pheromone will eventually drop to zero. On the other hand, ageing decreases the pheromone value taking into consideration the age of the ant. An older ant will lay down less pheromone than a younger one.

Bee Colony Optimization

Similar to ACO, the Bee Colony Optimization (BCO) (Teodorović *et al.*, 2011; Lucic & Teodorović, 2001; Lucic & Teodorović, 2002) is a nature-inspired meta-heuristic, which can be applied in order to find solutions for difficult combinatorial optimization problems. It is based on the idea that solutions to a given problem are built from scratch, within a number of certain execution steps, therefore, is iteration-based. Each iteration involves three steps. The forward pass, the backward pass and the final decision.

Once the problem is defined, the forward pass starts by artificial bees seeking a partial solution to the problem. They start exploring the search space, collecting solution components. After finding enough information, they return back to the hive. Here, the hive is not a well defined object and has no precise location. Rather, one needs to think of it as the synchronization point where bees meet and exchange their knowledge of the current state of the search. When bees are back in the hive, the partial solutions that they have obtained are shared between the searchers. This is the backward pass during which the collected solutions are evaluated based on their quality. Each artificial bee has to make a decision based on a certain probability, whether it will continue searching following its own path, or switch to a fellow searcher's solution. BCO steps are alternating in order to reach a point where all possible solutions are complete. The final step is to decide which solution is the best, based on certain criteria. The result is the iteration's contribution to the global best solution. After a number of iterations (determined again by stopping criteria), the global best solution is met.

BCO meta-heuristic is used to solve several problems in different domains (Wedde *et al.*, 2010; Lucic & Teodorovic, 2003; Teodorović & Orco, 2005). In this thesis, the discussion is limited to the application on telecommunication networks and, in particular, wireless ad hoc networks routing.

2.6.2 Ant-inspired routing protocols

As in traditional approaches in designing routing protocols, the first attempts of applying the ACO meta-heuristic were aiming for wired, packet-switched telecommunication networks. Therefore, it is historically fair to briefly mention some of these protocols before presenting those aimed for MANETs and WSNs.

One of the earliest routing protocols was the ABC protocol for packet-switched networks, proposed by Subramanian et al. (1997). There are two types of ants involved in this protocol; regular and uniform ants. A regular ant updates the pheromone values of the routing tables based on the accumulated cost of travelling to a particular node, whereas a uniform ant is randomly selecting its next hop and it updates the pheromone values of the routing table based on the costs in the direction opposite to its travel.

Another very famous routing protocol for wired environments is AntNet (Caro & Dorigo, 1998). As in ABC, AntNet involves two types of ant agents. Each ant is able to store the node addresses and trip time estimates of its visits. At regular intervals, a forward ant is launched from a source to its specific destination, depending upon the amount of traffic generated for the destination at the source. Once a forward ant reaches its destination, it becomes a backward ant and transfers all the information back to the source. Backward ants visit all intermediate nodes the forward ants had visited, but in reverse order. Also, a backward ant is allowed to update entries in the routing tables of the intermediate nodes. The goodness of each path is defined based on the trip time. Di Caro and M. Dorigo conducted a number of experiments on different topologies (Caro & Dorigo, 1998). Their experiments have shown that AntNet outperforms, with respect to throughput and delay, all other competitors which consist of proactive, reactive and hybrid approaches (Dhillon *et al.*, 2007).

AntHocNet. This is a hybrid algorithm proposed by Ducatelle (Ducatelle, 2007), Di Caro and Gambardella (Di Caro *et al.*, 2005). It explores the capabilities of the ACO mechanisms, combining them with both reactive and proactive ways of gathering and building routing data. When a data session is started between a source and a destination node, the source checks whether it has up-to-date routing information for the destination. If it has not, it reactively sends out an ant-like agent, called reactive forward ant, in order to look for paths to the destination. Therefore, a forward ant is used to gather information about the quality of the path it follows. Once it reaches the destination, it traces back the path to the source node, updating the routing tables in its path. On its way back, the forward ant becomes backward ant.

In AntHocNet a routing table consists of a destination, the next possible hop to it and a pheromone value. The latter indicates the estimate goodness of a path between a source and a destination. In this way, pheromone tables in different nodes indicate multiple paths between two nodes in the network and are stochastically spread over it (in each node they select the next hop with a probability proportional to its pheromone value). Once paths are set up and the data starts to flow, the source node starts to send proactive forward ants to the destination. This is a maintenance phase where each proactive forward ant follows the pheromone values in the same ways as the data, but has a small probability at each node of being broadcast. This technique serves as follows. If the forward ant reaches the destination without a single broadcast, it means that the current path is working, and it provides an efficient way of data transferring. On the other hand, if the ant is broadcast at any point, it leaves the currently known pheromone trails and it explores new paths. A threshold is used to avoid proactive forward ants being broadcast to the whole network, allowing the search for improvements or variations to be concentrated around the current paths. In the case of a link failure, a node may use an alternative path based on the pheromone values. However, if the failed link was the only one in each pheromone table, the node sends out a route repair ant that travels to the involved destination like a reactive forward ant would do. Simulation experiments have shown that AntHocNet can outperform AODV in terms of delivery ratio and average delay (Ducatelle, 2007).

Ant-colony based routing algorithm (ARA). Based on the same idea as AntNet, Gunes et al. proposed ARA (Gunes *et al.*, 2002), a purely reactive protocol. ARA consists of three phases. Starting with the first one, the route discovery phase is when new routes are created by the use of forward and backward ants. A forward ant is responsible for establishing the pheromone track to the source node. In contrast, the backward ant establishes the pheromone track to the destination. Furthermore, a forward ant is broadcast by the sender and relayed by its neighbouring nodes. Each node receiving such an ant message for the first time, it creates a record in its routing table. When a forward ant reaches its destination, it is destroyed and a backward ant takes over, travelling back to the source. Subsequently, once the sender node receives a backward ant, it is said that a routing path between them is established and data packets can be sent.

The second phase of ARA is the route maintenance. This phase is responsible for maintaining and improving the routes during the communication. ARA uses the concept of pheromone strengthening and evaporation. When a node relays a data packet, it increases the pheromone value of the corresponding path. In addition, pheromone evaporation is achieved by decreasing the value over time.

The third and last phase of ARA is called route failure handling. It is the phase which handles routing failures, usually caused by mobility. A broken link is detected by a missing acknowledgement packet. Once a link is detected as broken, the corresponding nodes get an RERR message and deactivate it by setting its pheromone value to zero. Nodes try to find an alternative routing by either looking at the routing table, or starting phase one again.

In terms of performance, simulation experiments have shown that ARA is very close to the performance of DSR, but generates less overhead (Gunes *et al.*, 2002).

Termite. The Termite protocol (Roth & Wicker, 2003) takes into consideration the ability of social insects to self organize and is based on the concept of the termite hill building (Roth & Wicker, 2005) behaviour of these little creatures. Hill building illustrates the four principles of self-organization of the ant-like societies, i.e., positive feedback, negative feedback, randomness and multiple interactions.

Termite associates a specific pheromone scent with each node in the network. Packets moving through the network are biased to move in the direction of the pheromone gradient of the destination node, exactly as biological termites are biased to move towards their hill. Positive feedback is gained from links that have stronger pheromone scent, whereas negative feedback is represented by pheromone evaporation. The randomness factor is used for termites that explore the network for the first time, and of course, multiple interactions are achieved by having multiple termite agents exchanging information as they pass through intermediate nodes.

The argument of Termite is that the SI framework can be used to competitively solve the routing problem in mobile wireless ad hoc networks, with a minimal use of control overhead. A small amount of control information is piggybacked in every data packet, which is usually sufficient for the network to maintain a current and accurate view of its state. Using that information the routing algorithm is able to generate routing decisions.

HOPNET. Hybrid in nature, the HOPNET (Wang *et al.*, 2009) protocol uses ant colony principles to provide routing solutions in MANETs. Similarly to ZRP (explained in section 2.5), the network is divided into zones, each one defining the neighbourhood of a node. Moreover, each node keeps routing information for paths to nodes within its neighbourhood (intrazone routing) and for paths to nodes outside its zone (interzone routing).

Intra-zone routing is done proactively. That is, nodes periodically send forward ants in order to discover available paths to all its neighbours. Each routing information to a neighbour node is assigned to a pheromone value, representing the pheromone concentration on the link. HOPNET uses pheromone evaporation as the method to control the pheromone levels. In addition, each node keeps hop count information for all other nodes in its neighbourhood. By this, it is able to know which nodes are at the borders of its zone.

Inter-zone routing is available when the intra-zone routing fails to provide a routing solution. An intra-zone route discovery starts by the source sending an forward ant. Since the node already knows which are its zone's border nodes, the forward ant is sent directly to them. At the boundary, the receiving node checks to see if the destination is within its zone. If not, it forwards the ant to its own border nodes. Upon reaching the required destination, a backward ant is created and sent using the forward ant's path in traverse.

Comparison experiments indicate that HOPNET is quite stable for high and low mobility networks, and is highly scalable as the network size has no impact on the protocol's performance.

T-ANT. Proposed by Selvakennedy et al. (2006), this protocol provides clustering and routing in WSNs. It uses the ACO meta-heuristic in order to aid cluster head election in such a way that optimal data aggregation with minimized overall energy dissipation can be achieved.

During a T-ANT cluster set-up, a sink in the WSN releases a number of ants. Each ant takes a random direction and walks in the network topology as deeply as its time-to-live (TTL) value allows. At each intermediate sensor node, the ant takes a random direction again. This continues until the TTL value becomes zero, when the ant stays and waits at the node. The sink may send off new ants at some time interval, and the whole process of initialization is controlled by a timer. Once the timer expires, the sensor node that has an ant becomes the cluster head.

A new cluster head starts advertising to its neighbouring nodes by broadcasting a message with its own node identification number (ID), and a new TTL value to restrict the message's propagation. Nodes that receive such messages extract valuable routing information, such as the cluster head ID, the sender's ID and the hop distance to the cluster head, and broadcasts it further. All this information is then used to decide which cluster to join, once a join-timer expires. The pheromone level of each path to the cluster head is based on the total hop distance, the number of cluster heads in the neighbourhood and its residual energy level. Pheromone control in T-ANT is achieved by using the idea of anti-pheromones. Before the ant leaves its current node, an amount of anti-pheromone is laid to mimic a rapid decay of pheromone level on the path.

The authors of T-ANT have conducted comparison experiments to other related work, such as LEACH (Heinzelman *et al.*, 2002) and TCCA (Selvakennedy & Sinnappan, 2005), the results of which show that T-ANT is able to store less state overhead in memory, providing decentralized and robust clustering with no position knowledge of the sensor nodes.

2.6.3 Bee-inspired routing protocols

Improving existing ant-inspired protocols, such as AntHocNet is a research objective frequently found in the literature. Unlike the ACO meta-heuristic, attempts to apply bee-inspired principles in the area of telecommunication networks have recently started. To the author's knowledge, BeeHive (Wedde *et al.*, 2004b; Farooq, 2006) introduced by Wedde, Farooq, and Zhang is the first routing protocol for wired packet-switched networks. For historical reasons, a brief overview of BeeHive is given below, followed by a list of the bee-inspired routing protocols that exist in the current literature.

The BeeHive protocol is built around two types of agents, the short distance and the long distance honeybee agents which are retroactively generated at the nodes, and are designed to mimic the way honeybee foragers respond to bee dances (the concept of the bee dance is explained in chapter 3). The responsibility of both types of agents is to explore the network and evaluate the quality of the paths that they traverse, in order to build nodes' routing tables. Short distance agents are allowed to move only up to a restricted number of hops in the network, whereas long distance agents have to collect and disseminate routing information in the complete topology. Moreover, BeeHive adopts a hierarchical organization for the network. The design enables intelligent agents to explore network regions, or foraging zones, and collect information which is then deposited to local routing tables. Each foraging zone has a representative node, which is the node with the lowest IP address within its region. Its role is to launch long distance honeybee agents that, as already described, their purpose is to collect information for the complete topology.

Apart from the network routing, BeeHive's contribution is also the first attempt to map nature to networks, using the following simple mapping rules:

- Each node in the network is considered as a hive that consists of honeybee agents that can behave exactly as real honeybee scouts do.
- These artificial scouts provide the nodes they visit with information about the propagation delay and queuing delay of the paths they explored. These two pieces of knowledge are then used internally to estimate the goodness of a neighbouring node.
- A honeybee agent decides whether to provide its path information to a node based on the quality of the path, examined against a threshold. This threshold depends on the number of hops a honeybee agent is allowed to take.
- A routing table is considered as the dance floor where honeybee agents provide information about the quality of the paths.
- The quality of paths is mapped onto the quality of nodes. Consequently, the quality of a node is formulated as a function of proportional quality of only those neighbouring nodes that possibly lie in the path towards the destination.
- Data packets in the network are considered as foragers. They also access the information in the routing tables of any node they visit, which enables them to become aware of the quality of different neighbouring nodes for reaching their destinations. Thus, they select the next neighbouring node toward the destination in a stochastic manner, depending upon its goodness.

BeeAdHoc. Extending BeeHive to fix MANETs needs, Wedde, Farooq et al. designed BeeAdHoc (Wedde *et al.*, 2005b) with the aim of creating a routing protocol which is at the same time energy-efficient and provides performance comparable to those of existing state-of-the-art in the area.

There are three types of agents (packets) involved in BeeAdHoc. Initially, a packer agent represents a food-storer bee that resides inside the network node.

The main task of a packer agent is to act as an interface between the transport layer and the network layer, receiving data packets from and store them in the upper layer. It is also responsible for finding a forager (will be explained later below) for the data packet at hand.

Next, BeeAdHoc uses scout agents to discover routes between sources and destinations. In a reactive way, when a route is required at the destination node, a scout agent is broadcast to the network just like a forward ant in ant-inspired protocols (see section 2.6.2). It also carries a unique path key based on its ID and source node, which is used to allow loop-free routes. Like backward ants, a scout is sent back to the source, following the new discovered path in traverse. Once a scout returns to its source node, it recruits foragers by using a metaphor of bee dance, as bees do in nature (see chapter 3 for details on bee dancing).

Finally, A forager is the bee agent that receives data packets from a packer and delivers them to their destination, in a source-routed modality (for source routing, see DSR description in section 2.5.2). Since BeeAdHoc is a power-aware routing protocol, there are two types of foragers. The delay and the lifetime. A delay forager gathers end-to-end delay information, while lifetime foragers gather information about the remaining battery power of the nodes.

In order to support UDP transmission, BeeAdHoc uses another special agent type, the beeswarm agent. A beeswarm is used to explicitly transport foragers back to their source nodes, where no acknowledgements are sent for the received data packets. The way by which this objective is achieved, is by comparing the number of the incoming foragers from a source to the number of existing waiting foragers at the destination received by that source. If it is above a threshold, then a beeswarm agent is launched in order to move foragers back to their hive. A forager is put at the header of a beeswarm packet, and the rest to its payload.

The architecture of the BeeAdHoc is designed in the way that all agents work efficiently. After extensive simulations, authors show that BeeAdHoc consumes significantly less wireless network card energy as compared with DSR, AODV, and DSDV.

BeeSensor. M. Saleem along with the authors of BeeAdHoc proposed BeeSensor (Saleem & Farooq, 2007b) for WSNs. Being inspired by the two previously described protocols, BeeSensor is mainly focusing on minimizing the energy costs. There are three bee agents involved. Namely, the packers, scouts and foragers.

As in BeeAdHoc, packers are located within the nodes. At the sensor nodes their purpose is to receive data from the transport layer and load them to an appropriate forager. At the sinks, packers recover data from the incoming foragers and deliver them to the transport layer.

Route discovery is achieved by using forward scouts and backward scouts. In more detail, when a sensor node (source) requires a route to a destination node (sink), and no previous routing information is available, a forward scout is sent. The forward scout carries information about the sensed event, which is expected to trigger the interested sink nodes. In addition to that, a forward scout also collects information regarding the remaining energy of the nodes it visits. Upon reception at the sink node, a backward scout is created and sent to the initial source.

Finally, forager scouts are doing the real data delivering. They receive data from the packers and are sent to the source on a hop by hop basis. One key characteristic of BeeSensor is that it does not use source routing like BeeAdHoc does. Rather, each forager is assigned with a path at the beginning of its journey, and at the intermediate nodes, it is forwarded based on it to the correct adjacent node. Therefore, BeeSensor forager header size is kept constant, enhancing its scalability. BeeSensor's performance is compared to an energy optimized version of AODV which is distributed with RMASE framework (Saleem & Farooq, 2007a; Zhang, 2012), part of the Prowler wireless sensor network simulator (Prowler, 2012). Results show that the bee-inspired protocol is able to transmit more packets than the improved version of AODV, achieving less control overhead and lower energy consumption.

BeeAdHoc and BeeSensor are the only two existing routing protocols, proposed for MANETs and WSNs respectively. They share common characteristics borrowed from BCO, such as the principle design of using bee-inspired agents in order to discover routes and maintain available paths. Another common characteristic is that they are both energy-aware routing protocols, designed to maintain routing by mainly considering the remaining energy of the participating nodes.

On the contrary, the proposed bee-inspired design of this research work focuses on an extended number of evaluation metrics, in order to measure the goodness of each link, and thus of each path between a source and a destination. In addition, the proposed design focuses on mapping the ability of real honeybees to constantly monitor and judge their finding based on previous knowledge, as will be discussed in chapter 3. This unique ability of honeybees is mapped to networks by using statistical tools, which allow the artificial honeybee dance to highlight the strongest candidates between alternative paths, and let artificial foragers (packets) to be transmitted across them in an efficient way. Moreover, in this work, the path selection mechanism is designed to be flexible and accept implementation changes easily, based on the application needs and requirements. This is a mechanism inspired by a behaviour of honeybees during the dance and, in particular, the ability of foragers to decide which is the most suitable dancer to follow depending on the commodity that is running low.

2.7 Alternative mechanisms for routing

This section discusses several alternative ways of providing routing solutions in mobile ad hoc and wireless sensor networks. Apart from the most common classification (reactive, proactive or hybrid), Internet-inspired routing is also classified based on each distinct strategical approach being applied. Solving the routing problem is a complex task which often requires several approaches to be adapted. Therefore, routing protocols do not exclusively fall into one class.

2.7.1 Location-aware routing

With the availability of location information technologies such as the global positioning system (GPS) (Hofmann-Wellenhof *et al.*, 2001) or compasses, routing solutions can be calculated according to geographical position of nodes. Several authors have proposed relevant work, examples of which are given below.

Location-aided routing (LAR). Proposed by Y. Ko (2000), the LAR protocol assumes that each node in the network topology is equipped with a GPS system. The key idea of the protocol is that at the beginning of a communication session the source node makes use of the location information as well as node mobility information of the destination node, in order to define the correct portion of the network topology where route discovery packets will be sent.

A source defines two topological regions. The first, termed ExpectedZone, is where the destination node is expected to be based on previous location and mobility information. If no past information is available, the entire topology is considered. This implies that as more knowledge is accumulated, the Expected-

Zone gets narrowed down with more accuracy. The RequestZone on the other hand, is the region which the control packets for finding the destination are allowed to flood. The sender determines this area by setting its boundaries at the header of the RREQ message. Timers control the discovery process and repeat the sending of RREQ messages with increased RequestZone, until the destination is eventually found.

LAR protocol is designed to reduce the amount of control overhead. Instead of flooding the entire network, the protocol allows route discovery messages to be propagated only within predefined geographical regions. However, the use of GPS infrastructure is required, which makes the protocol inappropriate where access to location information is absent.

Trigger-based distributed QoS routing (TDR). This reactive routing protocol (De *et al.*, 2002) was designed to support real-time applications in MANETs. Similarly to LAR, a selective forwarding mechanism for route discovery is applied. Taking advantage of its cached knowledge regarding geographical positions, the source node forwards its route requests only to specific neighbouring nodes towards the destination. For this, a GPS-based underlined system is used on each node.

In order to satisfy Quality of Service (QoS), TDR takes into consideration the receiving signal strength of each packet to assess the link quality between neighbouring nodes. Control packets are propagated only through those neighbours from whom the signal strength of previously received packets exceeded a threshold. An acknowledgement message is sent back to the source, once a successful route is discovered. Moreover, re-routing may take place when the signal strength of a transmission at an intermediate node is detected to be below a satisfactory value.

Like the previously described LAR, TDR uses GPS information to selectively flood portions of the network topology, reducing in that way the control overhead. However, re-routing is directly related to the receiving signal strength of packets, which can be affected by a number of wireless telecommunication phenomena (as described in section 2.3.1), and vary rapidly causing an increase to the number of route discoveries.

2.7.2 Energy-aware routing

Due to the growth of applications that require wireless ad hoc infrastructure and, in particular, WSNs, energy consumption has become a serious factor to be taken into consideration when designing new routing protocols. The size of the nodes as well as the nature of the wireless telecommunication brings the resource constraint problem to the front. Typically, energy-aware routing protocols calculate routes by considering the minimum total transmission power wastage over the path between the source and the destination (Banerjee & Misra, 2002).

Power-aware source routing (PSR). This protocol's (Maleki *et al.*, 2002) objective is to extend the life of MANETs. The authors address the problem of services being stopped within a MANET because a number of the nodes die out, dooming the rest of the topology. PSR is inspired by DSR, in terms of exchanging routing messages (RREQs and RREPs). However, the key difference is that despite DSR where the shortest path is selected, PSR considers the remaining energy to decide if a node will continue serving as part of the routing solution.

Route maintenance in PSR focuses primarily on monitoring energy depletion. While data packets are being transmitted, each intermediate node constantly monitors its remaining energy and the cost of it being used as part of the particular route. Then, nodes whose energy cost is beyond a threshold generate an RERR message and send it to the source as if the route has become invalid. Consequently, a source which receives such a message starts a new route discovery process. Such a strategy allows the protocol to utilize the topology better in terms of energy consumption, and is found to increase the life time of the whole network.

Energy aware routing. The authors of Vidhyapriya and Vanathi (2007) have proposed this routing protocol that provides routing solutions, taking into consideration the energy availability and the received signal strength of nodes in the WSN. The protocol assumes that sensor and sink nodes have static positions and are known to each other. It also assumes that links are bidirectional and symmetric[4] (refer to section 2.3.2 for more details).

During the route discovery process between the sink and source nodes, an energy-sufficient mechanism is applied. A node which receives a RREQ message

[4]Symmetric links have similar characteristics no matter the direction of the data stream.

has to initially consider its remaining energy. If it is found to be insufficient, the node decides not to take place in the potential communication session and discards the RREQ message. In the opposite scenario, the node calculates the receiving signal strength of the RREQ message. Then, a common assumption is made. Due to the wireless signal propagation, the weaker the signal, the farther the node will be from the sender. The protocol eliminates the number of control packet retransmission by allowing only the farther nodes (from the sender) to retransmit the RREQ message. In the end, the discovered path consists of only those nodes that have sufficient energy to participate to the data transmission.

The authors have run simulation-based experiments to compare the protocol's performance, and have shown that it consumes less power than the state-of-the-art AODV under realistic cases.

2.7.3 The cross-layer approach

Alternative mechanisms for routing are often achieved by approaching the problem in a cross-layer manner. Cross-layering versus the legendary layered approach in network designing has been a hot topic for debate, especially between researchers of strong background in wired-related networking (Conti *et al.*, 2004). Strict layering ensures that interactions between layers remain controlled, and protocols designed in that way can be easily added or removed in the network protocol stack of each deployment. Another strong argument opposed to cross-layering, is that it may produce spaghetti-like code that is very hard to maintain from a purely software development point of view. On the other hand, the ad hoc research community recognizes that a strict layering approach does not provide the level of adaptation required today (Srivastava & Motani, 2005).

The layered, waterfall-like architecture simplifies development of different components by keeping each layer isolated from the others. Originated from the wired networks' world, the concept of transparency is what makes OSI, TCP/IP and IEEE 802 models allow rapid and universal development and improvements (Forouzan, 2005). Figures 2.8 and 2.9 illustrate the differences between layered and cross-layered design.

Nevertheless, it has become evident that the traditional layered approach that separates routing, flow control, scheduling, and power control is suboptimal in the realm of wireless ad hoc networks. This can be attributed to the complex

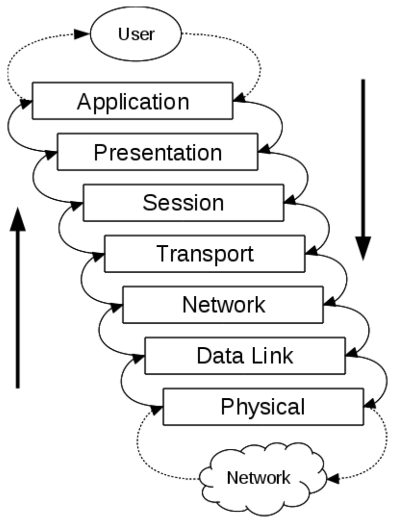


Figure 2.8: Traditional OSI waterfall fashion in network layer stack. Each layer is allowed to communicated only with one specific layer above or below (solid lines), and messages travel upwards or downwards by respecting the strict boundaries between layers.

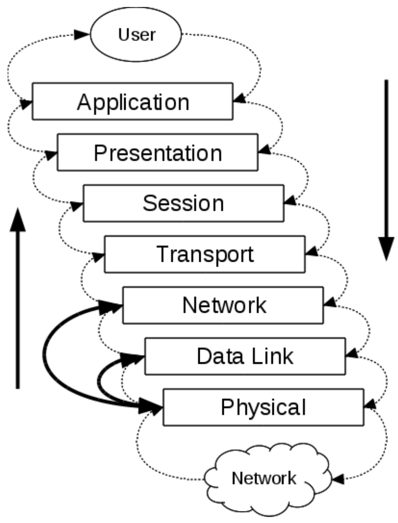


Figure 2.9: Cross-layered design allows different layers to communicate with each other and dynamically exchange values in order to offer optimized results. Here, apart from the waterfall connections (illustrated by the dashed lines) between adjacent layers, messages are dynamically exchanged between network and physical layers (solid thick lines). In addition, the data link layer is allowed to dynamically use values from the physical layer in order to optimize its behaviour, before or after forwarding the message up or down the stack.

and unpredictable nature of the wireless medium. Thus, the need for adaptation in network protocols remains high. In order to tackle the problems faced in ad hoc networks, a cross-layer design (Ibnkahla, 2009; Perez-Neira & Campalans, 2009) is desired to optimize across multiple layers of the stack. The basic idea of cross-layering is to make information produced or collected by a protocol available to the whole protocol stack, so as to enable optimization and improve network performance.

Until now several approaches have been proposed by researchers that use cross-layering in order to improve and optimize different network mechanisms. In most of the cases, the cross-layer design takes place between the MAC and the PHY layers. However, there are a number of recent examples that illustrate the benefits of having other layers jointly designed, such as network-data link layer, or even application-network.

Cross-layering offers a great degree of optimization in order to obtain the highest possible adaptivity. The following example protocols illustrate how adaptation is achieved by combining knowledge from different layers. These protocols are just a sample of the relevant literature. However, they prove that the cross-layer approach has a great potential in designing network protocols.

2.7.4 Cross-layer designed adaptive protocols

In order to bypass the resource constraints, Shah and Rabaey (2002) proposed an energy-aware routing protocol that uses a set of suboptimal paths occasionally in order to increase the lifetime of the network. The idea is that paths are chosen by means of a probability, which depends on how low the energy consumption of each path is. The energy consumption is a result of signal strengths, a piece of knowledge that can be found at the PHY layer of the stack. Hence, a cross-layer design allows access to the information and uses it in the routing layer (network layer) to make analogous decisions.

Cross-layer has also been a great help in designing cost-aware routing approaches. Suhonen et al. (2006) have proposed a protocol that uses cost metrics to create gradients from a source to a destination node. The cost metrics consist of energy, load (at nodes), delay, and link reliability information that provide traffic differentiation by allowing several routing choices based on delay, reliability or energy.

The MobileMAN (Borgia *et al.*, 2004) framework is one example of full cross-layer design which offers the ability of sharing knowledge between different layers, while it does not immediately violate the OSI structure. Its design is based on the concept of network status (NeST). NeST can be visualized as a vertical module which can be bidirectionally associated with each of the layers, and controls all cross-layer interactions through data sharing. Specifically the NeST module acts as a repository for information collected by the network protocols. In order for a protocol to deposit or use any information stored in that repository, it only has to implement a specific interface defined by the module. MobileMAN uses the cross-layer approach in a more secure and safe way, compared to other examples.

In Kozat et al. (2004) the authors present a framework for cross-layer designed efficient communication. This approach is characterized by a synergy between the PHY and the MAC layers, with a view towards inclusion of higher layers and, in particular, the network layer. In the following, the reason for having these layers working together in a joint design is explained. The key parameters of the PHY layer such as the transmission power, modulation, coding rate and the antenna beam coefficients have a direct impact on the media access control of the nodes in the wireless channels. Local adaptation of these parameters leads to better bit error rates in the PHY, which in turn affects the way routing and MAC decisions are made. Furthermore, as stated earlier the MAC layer is responsible for scheduling the transmissions and allocating the wireless channels.

While the concurrent transmissions create mutual interference, the time evolution of the scheduled transmissions ultimately determines the bandwidth allocated to each transmitter and the packet delays. Obviously, the interference imposed by the simultaneous transmissions naturally affects the performance of the PHY layer, in terms of successfully separating the desired signals from the rest. Additionally, as a result of transmission schedules, high packet delays and/or low bandwidth can occur, forcing the routing layer to change its routing decisions. It is clearly understood that PHY, MAC and network layers depend on each other in many ways, and that a joint design offers a way of tuning and improving their resulting performance.

Another cross-layer design approach is being proposed in Chen et al. (2006), where the authors consider the problem of congestion control and resource allocation through routing and scheduling over a multi-hop wireless ad hoc network. They have developed an extended dual algorithm to handle networks with time-varying channels and adaptive multi-rate devices. In more detail, the algorithm motivates a joint design where the source node adjusts its sending rate according to the congestion price generated locally.

The Eyes Source Routing (ESR) (van Hoesel *et al.*, 2004) is an example which illustrates how the topology information already provided by the MAC protocol can be used in order to efficiently manage the topology changes due to mobility, node and communication failures, and power duty cycling. For instance, ESR is able to detect link failures (e.g., nodes move out of each other's radio range) efficiently and fast, by accessing the knowledge already detected by the MAC layer.

The Label Routing Protocol (LRP) (Wang & Wu, 2006) is another ad hoc network routing protocol based on the idea of label concept[5], combined with the MAC layer in a cross-layer design's fashion. LRP is designed to be a more traffic-efficient and power-efficient source routing protocol. It is a virtual connection-oriented protocol that is able to setup, configure, and maintain a path between two or more end-points. In more detail, the path from a source to a destination node works as a tunnel identified by multiple labels and located between DLL and network layer. Using cross-layering, the label identifiers are used to identify a path or connection by considering the information gathered by the MAC layer. Additionally, the MAC layer itself is optimized in LRP for shorter delays, power saving and higher efficiency.

X-Layer (Beach *et al.*, 2008) is an implementation of a cross-layer network stack mainly designed for WSNs, under the assumption that different layers will interact and share resources directly with each other. X-Layer optimizes its capabilities both in terms of radio power and MAC layer transmission rates, in order to use less power and fewer bandwidth resources. The network layer on the other hand, takes advantage of the MAC layer's support of beacon signals and auto-acknowledgements, achieving better performance. The experiments conducted using X-Layer have shown that by using this network stack, better balance of network throughput, energy usage and latency is being achieved.

Cross-layer design can also be applied between higher layers. In their paper, Gharavi and Ban (2004) illustrate how this is done by proposing a feedback control mechanism that allows the application layer to adapt itself to a dynamically changing topology. They suggest a mechanism that offers dynamic adjustment

[5]The original idea of the label concept is given in Rosen et al. (2001) where multi-protocol label switching (MPLS) is described.

packet control for video communications in MANETs. The feedback mechanism can dynamically control the source bit rate in accordance with the characteristics of the multi-hop channel. Moreover, they prove that having knowledge about a link failure helps the application layer to control its packet transmission strategy accordingly.

2.8 Conclusion

This chapter was the first part of a review of the related literature, as it was dedicated to wireless ad hoc routing. A discussion about MANETs and WSN was presented in connection to the routing problem and those factors which affect it negatively. Physical restrictions caused by low-level wireless channel and signal propagation phenomena (path loss, shadowing, fading and signal interference), as well as problems that are related to the lower layers of the network stack (transport, data link and physical) were discussed, offering an in-depth understanding of the wireless telecommunication and its limitations over the wired counterpart.

Particular emphasis was also given to the two major aspects of MANETs and WSN, namely the node mobility and energy constraints. Allowing nodes to move freely around the topology not only causes frequent link breaks between nodes, but also changes the topology density. The latter affects the routing protocols in terms of path discovery and maintenance; the higher the density the more control overhead is required, and less throughput is achieved. Energy constraints also play an important role in routing, as nodes consume energy while receiving, transmitting and processing packets.

Section 2.5, 2.6, and 2.7 were used to discuss some of the most important routing techniques which are found in the literature and are related to the work presented in this thesis. Some of the most famous implementations were discussed, in order to prepare the ground for the next chapters, where the new proposed routing design will be presented. To summarise, the following classifications have been made:

- Internet-inspired protocols: reactive, proactive and hybrid based on their approach and internal route discovery and utilization mechanisms.
- Single-path and multi-path routing, based on the number of routes each discovery process is allowed to return.

- Hop by hop to source routing.
- Nature-inspired protocols: ant-inspired (ACO) and bee-inspired (BCO) routing protocols.
- Alternative approaches: location-aware routing protocols, energy-aware routing protocols, and more general examples of using cross-layering in network protocol design.

The proposed work is nature-inspired and, in particular, it is inspired by the collaborative behaviours found in honeybee colonies. The work is classified as reactive and uses the cross-layer approach in order to achieve its goals. What follows, is the second part of the literature review. The next chapter, chapter 3, is dedicated to the necessary background knowledge regarding the world of the real honeybees.

Chapter 3

The world of honeybees

"The bee's life is like a magic well: the more you draw from it, the more it fills with water"

– Prof. Karl von Frisch (1886-1982)

3.1 Introduction

In chapter 2, a review of the literature regarding wireless network communication and routing was given. Nevertheless, due to the biologically-inspired nature of the work presented in this thesis, attention has to be paid to the real honeybees and their behaviour. Material from the two literature review chapters presented in this thesis are also published in (Giagkos & Wilson, 2009), where the project was first introduced to the scientific community.

One of the pioneers in studying the collective behaviour of honeybees (apis mellifera) was the Austrian ethologist Professor Karl von Frisch[1]. In his book (von Frisch, 1967), von Frisch gave a full and in-depth description of foraging cycles, in terms of decoding the insect's dance language as a medium of communication and orientation. A complete study of honeybees has also been published more recently by Seeley (1995), who, based on experiments, answered significantly important questions regarding energy-efficient foraging. Studying the wisdom of the hive and the insect's simple yet collective behaviour is an active research today, such

[1]Nobel Prize in Physiology or Medicine in 1973, for his achievements in comparative behaviour physiology and pioneering work in communication between insects.

as Olague et al. (2006) in the domain of searching algorithms, and Randles et al. (2009) in load balancing[2].

In this chapter, the honeybee colony structure and collective behaviours in terms of foraging, are explained. Both natural scouting and foraging behaviours are covered, highlighting the concepts which have been the source of inspiration to certain ideas and features, in respect to the research being undertaken throughout the course of this work. Finally, a summary of the organizational principles found in the world of honeybees is included, along with a discussion of how they are mapped to computer networks and their contribution to solving the routing problem.

3.2 The honeybee colony

In this section, the most important characteristics of a honeybee colony are explained. This includes the different types of honeybees and roles a worker honeybee plays within a colony, as well as a description of the physical parts that constitute a hive.

3.2.1 Insect population

At all times of the year a honeybee colony contains a fertile female (the queen), and numerous infertile females (the workers). It also contains developing brood for most of the year, and male or drone honeybees for most of the spring and summer. A colony with an average strength has 30,000 workers and about 1,000 drones. Queen honeybees are specialized for egg-laying, whereas all other worker honeybees are responsible for performing the remaining tasks in the colony, apart from the drones whose purpose is to fertilize their receptive queen[3].

Depending on their age, they work either within the nest only, or act as scouts and foragers. As the honeybee size is related to its age, the tasks assigned at each stage of their life are different and become gradually difficult. They start by being responsible for cleaning the cells and feeding the larvae (nursing), to comb

[2]However, these examples are out of the scope of this thesis.

[3]Drones are defenceless due to their lack of stringer, which is primarily an egg-laying organ found in worker honeybees. However, they are still able to raise the alarm in case of an emergency, as well as being able to frighten the disturber by swinging their tail and wings. They are also responsible for controlling the internal temperature of the combs, producing and changing the air flow with their wings. The latter behaviour is also shared with the worker honeybees.

building and receiving/packing incoming pollen from the fellow foragers, to finally become part of the foraging strength of their hive.

3.2.2 The structure of the hive

A hive consists of several parts which are used for different reasons. Namely, the combs, the dance floor and the entrance. The combs inside a hive hang vertically and each is made of two layers of horizontal cells, with openings on opposite sides of the comb. These perfectly hexagonal-shaped cells made of wax, serve both as containers for stored food and as cradles for the developing immature honeybees. Additionally, the hive consists of cells dedicated to rearing workers (worker cells) and drones (drone cells). A typical hive in nature consists of approximately 100,000 cells.

Both natural and artificial hives have a small opening, usually facing down, called the entrance. The inside as well as the outside of the entrance is guarded by young workers, often named soldier or guard honeybees in the literature (Breed *et al.*, 1990). Honeybees that belong to the same colony share the same odour. Thus intruders can be easily detected and expelled. This screening process ensures that the colony stays safe at all times, from all strange creatures including honeybees of different colonies.

During both scouting and foraging operations, honeybees that return to the hive are expected to perform what is called a waggle dance in order to attract the attention of fellow workers and recruit them towards the new source of food. The process of dancing is explained in section 3.3.3. The place where dancing takes place is a particular region in the comb called the dance floor. The dance floor is generally close to the entrance. However its exact location is not precisely defined, as it depends on physical conditions such as the weather and the activity within the hive. For instance, the dance floor may be found further inside when it is cold or the hive is lightly populated, and closer to the entrance when the temperature is higher or there is lots of activity within the combs. Additional experiments in Free (1977) have shown that its position is influenced also by the distance of the food source from the hive. Finally, the dance floor is also used by unemployed honeybees to just wait, wandering around until potential recruitment takes place.

3.3 The process of foraging

The foraging process is what allows the members of a hive to collect its commodities, nectar, pollen and water. As they are all equally important to the colony's future, honeybees work ceaselessly in order to increase the overall production of energy within the hive (von Frisch, 1967; Seeley, 1995). In this section, honeybee foraging is explained. Emphasis is given to the ability of honeybees to effectively understand the quality of findings and based on that, to decide whether such information should be shared amongst the colony members. Sharing is achieved through the unique way of communication found in honeybees, the honeybee dance.

3.3.1 Towards a promising source of food

When the internal energy of the hive drops below the hive's threshold, thus the future of the population is negatively affected, some workers leave the hive to scout the environment in order to discover new sources of food. There are no specific scouting honeybees. Any one of the workers may set out on her way. Once a scout finds something interesting, she spends some time on the flower collecting the useful material. Depending on the type of the source found, she can either collect liquid (nectar or water) in her honey stomach, or construct a tiny ball of pollen to carry back with her legs, or both. Upon their return to the hive, those who bear pollen enter the hive directly and deposit it in the most satisfactory cell. Fellow younger workers do the rest of the processing of this food. A liquid bearer visits a worker where the latter uses her proboscis to quickly take the fluid from the others honey stomach.

Next is the advertising of the finding. The honeybee goes straight to the dance floor where she immediately hunts out recipients for her burden (i.e., pollen, nectar or sugar). The recruitment of fellow honeybees is done by performing a special dance, the honeybee dance, and is explained in section 3.3.3. During the recruitment, some previously unemployed honeybees get triggered by the interesting finding, and thus become foragers following the information taken from the initial scout. In turn, they stop wandering around and start working by leaving the entrance and flying towards the source of food. The number of initial recruits depends on the demand of energy in the hive, and the number of unemployed foragers. However, the recruitment strategy changes when more than one honeybee dance take place on the dance floor, which is the case most of the time. This is mainly due to the limited number of honeybees in the hive. The number of new recruits per honeybee dance tends to be fixed, independent of how many honeybees are waiting (Seeley & Towne, 1992; Gould, 1975; Gould *et al.*, 1970). Ethologists classify honeybee colonies as signaller-limited systems, where the recruitment strength is determined by the number of signallers instead of the number of potential recruits available (Dornhaus *et al.*, 2006). This is one of the main differences between honeybees and other insect colonies, in terms of communication. In ants for instance, a mass-recruitment system, the pheromone trail of a single ant can recruit a large number of workers as long as its odour persists (Vittori *et al.*, 2006).

The life of a forager is restless. When she returns to the hive with a full honey stomach, she expresses a similar behaviour to the initial scout. More specifically, the burden is deposited in the combs and then she goes straight to the dance floor. There, she can either decide to rest near her fellow foragers or perform a new honeybee dance to trigger others. That implies that helpers are not automatically recruited. Rather, their interest in following a specific finding depends on the future honeybee dances which, in turn, depends on the incoming experience. The richer the incoming experience, the more lively the honeybee dances, the more recruits triggered. Both von Frisch and Seeley agree that is the profitability of the food source and the foraging process that determines the number of honeybee dances (von Frisch, 1967). In more detail, the honeybee is able to measure and understand the quality of the foraging, including the quality of the flower, and use the information she gathers in order to decide whether it is worthy recruiting more helpers or not. In order to achieve that, she considers a number of quality factors, initially discovered and decoded by von Frisch (1967).

3.3.2 Quality factors considered by honeybees

As will be fully explained later in section 3.3.3, the pattern of the honeybee's dance is determined fundamentally by the distance of and the direction to the source of food. But whether dancing occurs at all, and how vivaciously and persistently it will be performed, depends on many factors that significantly regulate the relation between supply and demand in the hive. The factors that determine the release and duration of the honeybee dances are listed below.

The sweetness of sugar. An interesting source of food has to have a certain concentration of sugar. Honeybees are found to dance vigorously and attract more recruits when the sweetness is high. The acceptance threshold varies between a wide range of sucrose solution and does not depend on the quantity being carried. Furthermore, experiments have shown that during dancing, a sample of the collected nectar is released from the honey stomach, making it more interesting to the other honeybees, that not only decode the dancing, but they are also able to receive extra information by being allowed to taste the quality of the food. In addition, the tempo and frequency of moving the wings also changes with an increased concentration of sugar solution.

The purity of taste. Coupled with the sweetness described before, it has been found that the number of dances decreases as the sugar solution is replaced with ones slightly contaminated with salt. This is an indication that priority is given to the purest source of food, as the eagerness to dancing is gradually inhibited.

The quantity and ease of obtaining. Although the exact reasons behind these phenomena remain unexplained, it is discovered that even the sweetest sources release no dances, if the productivity of the foraging does not pace with the energy consumption within the hive. The time spent during foraging in connection to its productivity is also important. Saving energy is an important aspect of the colony and honeybees generally discard sources of food when the energy spent during the foraging exceeds the energy gained by the collected pollen or nectar. Also, any obstruction affects the quality of the foraging in a negative way, such as physical obstacles and dangers. In these cases, the number of honeybee dances are found to diminish.

Consistency and viscosity. Time does not always play a negative role. On account of its high viscosity, a source of food with concentrated sugar solution is physically sucked up more slowly and with more difficulty than a thin solution. Consequently, honeybees spend more time on the flower. Experiments have shown that in spite the time expenditure, the enthusiasm as well as the duration of the honeybee dance in such case are increased.

Weight of the burden. Another factor that increases the enthusiasm of honeybee dancers, is the weight of burden. Although the added weight of the load is accompanied by an increased flight time, the eagerness to dance is increased. Honeybees that return to their hives fully loaded are also fully satisfied by the finding and its quality. Like the viscosity described above, under natural conditions, increased weight is directly connected with a greater sugar content in the source of food.

Distance. The nearness of the source of food to the hive plays a very important role in the decision made by the honeybees. It is discovered that under identical conditions, the expenditure of energy and time over longer flights is proportional to the decrease in profitability. Obviously, flowers at a short distance from the hive allow more frequent visits by the foragers, and thus, are more productive than the same quality flowers at a long distance. The impact the distance has on the dances is not only seen for the duration of the honeybee dances, but also in how often they are performed. Taking into consideration that each dance is able to recruit a somewhat fixed number of unemployed foragers, it is obvious that collecting food from short distances can rapidly affect the distribution of foragers.

Type of the flower. In terms of this particular quality factor, it is found that the type of flower also plays an important role in successful foraging and number of dances. There are two reasons for this. Firstly, the strength of the fragrance which, although has nothing to do with the profitability of the food, it is considered to be an excellent tool in both enhancing the dances (as explained in 3.3.3), and aiding the recruits to the correct direction. Secondly, the shape of the flower and its potential in allowing the honeybees to be fed with no difficulty. Experiments have shown that honeybees are willing to advertise flowers with greater enthusiasm, when they physically allow easy access to the pollen and nectar.

Improvement over time. Another important quality factor that has been decoded is the improvement of the quality over time. Therefore, aside from the fact that a definitely better source of food (in term of sugar solution) releases more dances than one which is less sweet, a relative improvement over time has also a positive effect. It is discovered that under otherwise identical conditions, if honeybees are fed from a sugar solution whose quality is suddenly increased, the percentage of dancing foragers becomes greater. Similarly, when the quality of the sugar solution is decreased after contaminating it with other material, such as salt, less foragers are willing to perform their dances when they return to the hive.

Day conditions. Apart from several other characteristics of the source of food, such as the aroma, the taste, the consistency and the shape, the time of the day as well as the weather conditions can affect the performance of the foraging, and thus, the number of dances being performed in the hive. It is found that the number of flights from the hive to the surroundings usually drops temporarily around noon time. Moreover, the weather also affects their willingness to dance. Nature has given to them the gift of being able to detect impending rains and other severe conditions. Long before bad weather occurs, honeybees decrease any ongoing foraging activity by not dancing. Instead, they all gather inside their protective hive, until the weather improves.

3.3.3 Recruitement through dancing

Much work has been done since the first interest in honeybees to discover how they communicate with each other inside the hive, and how they coordinate their activities. Within the darkness or semi-darkness of the hive, visual systems can be of little or no importance. Hence, the known senses by which honeybees can recognize and signal to each other are auditory, tactile and chemical by the use of pheromones.

The main way of communication is achieved through the honeybee's special dance, which takes place in the dance floor of the hive (see section 3.2.2). The principle mechanism of the recruitment communication is the waggle dance, a unique behaviour in which a scout or a forager recruits fellow honeybees, by sharing information about a particular source of food. Seeley describes the honeybee dance as a miniaturized re-enactment of her recent journey to a source of food (Seeley, 1995).

Dancing patterns depend only on the distance and the direction of the particular source of food to the hive. In fact, depending on the distance, the dance can take two forms. There is the round dance for short distances (less than 100 metres from the hive), and waggle dance for longer distances[4]. During a round

[4]Ethologists have reported other types of honeybee dances that do not actually share infor-

dance, the performer attracts the attention of those unemployed honeybees that happen to be waiting next to her, by moving in a small circle of such tiny diameter that for the most part only a single cell lies within it. Just before the circle is complete, she suddenly reverses direction and moves again to her original course, and so on, as shown in figure 3.1(a). Interested honeybees are then touching her body, familiarizing in that way with any pollen residues which might have been stuck on her. For the same reason, if the performer was carrying nectar or water she releases some of it from her mouth. Hence, future foragers are aware of what they are about to look for, even if they do not know where exactly to find it (von Frisch, 1967).

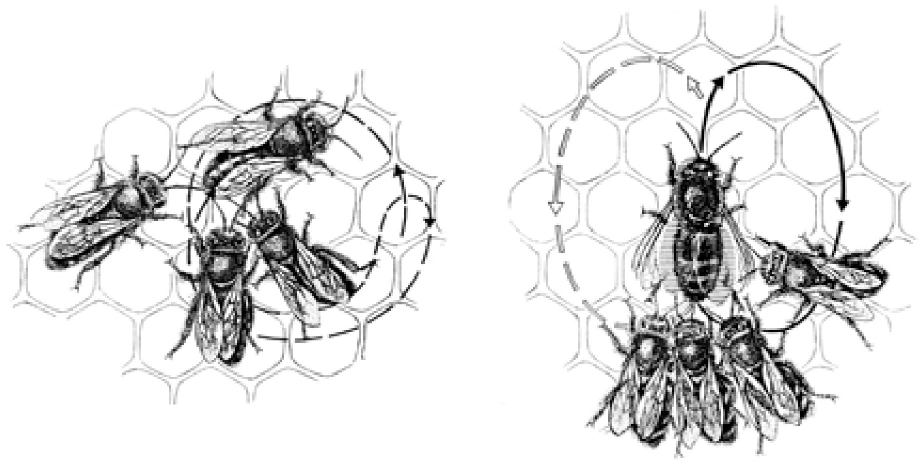

(a) The round dance. (b) The waggle dance.

Figure 3.1: A honeybee recruiting unemployed foragers by performing the round dance, left (a). Information about the direction is given though the waggle dance, right (b). Both images are taken from Prof. Karl von Frisch, "The Dance Language and Orientation of Bees", Oxford University Press, 1965.

The pattern is more complicated when the distance between the source of food and the hive is more than 100 metres. In such cases, the honeybees need to know the direction in which they should fly in order to find their target. The orientation of honeybees for long distances is achieved by the waggle dance, a dance which allows more information to be shared. In a typical waggle dance, the

mation about the foraging. For instance, the "grooming dance" is performed by honeybees who want to clean certain parts of their bodies that cannot be reached by their grooming devices, the "jerking dance" where two or more honeybees make contact to brush themselves with their antennae, and other forms of dancing which happen only during swarming (von Frisch, 1967).

scout or forager runs straight ahead for a short distance, where she then reverses and returns back to the starting point by completing a semicircle to either left or right. Once this movement is complete, she runs again straight ahead the short distance, reverses and returns back to the starting point completing another semicircle, but in the opposite direction. The exact pattern reminds of the Greek letter Θ turned by $90°$, and can be seen in figure 3.1(b). The performer gives particular emphasis to the straight part of the run, during which she vigorously dangles her whole body sidewise, making greatest moves at the tip of her abdomen (tail) and least at the head. The tail-wagging dance is coupled with wing shaking and buzzing, which both indicate the enthusiasm of the dancing. Like in round dance, the pattern may pause to allow fellow honeybees taste and touch samples of the collected material. The pattern of the waggle dance is more complicated than the round dance, because it includes the element of direction. In fact, the direction and duration of each waggle dance is closely correlated with the direction of and the distance from the source of food being advertised by the performing honeybee. Although the orientation of honeybees is out of the scope of this thesis, it is worth briefly explaining this magnificent phenomenon with an example. Keeping in mind the structure of the hive, where all combs are vertically built, a flower that is located directly in line with the sun is represented by waggle dance in an upward direction. Any angle to the right or left of the sun is coded by a corresponding angle to the right or left of the upward direction of the waggle dance. In terms of duration, the further the target, the longer the waggle dance.

To conclude, it is clear that honeybees have developed an extremely sophisticated yet simple mechanism to communicate with each other and share useful foraging information. This includes the direction to the source of food and the distance from it, the type of flower, its taste and fragrance, as well as an impression of how profitable this particular source of food has been found to be, by the visiting scout or forager. The more interesting and profitable the flower, the more enthusiastic the honeybee and her dance. In fact, dancing time becomes longer, with more noisy buzzing and higher tempo. On the contrary, a deterioration in the profitability as understood by the honeybees, leads to dull dances, or even no dance at all.

3.4 From natural to artificial colonies

All previous sections in this chapter have been dedicated to describe some important principles of the honeybee colony, as well as the collective foraging behaviours that are taking place within a hive. Due to the biology-inspired nature of this research work, it is very important to comprehend these behaviours, distinguish their characteristics and finally understand how they match to computer networks, and in particular, network routing. Consequently, this section outlines those particularly inspirational behaviours and characteristics and provides the medium that connects the two disciplines together. In addition, table 3.1 summarizes the mapping of the terminology and the concepts, from honeybee hives to networks, which is used throughout this work in order to enhance the understanding of the connection between the two worlds.

3.4.1 Agent-based collective and adaptive behaviour

Honeybee colonies, as well as other insects groups such as ants and termites, are naturally designed to be agent-based systems. Their strength is mainly driven from the size of their population, as well as the simplicity of the rules that all members are bound to obey. This allows nature to solve well-known optimization problems that are difficult or impossible to solve by an individual agent or a complex monolithic system. The allocation of resources under a constantly changing environment is one of them.

What is more, in a honeybee colony each insect can undertake several different duties throughout its life. This is a powerful adaptation mechanism which eliminates the need for developing special honeybee types, whereas at the same time each existing one is able to complete an allocated task very efficiently. For example, a honeybee worker which spends its day within the hive collecting the pollen being carried by older foragers, can switch role and become a soldier in order to defend the hive if an intruder decides to pass through the entrance. Similarly, a food processing worker might take the role of a nectar receiver, if the rate of collecting nectar is higher than processing it.

Honeybee concepts	**Network concepts**
the honeybee hive	the source node of a communication session
the source of food (e.g., a flower)	the destination node of a communication session
the path (or flying distance) to be traversed from the hive to a flower	the intermediate nodes in a multi-hop manner that form a path
the honeybee scout	the route request control packet e.g., the scout packet in BeeIP (explained in section 4.4.1) or the RREQ in AODV
the honeybee forager	the control packet whose payload is a piggybacked data packet e.g., the forager packet in BeeIP (explained in section 4.4.1)
the type of the missing commodity as it is monitored and signalled by the honeybees i.e., nectar, pollen or water	the desired behaviour for selecting the optimal path based on the type of traffic for a communication session (e.g., the fastest path or the path with the least energy consumption)
the foraging burden in the honey stomach or being carried with the honeybee' legs	the data payload (TCP segment) to be piggybacked to a forager packet
the level of satisfaction (the quality) of a food source and the foraging activity towards it	the quality of a path as calculated by the scout and the forager packets
the honeybee dance	the mechanism of making changes to the representation of how many data packets are allowed to use the particular path for future transmissions i.e., the foraging capacity in BeeIP (explained in section 4.2)
the honeybee recruitment	the allocation of foragers to appropriate paths for future transmissions (according to the foraging capacity of each path)
the entrance: the physical hole, usually at the bottom of the hive, which serves as both the entrance and the exit of the hive	the logical part of the source node which provides the interface to both MAC and transport layers
the dance floor: the physical place where the honeybee dance is performed	a logical part of the source node where the mechanism of making changes to the foraging capacity and the allocation of foragers are taking place
the combs: the physical places where honeybees rest and store their commodities	logical parts of the source node where control packets, scouts and foragers, are stored e.g., the local queue and the repository areas (see section 4.5.2 for details)

Table 3.1: Mapping honeybee concepts to networks.

3.4.2 Decentralized, distributed foraging and self-healing

As explained above, multi-agent systems eliminate the problem's complexity by distributing it over the system, instead of depending on an individual agent to act like the only solver. This is the foundation of any decentralized approach, making it independent of any strict hierarchy in decision making. Honeybees are able to coordinate their moves and to plan their foraging without the need of central guidance and authority. What allows thousands of honeybees to work in harmony is their loyalty to following simple rules, common to the whole population.

The distribution of work, equal rights and responsibility amongst the hive members, plays an important role to the overall future of the colony. The foraging activity and thus the energy level within the hive is not affected by the loss of a honeybee. Rather, the loss will be absorbed by the rest of the population, as the post will be automatically replaced. The self-healing characteristic allows the colony to adapt to any dynamic environment, independently of the frequency by which changes occur.

3.4.3 Adaptive foraging

Foraging, as well as other activities in the daily agenda of honeybees, e.g., nursing, food processing and drone reproduction (Seeley & Mikheyev, 2003), are influenced by internal and external factors. Honeybees constantly monitor the remaining energy in connection to the productivity within the hive, and make adjustments to their activities. A forager who discovered a good flower in terms of quality dances enthusiastically, providing positive feedback to her fellows. Similarly, she might decide to not dance at all, or provide negative feedback through her dancing by performing in a dull mood. Her decision is based on the measurements received from the environment and her quality dance thresholds.

Since the pioneering work of von Frisch, it is realized that honeybees are capable of changing the dance threshold according to several foraging conditions. Von Frisch was the first to notice that when the discovered source of food was of high quality (e.g., increased concentration of sugar), the forager was dancing vigorously in an attempt to attract the attention of the surrounding unemployed foragers, and possibly allocate them first before other dancers that might perform nearby. In addition, Seeley's extensive experiments (Seeley & Towne, 1992) have made it clear that between the two hypothesis: a) a forager compares her current finding with previously experienced findings, and b) the receiver honeybee within the hive preferentially accept those material with the highest sugar concentration, it is the first that is validated. This concludes that it is the foragers which assess the quality of a finding, and adjust their dancing threshold accordingly.

Another example of reinforcing the foraging activity based on up-to-date feedback from external factors, is when foragers switch from one commodity collection to another, e.g., pollen to water. It has been observed that switching depends on the time a forager spends within the hive, waiting for a receiver honeybee to accept her burden. When a forager spends very little time waiting for a nectar receiver, this is a signal that the need for this particular commodity is very high, hence foraging for it is more important to the colony.

Finally, another example of reinforcing and adaptability is presented in Seeley and Mikheyev (2003), where the authors show that a colony might trim its allocation of energy to drone production and even maintenance, if it is experiencing difficulty building the energy reserve it needs to survive a cold winter.

3.4.4 Grouping according to odour

The internal organization of the hive leads to another remarkable phenomenon, which was also first observed by von Frisch. During his studies with honeybee foraging, he was surprised to see how fast each message was grasped by interested groups of unemployed foragers waiting in the dance floor. His further experiments show that the foraging population of honeybees is separated into several groups, each one being unique by the odour of the particular targeting source of food. Until then, the odour that was known to be shared amongst honeybees was the odour of their own colony. Being used as the family's signature, honeybees from different colonies are easily recognized and driven off the hive.

Furthermore, grouping is also done by taking into consideration the foraging needs. This is another reinforcing learning example being met within the hive. When the need for a specific commodity is higher than a threshold, waiting foragers might become members of a different group in order to enhance its foraging strength.

3.4.5 Linking quality factors to network conditions

All the behaviours which are covered above, have been a great source of inspiration for the work presented in this thesis. In particular, they have been studied and mapped to the design model of the proposed routing protocol, which is fully covered in chapter 4. The proposed approach emphasizes the monitoring and evaluation of the performance of each artificial finding, i.e., a path between a source and a destination node in a mobile ad hoc network, in a similar way as real honeybees do in nature. That is, assessing the quality of the path by comparing new quality measurements with their past experience in using that path.

Section 3.3.2 lists a number of quality factors being considered by natural foragers in the real world. Each one has a different potential for use within a wireless network, a mapping of which is covered in this section. To start with, both the sweetness and purity of the source of food are values related to the amount of energy that will be generated in the hive, if material collected by the particular finding are used. In a mobile ad hoc network, finding a path between two nodes that fulfils the needs of the data transmission is equally important. Here, the need to produce energy within the natural hive can be considered as the need of transmitting data over an optimal and effective path. Common to both worlds, optimality depends on the requirements, either set by the colony (pollen, nectar, water) or by the source node in the computer network (transmit data with less delay, causing less traffic congestion, etc.).

Thinking of the ease of obtaining and weight of burden, these are both factors that natural foragers consider in order to tune their recruitment dance. In the artificial world, paths of high delay, packet loss and frequent broken links are decreasing the ease of maintaining an efficient communication session between a source and a destination. In a similar way, a fully loaded honeybee considers her finding very successful, a fact which can be interpreted as an increased of transmission data rate in computer networks.

Improvement over time is not only a characteristic that can be mapped from nature to networks, it is also one of the fundamental ideas behind any adaptive routing protocol in the field of mobile ad hoc networks. Being able to respond to changes by adjusting routing, not only enhances the self-healing aspect of the protocols, but also aids their scalability to different network sizes. Real honeybees have solved these problems by constantly monitoring and evaluating their activities, and making changes according to their understanding.

As described in the previous chapter 2, traditional routing protocols provide routing solutions by considering the number of hops between a source and a destination, as their performance metric. The link between computer networks and nature at this point is pretty obvious. Honeybees consider distance to be one of the quality factors of a source of food. Depending on the conditions in each world, decreasing the distance between the two flight points often increases the amount of material (data in networks) being carried back and forth.

The type of the flower as a quality factor in real honeybees can also be mapped to networks, if the Quality of Service (QoS) (Zhu *et al.*, 2004) is being considered. For instance, sending real-time audio and video has more strict service demands than other types of transmissions. Being able to set the requirements and categorize routing solutions based on them, is a similar approach to what honeybees follow in nature by understanding that a particular type of flower is more productive than others. Coupled with what has been described in section 3.4.4 regarding adaptation and grouping, QoS-aware routing protocols can take advantage of such a nature-inspired approach and distinguish between traffic types and select accordingly.

Finally, the requirements can also be set by a network authority, such as the administrator, to follow a specific communication policy depending on the time of the day. This is very important for some companies, for example, as it allows them to manage the availability of certain services and distribute the load within their networks without causing inconvenience to the users. Mapped to real world, the time of day is the equivalent quality factor that honeybees consider in managing the foraging profit.

3.4.6 Swarming and colony reproduction

The honeybee colony reproduces by swarming. Normally, there is only one queen in each colony, but before reproduction additional queens are being reared. When one of the new queens is grown up enough to reach the pupal stage, the old one accompanied by a proportion of the hive's population, including workers and drones, usually leave and fly off in a well defined group called a swarm. The swarm might spend some time in temporary places such as branches of trees, until they all settle down to a satisfactory place and build their new hive. Most of the additional queens follow the same destiny but one, which stays behind in the colony, mates, and lays eggs. During this process, the members of the new colonies develop their own odour and create strong, new families. Thus, they are not allowed to return to their parent hive, for they will then appear as intruders and threats.

In the communication network environment, the principle of swarming, i.e., some proportion of the population fly off and never returns back to the hive, can be found in unreliable transport protocols such as UDP. In UDP, packets are being delivered without acknowledging their transmission success, in contradiction to TCP where each packet's reception is acknowledged. A first attempt to map such an idea in routing protocols has been done in BeeAdHoc by Wedde et al. (2005a), when UDP is chosen instead of TCP. This is better explained in section 2.6.3 of chapter 2.

3.5 Conclusion

This chapter concludes the literature review of this research work. In particular, it discusses the most important behaviours in terms of food collection and communication between members of a honeybee colony, within their hives. In the first part (sections 3.2 and 3.3), the types of honeybees, the hive structure and its most important internal parts, as well as the natural foraging process itself were described. Emphasis was given to the ability of honeybees to constantly measure and evaluate the quality of their findings, based on a number of quality factors and their previous knowledge.

Furthermore, in the second part of this chapter (section 3.4) a mapping between nature to networks was given. This includes a discussion of all those characteristics and inspirational behaviours of the insects, in terms of foraging and communication, which match aspects and required behaviours in wireless ad hoc network routing.

In the next chapter, the proposed bee-inspired design for routing is thoroughly presented. BeeIP (the name of the protocol), combines ideas from previous work, which found in the area of wireless ad hoc network routing, with the ideas from the organizational behaviours found in honeybee colonies.

Chapter 4

BeeIP: The bee-inspired protocol

4.1 Introduction

In this chapter BeeIP is described. BeeIP is a bee-inspired network layer routing protocol for mobile ad hoc networks. By following the simple yet collaborative principles of real honeybees, it is able to discover paths between sources and destinations and utilize them in order to allow real data transmission. Novelty is found in the way paths are constantly monitored and evaluated based on previous knowledge that the artificial honeybees possess. The proposed features allow the protocol to utilize a number of important low-level parameters in order to represent the goodness of the paths. Furthermore, a novel way of modelling the honeybee dance is presented. During the dance, prior knowledge is considered in order to detect any improvement or deterioration over time. This allows the protocol to build a list of promising paths from which the optimal one is selected, making sure that packets are transmitted in a multi-path manner.

The rest of this chapter is organized as follows. First, a general overview of the protocol is given. This includes a high level description in words as well as a schematic representation of the most important states of the protocol's algorithm. Then, each component of the protocol is discussed in detail, providing all the required information to understand the inner mechanisms and behaviour.

4.2 General protocol description

BeeIP is a reactive routing protocol. As in other reactive approaches, it does not maintain any routing information regarding the topology without a route request being previously triggered. Discovering new paths is done by emulating the scouting behaviour of real honeybees. Real honeybee scouts are sent to explore the surroundings of the hive towards the appropriate flower, and are expected to advertise their finding upon their return to the hive. In BeeIP context, when a route is required at the source node, the source searches its routing table for any available paths to the destination. If none are found, a scout packet is generated and sent by broadcast to the rest of the network. At the same time, information about the new scouting is created, marked as unacknowledged, and scored internally. This ensures that as long as a scout has been already sent out, future scouting processes for the same destination will be avoided.

When a scout packet is received at a neighbouring node, information about the particular scouting as well as the scout's sender are created and stored internally. This allows the nodes to be introduced to each other, and also to be able to detect future loops as they become familiar with the particular scouting process. Then, one of the following happens; If the receiving node is not the destination node, the scout is broadcast further. Otherwise, the receiving node constructs a reply packet, called ack_scout, and sends it back to the source to confirm a successful path. Unicast is used instead of broadcast, because the ack_scout already knows which nodes to visit in order to return. That is, the reverse of the path the scout traversed in the first place.

During the journey back, ack_scouts constantly collect and deliver low-level data received from the lower layers, from one node to the other. All receiving nodes consider the scouting process successful, create routing information, and mark the path as acknowledged. They also update their knowledge about the neighbouring nodes using the data attached to the ack_scouts. This is a very important step, because routing is maintained through these updates as nodes are able to evaluate the performance of the links between them and their neighbours.

BeeIP is also a multi-path protocol, meaning that a destination node is allowed to generate more than one ack_scout, acknowledging the success of more than one unique path. This feature allows the protocol to have easy access to alternatives in case of path failures, or paths being dropped deliberately due to bad performance.

While a scouting process is in progress, the application layer of the source may require to send more packets to the same destination. This is a behaviour caused by transport layer protocols such as TCP, due to their own communication strategies. These packets are put in a local buffer queue and stay either until a path is discovered, so they can be transmitted, or until they age more than a constant number of seconds.

Data transmission is achieved by creating, loading, and sending forager packets. Each forager is responsible for carrying data payload received from the transport layer. Carrying is done by traversing one of the previously discovered paths towards the destination. The selection mechanism for the most appropriate path depends on the desired behaviour of the protocol, subject to its implementation (described in section 4.7). However, the default selection is based on the fastest path between the source and the destination.

Once a forager has piggybacked the data and is given a path to work with, it starts its journey. At each intermediate node, the next hop is easily retrieved from the local routing information, based on the direction of the forager. This approach is inspired by the source routing (Forouzan, 2005; Postel, 1981) met in other reactive protocols such as DSR (Johnson & Maltz, 1996). However, in BeeIP the forager packet does not need to carry the whole path within its header. Instead, each intermediate node already knows which paths it is a member of, and based on the direction of the forager, it can decide the next node towards the source of the destination.

When received by the destination node, the forager packet delivers the data and stays in a buffer queue until it is again assigned to carry data back to the source. This behaviour mimics the real honeybees which stay on the flower for some time collecting the required commodity, such as pollen or nectar. For scheduling purposes, when they are sent back to the source as ack_foragers, they are utilized in a first-in-first-out fashion. The next hop selection for an ack_forager is then done again by considering the forager's direction.

As seen in chapter 3, foraging in nature consists of several phases; a honeybee being recruited to follow a specific mission, flying to the target source of food, collecting the actual pollen, nectar or water, and carrying it back to the hive. When a honeybee completes a foraging cycle, thus she is back at the hive, she can either perform a honeybee dance to share up-to-date information about the foraging, or forget it and focus her attention to something else. The decision is based on the quality factors listed in chapter 3. BeeIP foragers are designed to follow similar principles. In particular, ack_foragers are able to judge the quality of the path and detect any possible improvement or deterioration over time. Monitoring a path's quality is both a complicated and network-dependent procedure, the outcome of which allows the source to adjust a path's foraging capacity. The foraging capacity is defined as the total number of foragers allowed to be recruited and use the path in the future. In more detail, every time a new artificial dance is released for a particular path, a small number of foragers is added (recruited) to the path's foraging capacity. This approach ensures that paths which release more positive dances will end up with higher foraging capacities and become more active in the long run, than others which release negative dances or no dances at all.

4.3 The internal states and transitions

The internal processes of the protocol and the steps that connect them are shown in figure 4.1. A node operating with BeeIP starts with the "initial" state, where internal variables are initialized. Being a reactive protocol, BeeIP does not start any route-related bootstrap process. Rather, it stays in "idle" state while it waits for an incoming packet, either from its application or the network. The state named "send to next layer" is the state from where a packet leaves the network layer.

In an event of a packet being received, the protocol acts according to the packet's type. If it is a non-BeeIP control packet (this includes the artificial foragers that carry real data), then it is handled by the "recv data pkt" state. A packet that is either marked for broadcast or its destination is the receiving node, is sent directly to the "send to next layer" state for immediate delivery. Otherwise, the receiving node must forward the packet in such a way that it will eventually find its destination.

There are three scenarios at this point. The receiving node checks whether there is already some routing information regarding the destination of the packet. If the information exists, the packet is then prepared for sending by following the states "allocate new forager", "update hive (update tables)", "piggyback data" and "send to next layer". If no routing information is available, the packet is pushed into the BeeIP queue, and a new scout packet is created and sent, by following the states "create/edit scout", "update hive (update tables)", "send to next

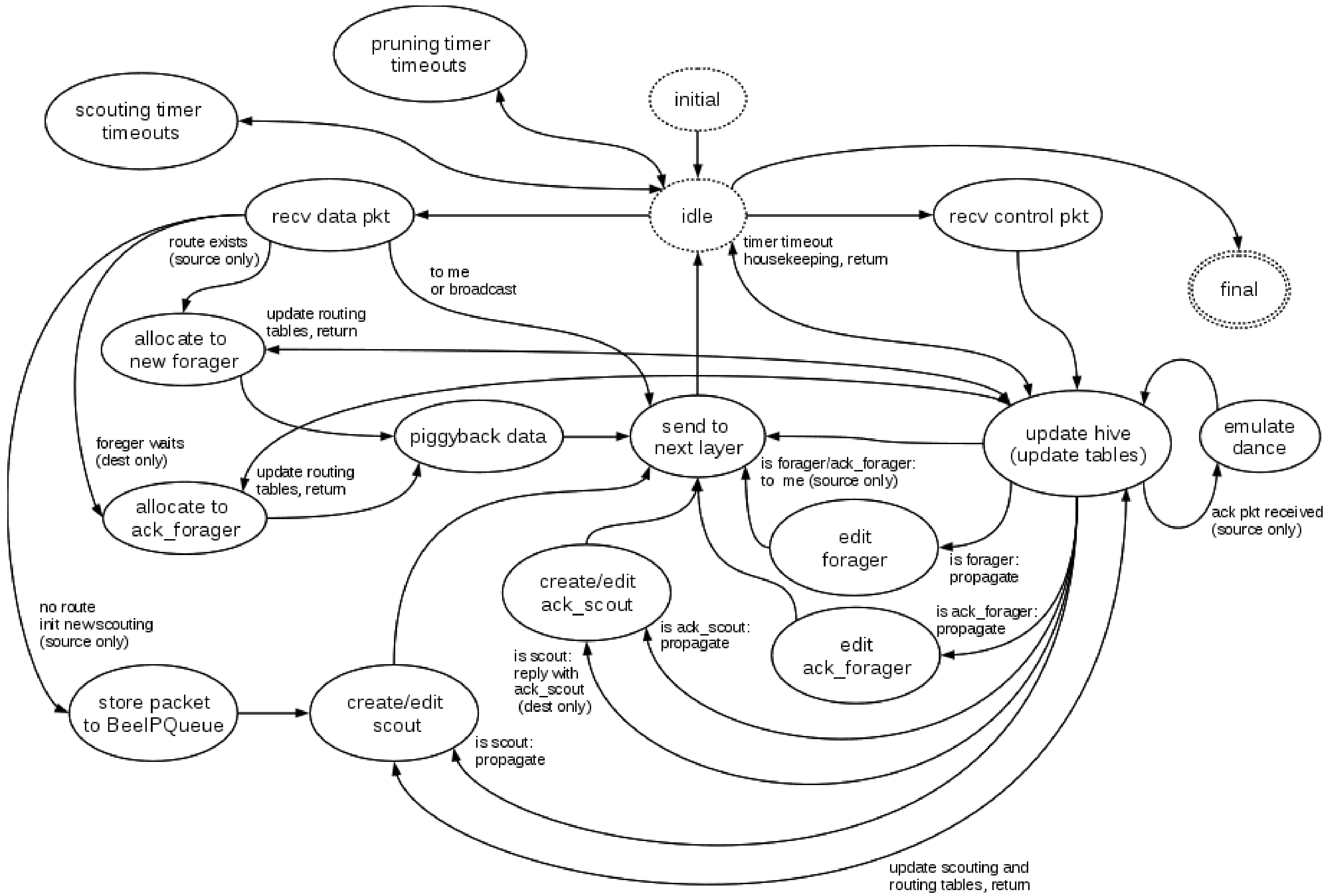

pruning timer timeouts
initial
scouting timer timeouts
recv data pkt
idle
recv control pkt
route exists (source only)
timer timeout housekeeping, return
to me or broadcast
final
update routing tables, return
allocate to new forager
foreger waits (dest only)
piggyback data
send to next layer
update hive (update tables)
emulate dance
update routing tables, return
is forager/ack_forager: to me (source only)
allocate to ack_forager
edit forager
ack pkt received (source only)
is forager: propagate
create/edit ack_scout
no route init newscouting (source only)
is ack_forager: propagate
is ack_scout: propagate
edit ack_forager
is scout: reply with ack_scout (dest only)
store packet to BeeIPQueue
create/edit scout
is scout: propagate
update scouting and routing tables, return

Figure 4.1: A diagram that summarizes all the processes and steps related to the network layer, within a BeeIP node.

layer". The third scenario describes the case where the receiving node is in fact the destination of a previous communication session. In this case, the node looks for an ack_forager which may be idling in it, after having successfully delivered some data of a previous transmission. An available ack_forager is immediately allocated to piggyback the data packet, and is sent to the next layer. The protocol moves to the "update hive (update tables)" state, in order to build the appropriate data structures for both scouting and routing tables (section 4.5.2). These behaviours are described in detail in the next sections, 4.5 and 4.7.

If the receiving packet is in fact a BeeIP control packet (this includes the artificial foragers, refer to section 4.4.1 for a full description), it is handled by the "recv control pkt" state. Initially, a transit to the state "update hive (update tables)" examines whether the received control packet contains useful information, in order to update its scouting or routing tables of the receiving node. Then an action is taken according to the type of the control packet.

BeeIP moves to a special state, named "emulate dance", only if the receiving node is the source of a communication session and the received control packet is either an ack_scout or ack_forager. This state is responsible for recalculating the foraging capacity of the particular route, mimicking the special dance of honey bees. More details about this process are given in section 4.6.4.

The possible scenarios when receiving a control packet are illustrated in figure 4.1, are as follows. A scout packet which has found its destination is replied by an ack_scout, after moving to "create/edit ack_scout" state, where a new ack_scout packet is created. A scout packet which needs to be propagated further is passed to the "create/edit scout" state, where it is prepared before "send to next layer" broadcasts it further. In a similar manner, ack_scout, forager and ack_forager packets are either accepted and sent to the upper layer of the receiving node, or propagated further via the "create/edit ack_scout", "edit forager", or "edit ack_forager" states.

4.4 Artificial hive

Any node operating with BeeIP that participates in a communication session can be seen as either an artificial hive, a member of a scouting/foraging path, or an artificial flower. Depending on the network traffic, nodes may play any of these three roles at any time. BeeIP's potential is derived from its minimalistic yet modular design, simple searching algorithm, and regular housekeeping using clock timers. In this section, the internal structure of the BeeIP node as well as the BeeIP packet types are presented in detail.

4.4.1 BeeIP control packets

There are four control packets that are used by the protocol; scouts, ack_scouts, foragers and ack_foragers. These packets not only allow path discovery and routing maintenance, but also carry real data across the wireless network. Their headers as well as their life span are presented below.

The scout

A scout is created and sent by the source node to discover new paths. Sending is done by broadcast and is controlled by the use of the time-to-live (TTL) value of the IP header. This is a very popular technique met in other reactive protocols such as AODV (Perkins *et al.*, 1999). A scout packet is initially sent with a small TTL value. If it is not received back in a certain period of time, then a new scout packet will be created and sent with a TTL value increased by a TTL step. This will continue up to a maximum number of retries, when the TTL will get the predefined network diameter value. The default values are presented in table A.1 of Appendix A. This strategy reduces the control overhead which is produced by the path discoveries. Moreover, while a scouting is in progress, any path request for the same destination at the source node is ignored. A typical scout's header contains the fields presented in table 4.1.

The packet type (0 for scout)
The creation time-stamp of the packet
The packet header size
The IP address of the source
The IP address of the destination
The list of IP addresses of nodes already visited
The number of nodes (hops) in the IP address list

Table 4.1: The scout packet header.

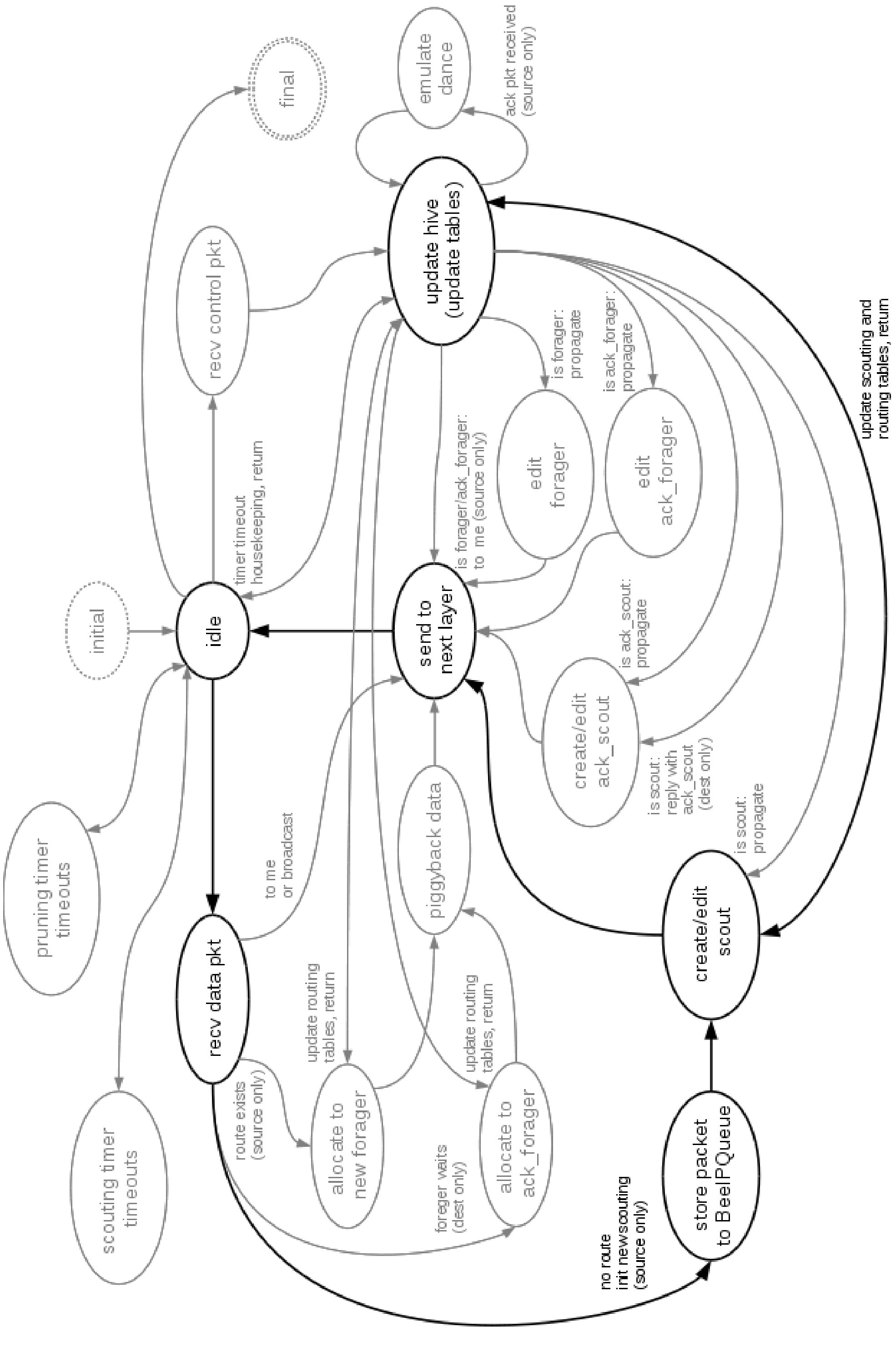


Figure 4.2: A diagram which highlights the process of scouting and the corresponding steps within a BeeIP node.

Figure 4.2 as well as flowchart 4.3 highlight the process of a new scouting[1]. When no route is available to a destination, a new scouting is created and the internal scouting and routing tables are updated. Then, the packet is sent to the next layer to be broadcast to the neighbouring nodes. All nodes that receive a new scout packet immediately store the scouting details found in its header. If it has been already received, based on its time-stamp value, then it is silently dropped in order to avoid loops. If a receiving node is in fact the destination node, then the scouting is marked as acknowledged and an ack_scout packet is created for the journey back. Otherwise, the scouting is marked as unacknowledged, and the scout packet is prepared to be propagated further. The latter involves adding the current node's IP address to the list of nodes that have been already visited, increasing its hop counter by 1, and finally updating the packet's size.

The size of the scout is larger after each propagation and is directly related to the density of the topology, thus, the number of hops between the source and the destination. This is a common pitfall to similar route request discoveries and source routing. To avoid packet sizes going beyond limits, the network diameter is usually set to a constant value such as 35 for AODV (Perkins *et al.*, 1999).

When the scout is received by its target destination, the destination immediately acknowledges the path's success and creates a routing table to store the information about the fresh discovery. It is also the destination's responsibility to assign the path a unique path identification number, which will be used to distinguish it from other future findings. The routing information and table at a destination node is properly described in section 4.4.2. Algorithm 1 summarises the receiving of a scout packet at a BeeIP node.

The ack_scout

An ack_scout is created by the destination as a reply to a previously accepted scout. Due to the multi-path nature of the protocol, the destination is allowed to answer to more than one scout that belong to the same source. In addition to the scout header's fields presented in table 4.1, a typical ack_scout's header also carries the fields listed in table 4.2.

The packet type field for an ack_scout is set to 1. The purpose of an ack_scout

[1]Notice that states and steps diagrams are the same for all nodes operating with BeeIP. Depending on their particular role (i.e., hive, path or flower) different parts of the algorithm are active.

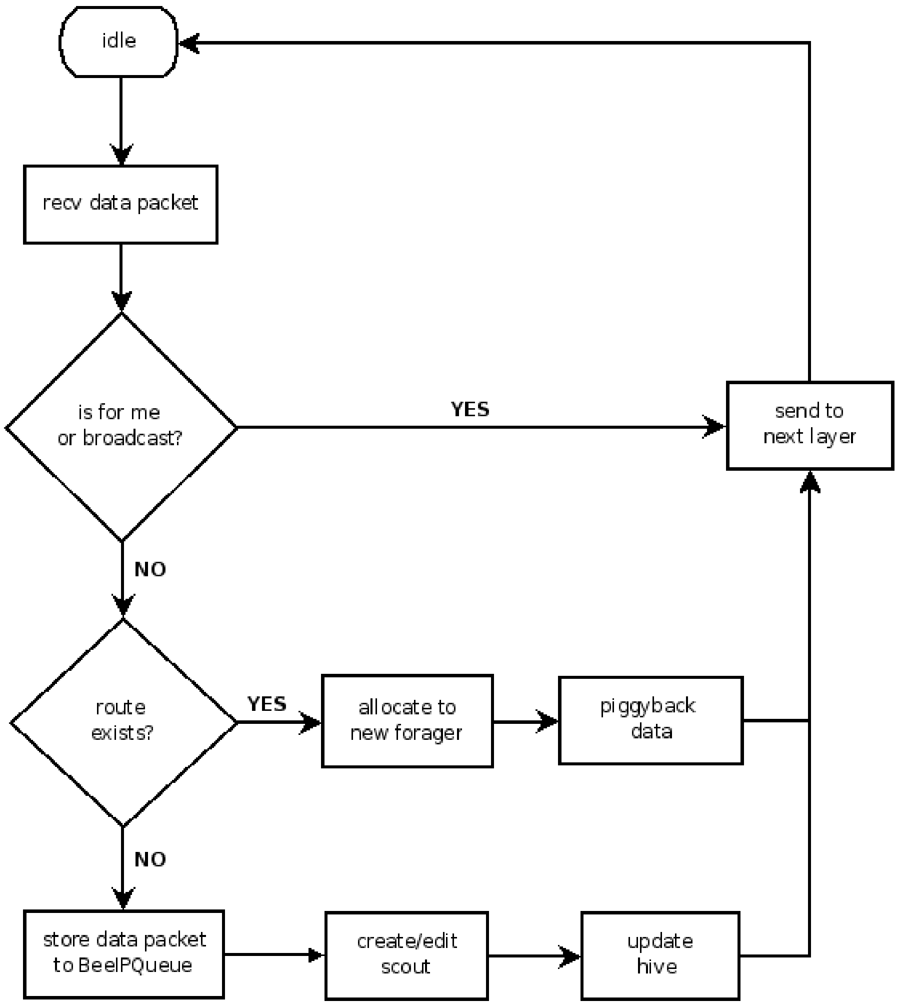


Figure 4.3: A process flowchart that presents the internal decisions and processes in terms of initializing a new scouting within a BeeIP source node. A data packet is received at the network layer and the routing protocol needs to forward it to the correct direction. If no route exists, then a new scout packet is created and sent to the network.

Algorithm 1 Receiving packets at a BeeIP node: Part 1 - The scouts

Input: A packet *pkt* received from either the upper or the lower layer

```
1:  for all received $pkt$ do
2:      if $pkt.Originator$ is me then
3:          DROP($pkt$)
4:          return
5:      end if
6:      if $pkt.Type$ is scout then
7:          $result \leftarrow$ RECVSCOUT($pkt$)            ▷ See Algorithm 7 in Appendix C
8:          if $result$ is loop then
9:              DROP($pkt$)
10:             return
11:         else if $result$ is reply with ack_scout then
12:             ADDROUTINGDATA($pkt$, FLOWER)
13:             SENDACKSCOUT($pkt$)
14:         else if $result$ is propagate scout then
15:             PROPAGATESOUT($pkt$)
16:         end if
17:     end if
```

during its life span is threefold. Firstly, it acknowledges the success of the path and advertises its unique path identification number to the nodes it visits. Secondly, it carries the appropriate information and triggers the mechanism to create routing information at each receiving node. Thirdly, it initializes the monitoring of each link within the path, collects information about their quality, and finally reports it back to the source. Although, the monitoring activity is described in detail in sections 4.6.1 and 4.6.3, at this stage it is important to understand the data structure used to carry the appropriate low-level information from one hop to the other. The sender state is itself a collection of values which are selected from each sender, and is delivered to the next receiver in the path. Any intermediate

The unique path identification number
The path selection metric value
The path quality value
The previous ack_scout sender's state

Table 4.2: Additional fields for the ack_scout packet header.

node as well as the source node which receives the ack_scout, stores these values to its current neighbouring knowledge, and uses them to calculate the quality of the intermediate transmitting link. The process is repeated until the ack_scout is finally received at the source node, where the overall quality of the path (chain of links) is reported. The path quality field of the header is used to hold updates of the overall quality, and its value is updated at every next hop.

The path selection metric field is what is used to keep updates of the metric value used for selecting the most appropriate path at the source node. Selecting the path is designed to be a separate, flexible and configurable feature (see section 4.7 for details). Considering BeeIP as an abstract design to implement honeybee inspired routing protocols, selecting which path is the most appropriate for a transmission at the source node depends on the required behaviour. In any case, the path selection metric field can be used to hold the appropriate metric's value.

In terms of propagation, source routing is applied to ack_scouts in order to reach the source. Finding the next hop on the path is an easy process, because the full value can be inherited from the scout unchanged. A successful single path discovery ends when an ack_scout is finally received at the source node. However, the scouting process itself may continue, as several paths might be available and other ack_scouts are expected to return to the source. New routing information is created for each new path discovery (initial and additional), and stored in both intermediate and source nodes. This is particularly important as the routing tables will be used to both maintain routing and provide routing solutions.

When the ack_scout is finally at the source node, another important activity takes place. The final update of the path quality field is calculated and its value is reported. After a series of calculations, which are fully described in section 4.6.3, the protocol is able to make adjustments to the foraging capacity of the path, i.e., the number of foragers that will be allowed to use it in the future. The calculations as well as the adjustments to the foraging capacity can be considered as the artificial counterpart of the real honeybees' dance. The receiving of an ack_scout is illustrated in Algorithm 2.

The forager

Foragers are the workers of the BeeIP routing protocol. They are specially crafted packets which provide the transport medium for real data. Every time the trans-

Algorithm 2 Receiving packets at a BeeIP node: Part 2 - The ack_scouts
Input: A packet *pkt* received from either the upper or the lower layer

```
18:     if $pkt.Type$ is ack_scout then
19:         $result \leftarrow$ RECVACKSCOUT($pkt$)        ▷ See Algorithm 8 of Appendix C
20:         if $result$ is propagate ack_scout then
21:             ADDROUTINGDATA($pkt$, PATH)
22:             PROPAGATEACKSCOUT($pkt$)
23:         else
24:             // $result$ is ack_scout at hive
25:             ADDROUTINGDATA($pkt$, HIVE)
26:             EMULATEDANCE($pkt$)                        ▷ See Algorithm 5
27:             // Check forager capacity of all available paths to $pkt.Dest$
28:             // and dequeue all destination-related data packets waiting
29:             // in BeeIPQueue. Send as many as possible:
30:             $bees \leftarrow$ OVERALLFORAGINGCAPACITYTO($pkt.Dest$)
31:             while $dataPkt \leftarrow$ next queuing data packet and $bees > 0$ do
32:                 $bees \leftarrow bees - 1$
33:                 // Encapsulate $dataPkt$ to new forager.
34:                 $ppkt \leftarrow$ ALLOCATETOFORAGER($dataPkt$)
35:                 FORWARDFORAGER($ppkt$)
36:             end while
37:             DROP($pkt$)
38:             return
39:         end if
40:     end if
```

port layer protocol delivers a segment with a destination address written on to the network layer, a forager packet is created to undertake the packet's transmission. A typical forager's header is shown in table 4.3.

To start its journey, a forager requires an identification number of a path which will lead to the destination. BeeIP is a multi-path routing protocol, which implies that there may be more than one path available between a source and a destination. Therefore, the most appropriate one is selected from a populated list of alternatives. If there is only one path available on the list, then its path identification number is automatically selected. Continually, the source node IP address is set to the current node's IP address, and the destination node IP address is set to the next hop, which is found by looking up the appropriate routing information from the routing table, using the given path identification number.

In the unfortunate situation when there is no path available, e.g., no paths have

The packet type (2 for forager)
The unique path identification number
The creation time-stamp of the packet
The packet header size
The IP address of the source node
The IP address of the destination node

Table 4.3: The forager packet header.

been discovered yet, foragers are pushed in a local buffer queue at the network layer. At that time, a new scouting process may be triggered if no older scouting is already in progress. The foragers can stay in the buffer queue for up to 4 seconds. Then they are automatically dropped.

One of the strengths of BeeIP routing is the fast packet forwarding mechanism. In particular, finding the next hop at each node is a matter of knowing the direction of the forager, and the identification number of the path that it is bound to traverse. The routing information for each path includes the IP addresses of the two adjacent neighbouring nodes, which can be used to forward either towards the source, or towards the destination. An example is presented in figure 4.4. According to node's B routing table, when a forager is received from node A with path identification number equal to 1024, the next hop towards the destination is C. Notice that B's rather limited knowledge of the rest of the topology is enough to allow packet forwarding, occupying as little memory as possible. Hence, all B needs to do in order to forward the forager, is to alter both the forager's source and destination IP addresses and send it off to the data link layer.

The ack_forager

Like a forager, an ack_forager is a real data carrier. However, its responsibilities also include those of an ack_scout. Namely, an ack_forager monitors the quality of each link within the path, collects the information and finally reports it back to the source. As in ack_scouts, reporting the quality of the path affects the path's foraging capacity. Ack_foragers are generated at the destination node by converting foragers who have been previously received and are waiting, until there is some data that need to be transmitted back to the source. In a typical TCP scenario,

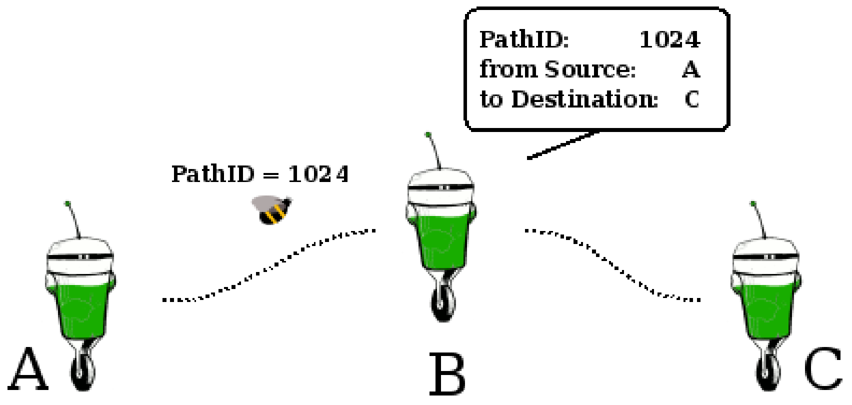


Figure 4.4: A forager packet forwarding at the intermediate node B. The dotted lines represent wireless links. The forager is bound to traverse the path with identification number equal to 1024. The next hop, C, is found based on the path identification number and the forager's direction by looking at the routing table.

TCP acknowledgements are regularly sent back and forth during a communication session. Therefore, the foragers waiting at the destination are not expected to stay for too long. They get converted to ack_foragers one by one, in a first-in-first-out manner. A typical ack_forager's header is similar to a forager's presented in table 4.3, with an addition of the fields presented in table 4.4.

The path selection metric value
The path quality value
The previous ack_scout sender's state

Table 4.4: Additional fields for the ack_forager packet header.

Acting as an ack_scout, an ack_forager is designed to carry low-level values from one hop to the next, update the value of the path quality field and perform the artificial honeybee dance at the source node. Also, it constantly updates the path selection metric value at every hop. The internal decisions for receiving a forager and an ack_forager, as well as a non-BeeIP data packet which has to be piggybacked are illustrated in Algorithm 3 and Algorithm 4 respectively. In figure 4.5 the important steps in terms of receiving an ack_forager at the BeeIP source node are highlighted. The source node updates its internal routing tables by emulating the special honeybee dancing. Then, the packet is given to the next

Algorithm 3 Receiving packets at a BeeIP node: Part 3 - The (ack_)foragers

Input: A packet *pkt* received from either the upper or the lower layer

```
41:     if $pkt.Type$ is forager or ack_forager then
42:         $result \leftarrow$ recvForager($pkt$)          ▷ See Algorithm 9 of Appendix C
43:         if $result$ is forager at flower then
44:             addWaitingForagerToFlowerQueue($pkt$)
45:         else if result is ack_forager at hive then
46:             emulateDance($pkt$)                          ▷ See Algorithm 5
47:         end if
48:         forwardForager($pkt$)
49:     end if
```

layer, to make sure that any data in its payload is delivered to the application layer. The internal decisions and processes of a BeeIP node in terms of accepting and dealing with forager and ack_forager packets are also shown in the flowchart 4.6.

Algorithm 4 Receiving packets at a BeeIP node: Part 4 - The non-BeeIP

Input: A packet *pkt* received from either the upper or the lower layer

```
50:     if $pkt.Type$ not BeeIP control packet then
51:         $pkt.TTL \leftarrow pkt.TTL - 1$
52:         if $pkt.TTL = 0$ then
53:             drop($pkt$)
54:             return
55:         end if
56:         if $pkt.Dest$ is me or BROADCAST then
57:             sendToUpperLayer($pkt$)
58:             return
59:         end if
60:         // This is a new data packet from the upper layer.
61:         $ppkt \leftarrow$ allocateToForager($pkt$)
62:         forwardForager($ppkt$)
63:     end if
64: end for
```

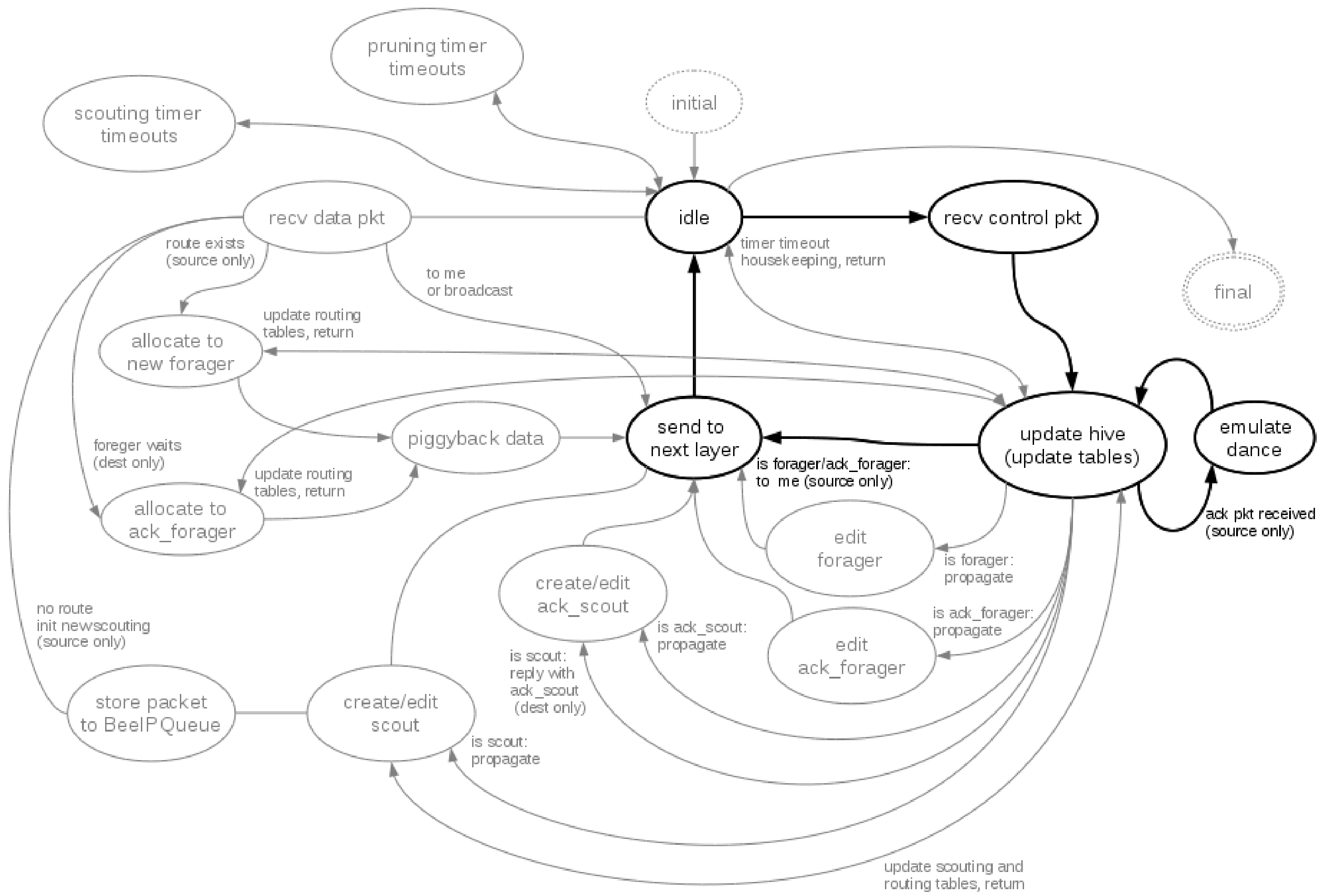

pruning timer timeouts
scouting timer timeouts
initial
idle
recv control pkt
final
recv data pkt
route exists (source only)
timer timeout housekeeping, return
to me or broadcast
allocate to new forager
update routing tables, return
send to next layer
update hive (update tables)
emulate dance
piggyback data
foreger waits (dest only)
update routing tables, return
allocate to ack_forager
is forager/ack_forager: to me (source only)
ack pkt received (source only)
edit forager
is forager: propagate
create/edit ack_scout
no route init newscouting (source only)
is ack_scout: propagate
edit ack_forager
is ack_forager: propagate
is scout: reply with ack_scout (dest only)
store packet to BeeIPQueue
create/edit scout
is scout: propagate
update scouting and routing tables, return

Figure 4.5: A diagram which highlights the process of foraging and, in particular, the receiving of an ack_forager packet at the BeeIP source node of a communication session.

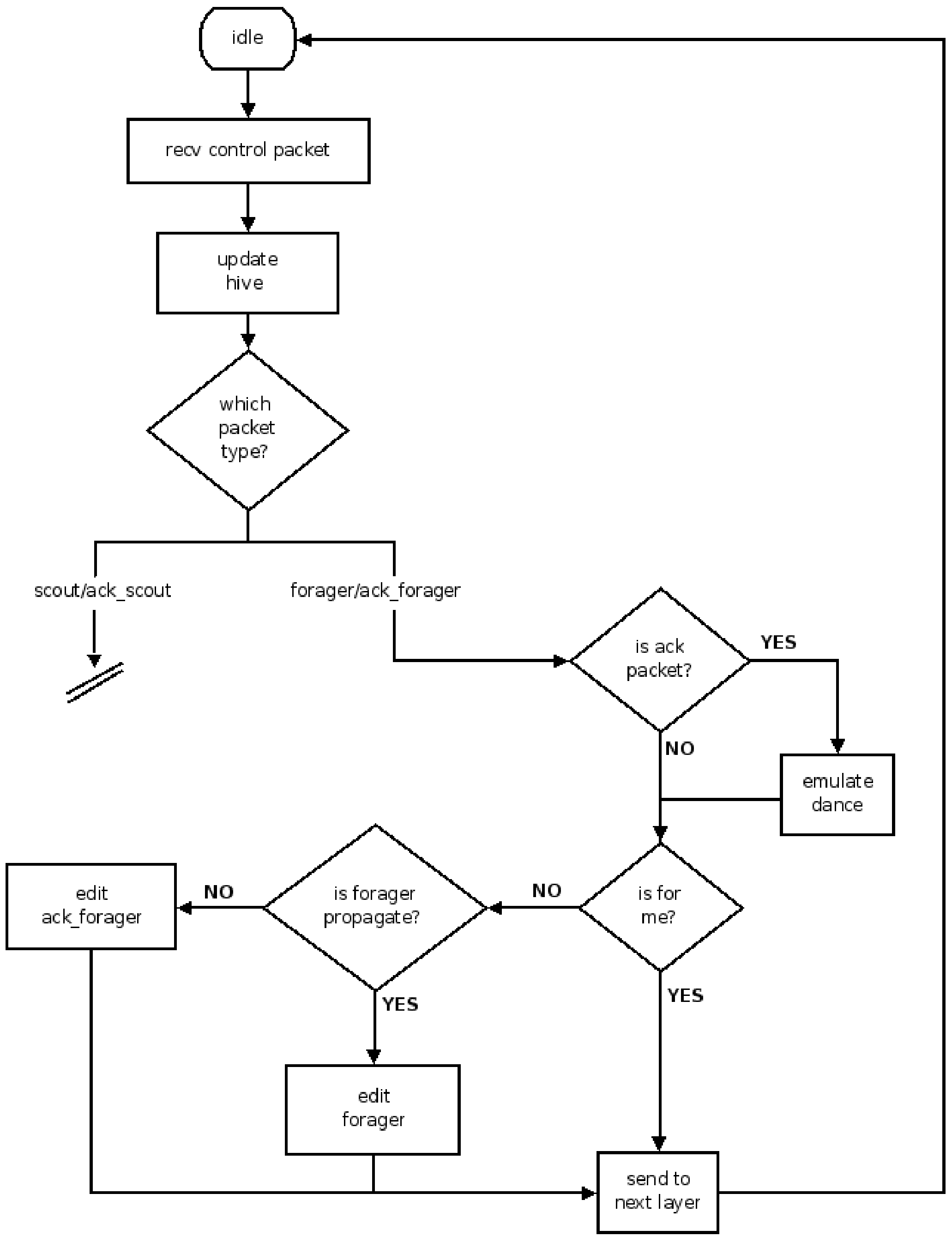


Figure 4.6: A process flowchart that presents the internal workings of a foraging process of a BeeIP node. The scouting part is omitted. Here, the concentration is on the way BeeIP deals with forager and ack_forager packets at the nodes where control packets need to be either propagated or delivered directly to the upper layer.

4.4.2 Routing information and tables

Every time an ack_scout is received, the BeeIP node acknowledges a path and stores the appropriate routing information to its routing table. The information is saved in such a way, that it can be easily accessed and updated when needed. What is kept each time depends on the role that the node plays for a given communication session. For instance, a source node is required to keep more details regarding a specific path, because it will have to decide the appropriate one at the beginning of the next transmission. An intermediate node on the other hand, does not influence the transmission, apart from indicating the next hop to a visiting forager, or ack_forager.

A routing table row holds information about a single path between a source and a destination. It is a collection of values which can be retrieved using the unique path's identification number. Common routing information stored in a row includes a flag to indicate whether the path is acknowledged, its length in hops, and the IP addresses of the neighbouring nodes which are used as the next hops to either the source or destination. In the case of the two end peers of the path, source and destination, the next hop is set to the node's own IP address. Furthermore, a routing table row includes a time-stamp value which is updated every time the row is used to provide a routing solution. This allows it to stay active and to avoid being removed by the housekeeping timer.

In addition to the common routing information, a source node stores two values that are related to the population of the recruited foragers. The number of foragers that are allowed to use the corresponding path in future transmissions, and the number of them who are already using it. Both numbers describe the foraging capacity of the path, which is used to bias the distribution of packets across multiple paths. Mapping nature to networks, the number of foragers that are allowed to use the path is considered as the number of unemployed foragers, which are grouped and allocated to use the particular path. As in nature, they are normally waiting in the dance floor for new dance releases to occur. Any foragers that are already recruited and sent off from the hive are expected to return back as ack_foragers and are also used for detecting link failures. In real honeybees, the number of foragers that become interested in working towards a specific source of food is controlled by the number of dances being released. In turn, the number of dances is related to the enthusiasm of the fellow honeybees who report the food's goodness. Inspired by this behaviour, in BeeIP the foraging capacity of a path changes according to the outcome of its monitoring activity. Every time that an improvement is detected, a new artificial dance will be released and a small number of artificial foragers will be recruited, or in other words, added to the number of unemployed but waiting foragers.

As introduced and described in chapter 3, real honeybees possess the ability to understand and evaluate the quality of a source of food, based on the experience of their previous foraging activity. In this work, this feature of honeybees has been investigated and mapped to the world of networks. Honeybee experience is represented by keeping a record of the previous monitoring outcomes, delivered by successful ack_foragers for the same path. The record is a matrix of two columns, namely time-stamp and path quality. Each time an ack_forager returns back to the source, a new pair of data is added to the matrix, updating in that way what the artificial honeybees know about the performance of the path. To keep the information fresh, the matrix always contains data from the last most recently received ack_foragers. The size of the matrix depends on how many memories foragers need to have in order to be able to detect any improvement or deterioration over time. The default value is empirically set to 10 and is directly related to the level of adaptability the protocol is required to achieve. Experiments have shown that in most cases, a number of 10 memories allow a satisfying detection of changes to the path's quality, where at the same time it balances the relation between the time that is required to collect the memories (each memory is a result of a single flight, i.e., a packet transmission), and the value of the detection based on them. The larger the size of the matrix the longer the time that is required to be spent in order to collect the memories, and less representative sample is generated in order to make efficient detections. The evaluation of the performance as well as the methodology of using the matrix data to understand a paths usability and adjust its foraging capacity, are given in section 4.6.3 of this chapter.

4.5 Internal components of a BeeIP node

Figure 4.7 illustrates the internal interconnected components that constitute a BeeIP node. Each component has unique networking responsibilities, while at the same time mimics the internal structure of a real honeybee hive.

4.5.1 Entrance

A communication session between two nodes begins when the source is required to forward data to the destination. In a typical TCP scenario, having received data from the upper layers, the transport layer prepares segments and forwards them down to the network layer for further actions. Likewise, when the physical layer picks up data from the wireless medium, it ends up being sent to the network layer. The entrance, figure 4.7, provides the link of communication to either one step up or down the OSI network stack (Zimmermann, 1980). Also, a screening process for every incoming packet is done at this component. Similar to nature, where unknown honeybees are not welcome in the hive unless they have adopted the hive's common odour, unrecognised packets are not allowed to enter until they are converted to BeeIP foragers. An exception is made for those packets that target

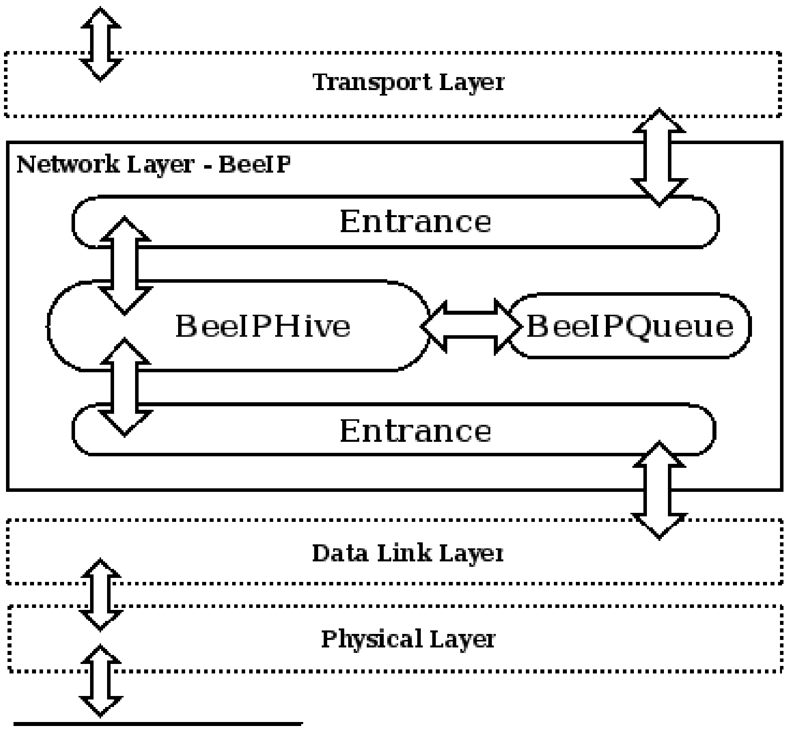


Figure 4.7: The BeeIP internal structure at the network layer of a node. It consists of three interconnected components (Entrance, BeeIPHive and BeeIPQueue) which handle the incoming and outgoing packets from and to the other layers of the network stack. Notice that the entrance is a unique component which is used to pass packets to either one layer up or down the network stack and communicates with the BeeIPHive only.

the source node itself, for instance broadcast packets. Such packets are forwarded directly to the upper layer. The de facto rule is that unless there is another routing protocol operating within the same network, all incoming packets from the wireless medium should have been already converted to BeeIP compatible.

4.5.2 BeeIPHive

This component is the place where the business logic of the routing protocol takes place. It is where information regarding scouting, routing and neighbouring nodes is stored and maintained. BeeIPHive is also responsible for controlling the population of the artificial honeybees, i.e., the BeeIP packets. In terms of nature to network mapping, the BeeIPHive component can be considered as the inner parts of the hive; the artificial combs. In order to describe the BeeIPHive better, it is separated into four areas as illustrated in figure 4.8.

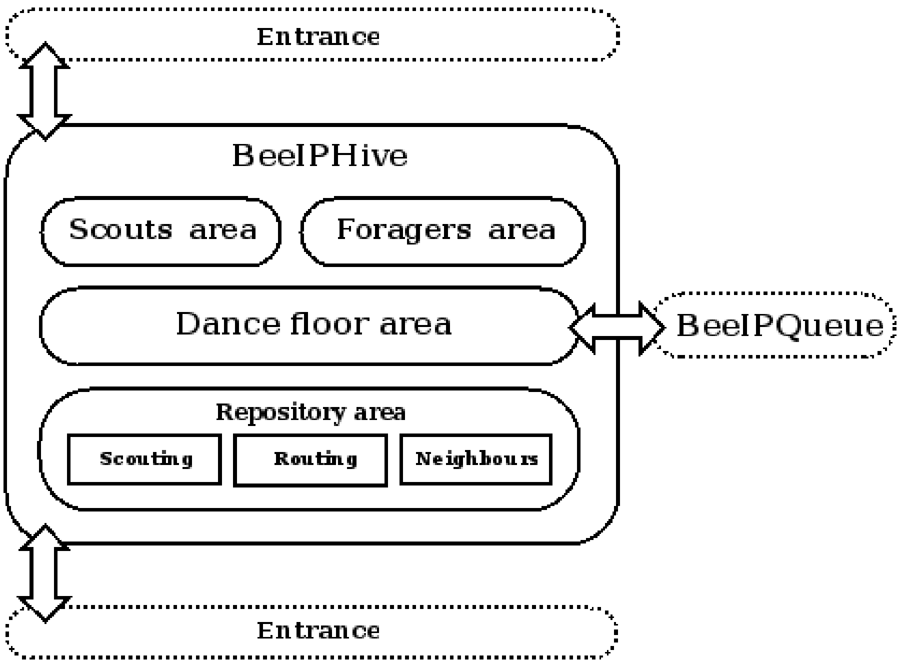


Figure 4.8: The internal structure of the BeeIPHive as one of the three components of a BeeIP node.

Scouts area

To start with, the scouts area is responsible for handling both scouts and ack_scouts. When either of these packets enters the BeeIPHive, the information found in its header is delivered to the scouting and routing repositories. If the packet is an ack_scout, the neighbouring nodes repository is also updated. Also, the scouts area of the BeeIPHive is where scouts and ack_scouts are prepared in order to be propagated by broadcast, or sent by unicast, to the appropriate next hop.

Foragers area

The foragers area controls any foragers and ack_foragers which are created, received and sent by the BeeIP node. Same as in ack_scouts, the neighbouring nodes repository is updated when an ack_forager is received. The foragers area is firmly collaborating with the routing repository and the dance floor, in a manner that mimics real honeybees and their dancing activity. In nature, available foragers are recruited at the dance floor of the honeybee hive, in where they witness other fellow honeybees' dance and get attracted by them. During the recruitment, a real honeybee will collect what she needs to know in order to find the particular source of food. This is how they communicate in order to participate in a collaborative labour towards the collection of food. In BeeIP, foragers are recruited in the dance floor area of the BeeIPHive. They are given the unique identification number of the appropriate path, and the next hop towards the destination. These details are read from the routing repository.

Another responsibility of the foragers area is to provide a temporary home for the incoming foragers, when the current node is the destination of the path, i.e., the destination node of the communication session. This emulates the phase of the real foraging, when the honeybee stays on the flower in order to collect the nectar into her honeybee stomach, or grab a ball of pollen with her legs. This area is not to be confused with the BeeIPQueue component, which is used to buffer the foragers during a scouting process (details in section 4.5.3).

The dance floor area

Following the discussion above, the dance floor area is the place where the recruitment of foragers is implemented. Each incoming ack_scout and ack_forager delivers fresh knowledge regarding a particular path. The knowledge is added to the routing repository, the place where the routing tables lie. Then, along with the knowledge from previous flights, it is used to evaluate the performance of the path and understand its ability to serve based on experience. Both evaluating and understanding are achieved during an artificial honeybee dance, a method inspired by the real waggle dance found in the world of honeybees (explained in 3.3.3). Based on the outcome of the evaluation, adjustments are made to the path's foraging capacity, affecting in that way the maximum number of forager packets that are allowed to be recruited. The artificial dance is explained in more detail in section 4.6.4.

In the dance floor within a real hive several honeybees might dance at a time, advertising different interesting sites. The decision of which dance triggers a willing honeybee depends on her location on the dance floor, and the strength of the signals she receives from the dancing honeybee[2]. For instance, if two dancers perform at the same time, the one who dances more enthusiastically will receive more attention. This behaviour is also emulated in the BeeIPHive's dance floor. Unemployed foragers are allowed to select the most appropriate path from a list of alternatives, based on the selection path metric, described in section 4.7 in detail.

Repository areas

The neighbouring nodes repository, as well as the scouting and routing repositories, are used to store all the data required by BeeIP to function. There are in fact three tables of data, where rows are created, retrieved and removed according to the protocol's mechanisms. For instance, the low-level values, collected from and delivered to the adjacent nodes by the returning ack_scouts and ack_foragers, are stored in the appropriate rows of the neighbouring nodes table. Due to limited resources often found in mobile nodes, good memory management is required. Therefore, clock timers are dedicated to removing old and unused table rows, by checking the time-stamp when they were last used. A typical neighbouring nodes entry includes:

- The time-stamp of the last time the information was used.
- The signal strength of the last received ack_scout or ack_forager at the receiving node.

[2]Assuming that both the dancers and the unemployed foragers belong to the same group, and work towards the same commodity.

- The sender's last reported remaining energy level.
- The sender's last reported speed (velocity).
- The last reported size of the queue at the sender's MAC layer.
- The transmission delay between the sender and the receiver (current node).
- The result of the (local) quality calculation for the link between the sender and the receiver (current node).

Apart from the obvious time-stamp and local quality values, the rest are the five low-level parameters used to evaluate the quality of the link between the two adjacent nodes. A complete discussion and reasoning for selecting the parameters is given in section 4.6 and, in particular, in section 4.6.1. In terms of the scouting repository, a typical entry includes:

- The time-stamp of the beginning of the scouting process.
- The IP address of the source node which started the scouting process.
- The IP address of the target destination of the scouting process.
- The type of the node, can either be "hive" (source), "path" (intermediate), or "flower" (destination node).
- A flag for acknowledging the success of the scouting process.
- A time-to-live value (source nodes only).

The time-to-live is only set at a source node, when the type is also set to "hive". Its value is used to control the propagation of the scouts while they are broadcast to the network. The methodology is explained in 4.4.1, "The scout". Finally, a typical routing information entry consists of:

- The unique path identification number.
- The role of the node in terms of the particular path. Can either be "hive" (source), "path" (intermediate), or "flower" (destination node).
- The time-stamp of the last time an ack_forager returned back to the source using this path.

- A flag for acknowledging the path, active or not active.
- The IP address of the next hop towards the source. Set to the current node's IP address if it is in fact the source of the path.
- The IP address of the next hop towards the destination. Set to the current node's IP address if it is in fact the destination of the path.
- The number of hops within the path.

The details above are common to all types of routing entries, namely "hive", "path", and "flower". However, those which are created with their type set to "hive", the source node of the path, also include the following:

- The number of foragers that are allowed to be recruited, and use the particular path.
- The number of foragers that are already using the path and are expected to return back as ack_foragers.
- The value of the metric to be used as the appropriate path selection metric.
- The two-column matrix, time-stamp and path quality, which is used to keep the past experience of the performance for the particular path.
- The number of rows already in the matrix.

4.5.3 BeeIPQueue

The BeeIPQueue is the last component of a BeeIP node, used to buffer all the forager packets whose path is not yet discovered. Coupled with the dance floor previously described, the BeeIPQueue can be considered as the place where artificial foragers wander around until they are recruited by the dance of a fellow artificial honeybee. Once a data segment is prepared by the transport layer and the routing protocol is required to forward it to the correct destination, a forager is created to carry it further. A scouting process begins when no path is available in the routing tables of the source, thus, no path identification number can be assigned to the newborn forager. At this stage, the forager is added in the buffer and is willing to accept the call of a fellow artificial honeybee dancer. This maps the behaviour met in the real hives. Moreover, the queue can host multiple foragers for the same destination. When an ack_scout finally arrives at the source, the corresponding foragers leave the queue in a first-in-first-out manner. This schedules the packets and ensures that they will at least leave the source in the correct order.

BeeIPQueue has a default capacity of 40 packets. A clock timer ensures that old packets do not stay in the queue for ever. Packets older than 4 seconds are silently dropped. The numbers are configurable and depend on the implementation. However, during the experiments of this research work, no packet has ever been waiting in the buffer queue for more than 2 seconds.

4.6 Routing maintenance

As already mentioned in previous sections, BeeIP is a multi-path routing protocol. Each scouting process may result in more than one path discovery, the details of each are stored in the local routing tables of the nodes involved. Inspired by real honeybee behaviour, BeeIP routing maintenance concentrates on the contiguous quality monitoring of all working paths. In order to achieve that, ack_scouts and ack_foragers constantly collect low-level parameter values from the nodes they visit, and deliver them to their next adjacent hop. Moreover, BeeIP monitoring activity is based on the assumption that the transmission links between the nodes are bidirectional. An explanation is given in section 2.3.2.

Nevertheless, ack_scouts and ack_foragers monitor the path on their way back to the source. BeeIP is designed to match nature and real honeybees as much as possible. Foragers that return back to their hives with a honey stomach full of nectar have completed a successful foraging cycle. That is itself proof that the path is at least not broken at the time of the particular trip. Similarly, when either an ack_scout or an ack_forager packet returns back to the source, it is automatically implied that the corresponding routing path is still usable. However, its performance depends on the quality value being carried within the header of the packet.

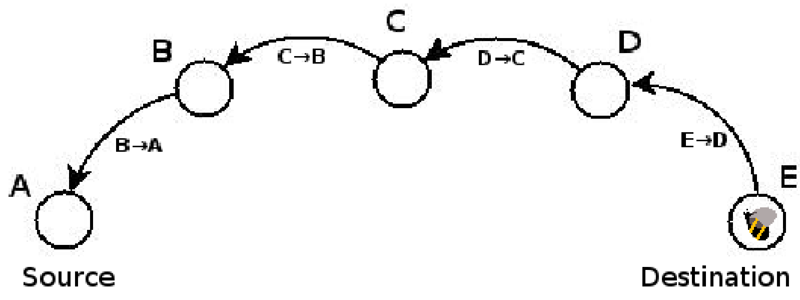


Figure 4.9: An artificial honeybee is monitoring the quality of the path, by measuring the quality for each intermediate link. The packet carries low-level parameter values from each node, and delivers them to the next hop in the path, towards the destination. Starting from destination E, to D, etc., until it is back to the source A.

4.6.1 Monitoring the quality of links

Monitoring the quality of a path is the most important action performed by both ack_scouts and ack_foragers (ack packets). To be able to understand and define the quality of a path, the path is split into a chain of links between the intermediate nodes from the destination back to the source. While traversing it, the ack packet is responsible for collecting low-level parameter values that represent not only the current state of its senders, but also, the network effectiveness of each intermediate link. These are then delivered to the next hop, where the quality of the previous link is measured. The ack packet collects the result and continues the journey back. The process repeats until it is back at the source node. An example is given in figure 4.9. When flying from the destination node E to the source node A, the path quality is calculated link by link, i.e., $E \rightarrow D$, $D \rightarrow C$, $C \rightarrow B$, and finally $B \rightarrow A$. The list of the low-level parameters used by the protocol, followed by a brief explanation of their importance is given below.

1. The ack packet's signal strength at the receiving node in dBms or Watts.
   When received at a node, the ack_scout or ack_forager packet's signal strength is measured. A weak signal strength can be an indication of long distance between the two nodes, intermediate obstacles, or both.

2. The moving speed of the sender (velocity) in m/s.
   A moving node can easily go outside the transmission range and cause weak or broken links. For the same reason, a node with a fixed position is more

promising than one which moves around constantly and at high speed.

3. The sender's remaining energy level in Joules.
   Nodes with sufficient remaining energy are less vulnerable and better candidates for future packet transmissions.

4. The size of the MAC queue of the sender in bits.
   The precise size of the queue at the MAC layer of the sender is an indication of how busy the sender is, in terms of traffic and network congestion.

5. The transmission delay between the sender and the receiver of a link in seconds.
   The use of time-stamps and synchronized clocks allows the measurement of the time an ack packet requires to be transmitted from the sender to the receiver in a link.

The parameters above can be retrieved by accessing the appropriate layers of the OSI model, using cross-layer design (Conti *et al.*, 2004). Unlike the traditional way, where each layer of the stack is allowed to communicate only with its adjacent layers (up and down), cross-layer designing allows information to be exchanged between any layers (see section 2.7.3 for details). Figures 4.10 and 4.11 illustrate the different designs, in the context of BeeIP. Traditionally, the network layer is not allowed to communicate with the physical layer or the application layer. However, in BeeIP sharing information between those layers is an important aspect of the protocol's design.

The application layer can provide both the moving speed and the remaining battery strength on the node. In a similar way, the physical layer can provide the packet's signal strength upon its reception (Ogunjemilua *et al.*, 2009; Bardwell, 2002). Also, knowing the size of the MAC layer's queue (in the data link layer[3]) and the allowed data rate as defined by the 802.11 network standards (Board, IEEE-SA Standards, 2003) of the wireless adaptor, the queuing delay can be found. Finally, the transmission delay is calculated by considering the time-stamps of sending and receiving the packet between the two nodes of a link.

Furthermore, the parameter values are of different scales and have difference units. In order to be used effectively, they need to be normalized. The normal-

[3] ISO model describes the MAC (media access control) layer as a lower sub-layer of the data link layer. The upper sub-layer is called LLC (logical link control).

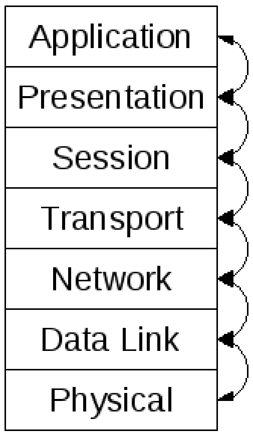


Figure 4.10: OSI model traditional design. Only adjacent layers are allowed to communicate with each other following a waterfall fashion.

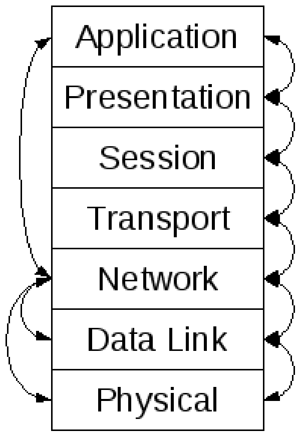


Figure 4.11: Cross-layer design for BeeIP. The network layer dynamically communicates with the application, DLL, and PHY layers.

ization can be done by knowing the minimum and the maximum values expected. Using cross-layering, the protocol is able to acquire these values, as they are either part of the network configuration (set by the wireless adaptors, network standards, etc.), or set by the software controllers (e.g., the driver which controls the battery). Normalizing the values is done by applying the linear transformation (see B.1 of Appendix B). It is important to notice here that the moving speed, and both queue size and transmission delay are adversely affecting the performance. This is better understood with an example. A node's speed equal to zero does not affect the transmission, as it does not alter the distance between the source and the destination. This makes it preferable compared to a node of high moving speed. Likewise, a node which has the smallest MAC buffer size is considered free of congestion. Hence, inverse normalization is used for these parameters.

The next step is to utilize these numbers so that the quality of the link can be represented. As stated above, each parameter describes a unique aspect or characteristic of the communication between two neighbouring nodes. Their role is important but they differ in the way they affect the performance, and thus, the quality. Therefore, to use them efficiently, a way to express their relative importance needs to be found. In BeeIP, a weighting system is used for this purpose. The formula to calculate the quality q of a link from node j to k as traversed by the ack packet b:

$$\begin{aligned} q_{jk} \quad = \quad & sig'_b * w_{sig} \\ & + speed'_j * w_{speed} \\ & + energy'_j * w_{energy} \\ & + qd'_j * w_{qd} \\ & + txd'_{jk} * w_{txd} \end{aligned} \tag{4.1}$$

where the prime numbers are the normalized values of the parameters: *sig* for signal's strength, *speed* for current speed, *energy* for remaining energy, *qd* for queuing delay, and *txd* for transmission delay. The $w's$ are the appropriate weights.

Finding an appropriate set of weights for the formula 4.1 is a separate and complicated problem. In a previous version of BeeIP published in Giagkos and Wilson (2010), an attempt of producing a weighting system based on experience was presented. Table 4.5 summarizes the resulting weights found empirically. Notice that the numbers are used to represent the importance of each parameter, therefore, they are a way of prioritizing them and express which plays the most and least significant role in affecting the performance of the communication link.

Parameter:	Signal	Pow Speed	Energy	Q-Delay	Tx-Delay
Weight (w):	0.40	0.20	0.20	0.15	0.05

Table 4.5: A weighting system based on experience. The weights are a first attempt to prioritize the low-level parameters based on their importance, in terms of the performance of the communication link between two nodes.

Although the results were quite promising, it was recognized that finding the relative importance between the low-level parameters was subject to further investigation. The next part of this section describes the methodology used to produce a new and more accurate set of weights, using Machine Learning (ML) (Goldberg, 1989) and, in particular, Artificial Neural Networks (ANN) (Gurney, 1997; Haykin, 1994; Lau, 1992).

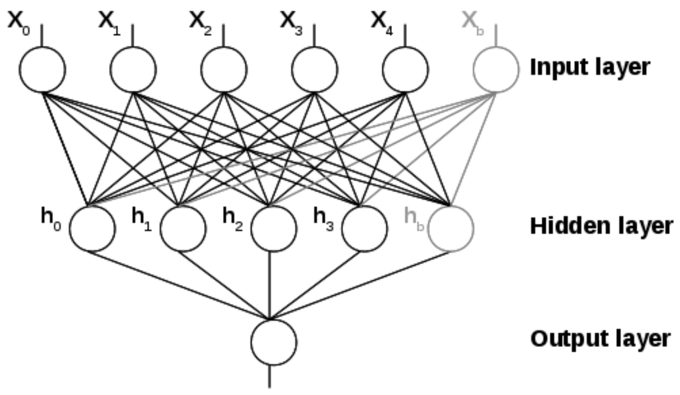


Figure 4.12: The Multi-Layer Perceptron used to generate an accurate set of weights. Each input parameter corresponds to one of the low-level parameter used by BeeIP to measure the quality of a link between to nodes. The neural network was trained by off-line supervised learning based on data sets produced from the wireless network by data packet transmissions. The output was set to the round trip time of each packet.

4.6.2 Towards a weighting system

An artificial neural network is a parallel system, capable of resolving paradigms that linear computing cannot. It is an interconnected assembly of simple processing units, the neurons, whose functionality is based on the human or animal brain. Neural networks consist of different layers or neurons, either fully connected or partially connected. Each neuron's processing ability is represented by its connection strength, the weight, obtained by a learning process. ANNs are able to solve a number of different problems in different areas, including classification (pattern recognition), and regression analysis or function approximation (Zhang, 2000; Zainuddin & Pauline, 2008). Furthermore, they can help in order to understand the relative importance between each input, and their influence to the output of the neural network.

A multi-layer perceptron (MLP) (Gurney, 1997; Trenn, 2008) is a neural network able to express a rich variety of non-linear decision surfaces of the pattern space. It consists of the input layer, the hidden layer (or layers) and the output layer. In each neuron in the hidden and output layers, the activation function is non-linear. Using a sigmoid function is expected to reduce the mean squared error (MSE) of each epoch and produce a more accurate weighting system.

In BeeIP, an MLP was used as a way to generate the set of weights. This was achieved by applying off-line supervised learning based on data sets produced by the wireless network. During its training, the MLP was supplied by a variety of patterns. A pattern consists of five inputs, each one representing a low-level parameter used in formula 4.1, and an actual output. For the latter, the round trip time (RTT) between two nodes was used. The reason is that RTT is affected by all the input parameters, as well as it is a collection of delays that exist and cannot be measured by the protocol itself. For instance, the RTT of a transmission includes the propagation delay of the packet, which is connected to the distance between the nodes. It also includes the computation delays within both the transmitter and the receiver nodes. The goal of the training was to find a set of weights that will cause the output from the neural network to match the actual target values of the data sets, as closely as possible.

Honeybees are empirically found to make judgement based on prior foraging knowledge and current quality observations. Von Frisch's experiments have illustrated the existence of the quality factors and the honeybees' ability to combine them together in order to understand the goodness of their findings. Although he proved that the quality factors exist, the exact values of their thresholds are still unclear to us (von Frisch, 1967). Additionally, Seeley on page 121 states: "*A forager must have some means of knowing whether the flower patch she has just visited represents a source of nectar of low, medium, or high quality. [snip] It is now clear that each nectar forager independently assesses the profitability of her flower patch by integrating information about energetics of foraging at her particular patch.*" (Seeley, 1995). As described before, the aim of the set of weights in BeeIP is to illustrate the relative importance of the 5 parameters in connection to what can be generally considered as a working path of good performance. Combining those parameters together is directly mimicking real honeybee behaviour by which the honeybees understand whether the finding is worth exploiting (and dance for it within the hive) or not by collecting information from the environment (quality factors), and considering prior knowledge (waggle dances at the dance floor). An MLP trained with data that represent both normal and extreme network conditions, such as the energy of the nodes drops to 0, the MAC buffers reach their limit causing TCP packets to wait for long time and eventually drop, and the node's moving speed varies between low to unrealistically high values causing frequent changes to the distance between transmitters and receivers, allows a generic set of weights to be discovered and used to combine the quality factors more efficiently. The values of the parameters being optimized are not influenced by BeeIP algorithm. In fact, BeeIP does not make any changes to those parameters, rather, the artificial honeybees collect their values from the artificial environment (i.e., the layers of the network stack of each node they traverse) using cross-layering and use them in order to represent the quality of the links and paths.

Figure 4.12 illustrates the MLP which was used in this work. It consists of one hidden layer of four neurons. The grey neurons x_b and h_b are the bias neurons. MLPs with no hidden layers are unable to solve complex problems (non-linear), whereas those with one hidden layer are able to solve most of the problems sufficiently. Two hidden layers are required for modelling problems of data with discontinuities (e.g., a saw tooth wave pattern). Based on the literature, there is currently no theoretical reason to use neural networks with any more than two hidden layers (Trenn, 2008). The number of neurons that exist in each hidden layer is also important in order to avoid several problems. For instance, under-fitting occurs when there are too few neurons in a hidden layer to adequately detect the signals in a complicated data set. On the contrary, too many neurons may lead to over-fitting, a case when the neural network has so much information processing capacity, that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layer. Another problem of using too many neurons is that it increases the time it takes to train the network. Nevertheless, training the MLPs in the context of BeeIP happens off-line, and this problem is ignored. The decision to use four neurons in the hidden layer, is driven by the three rules of thumb, generally accepted in the literature (Swingler, 1996). The number of hidden neurons should be:

- Between the size of the input layer and the size of the output layer $(5 > 4 > 1)$.
- 2/3 the size of the input layer, plus the size of the output layer $(3.3 + 1 \simeq 4)$.
- Less than twice the size of the input layer $(4 < 10)$.

To train the MLP, the well-known back propagation algorithm (McClelland & Rumelhart, 1986) was used, as it is the most widely used algorithm for training feed-forward neural networks. Although it is seen from simulations that it takes a long time to converge (Kuan & Hornik, 1991), in BeeIP's content, training happens off-line and thus, it is not a real issue. Table 4.6 lists all the training parameters.

Training algorithm:	Back Propagation
Datasets:	10 x 10.000 patterns
Training duration:	1000 epochs
Normalization:	0 to 1
Initial weights:	-0.5 to 0.5
Learning rate:	α set to 0.2

Table 4.6: The MLP training parameters.

A number of different methods exist in order to measure the importance of each input in terms of the output in back propagation networks, such as the above MLP (Sung, 1998b). These include sensitivity analysis (SA) (Saltelli *et al.*, 2000), change of mean squared error (MSE) and fuzzy curves (FC) (Lin & Cunningham, 1995). One important characteristic of both SA and change of MSE is that they require training the neural network, whereas the FC only analyses the inputs/output relationships by plotting and observing the resulting fuzzy curves of each input (Sung, 1998a). In this work, the method of observing the change of MSE is used, due to its simplicity. In more detail, the resulting MSE of the trained neural network when all the inputs are present, is compared against the MSE obtained from the trained network when one of the inputs is omitted. The idea behind this methodology, is to observe the differences between the MSE values, and thus make verdicts about the importance of each input accordingly. The input which produces the biggest MSE difference when missing from the neural network, is the one with the highest importance. The simplified version (only one neuron in the output layer) of the MSE function as documented in Sung (1998a) is:

$$MSE = \sum_{p=1}^{P} (target_p - actual_p)^2 / P \tag{4.2}$$

where, $target_p$ and $actual_p$ are the desired output and the calculated output respectively, of the output neuron for the p^{th} pattern. P is the total number of patterns.

Using the formula 4.2 to calculate the MSE, training the MLP is repeated six times. One with all inputs present and five with an input missing each time. The summary of the results is found in table 4.7.

Again, the degree to which both the signal strength and the remaining energy affect the performance is clearly high, matching those empirically determined. An interesting observation is that the difference that the speed parameter makes to the MSE is a very small number. The speed of the nodes does not affect the performance of the link as much as it was thought in the first weighting system. The reason is the lack of knowledge in connection to the directions of the moving node. While two nodes are moving towards its other and the distance between them is getting smaller, their connection link is enhanced. On the contrary, if they move at opposite directions the distance is increased and their connectivity weakens. Clearly the speed could be a more valuable parameter for the system, if it was connected to the distance between the nodes. However, the version of BeeIP described in this work does not possess the mechanism to accurately find the distance of the two nodes. Also, in reality, the distance is often not available. Hence, the parameter of speed is weighted with the small outcome number, to represent the small impact it has on the performance.

4.6.3 Monitoring the quality of paths

Following the discussion in section 4.6.1 regarding the quality of a link, a new q_{jk} (formula 4.1) is calculated at every node visited by the ack packet, and when it finally arrives at the source node, the quality of the path from the destination d to the source s can be expressed as:

$$Q_{ds} = \sum_{n=1}^{m-1} (q_{N_{n+1} \to N_n}), \qquad [d = N_m,\ s = N_1] \tag{4.3}$$

where m is the total number of nodes in a numerically ordered path, and $N_{n+1} \to N_n$ the pair of nodes with direction towards the source node.

The result obtained by formula 4.3 is a number that can be used to represent the current quality of the path, in terms of the five low-level parameters found in each intermediate node. However, it represents only one single flight. The nature of the network is such, that obtaining and using the results of only a single

	All present	Missing parameters				
		S.Strength	Energy	Speed	Qu delay	Tx delay
Dataset 0:	0.002321	0.009162	0.008905	0.002358	0.003531	0.003249
Dataset 1:	0.001683	0.005643	0.005185	0.001880	0.004042	0.004493
Dataset 2:	0.001963	0.003975	0.004039	0.002234	0.002727	0.002655
Dataset 3:	0.003640	0.011833	0.011936	0.003714	0.005993	0.006500
Dataset 4:	0.000895	0.001496	0.001515	0.001348	0.001920	0.001718
Dataset 5:	0.002062	0.006353	0.006891	0.002155	0.004391	0.002456
Dataset 6:	0.003306	0.010000	0.009404	0.003526	0.003984	0.004705
Dataset 7:	0.000913	0.003152	0.003151	0.001132	0.000916	0.001192
Dataset 8:	0.002273	0.006243	0.006379	0.002145	0.002979	0.002428
Dataset 9:	0.001805	0.005780	0.006073	0.001834	0.002282	0.002751
Avg MSE:	0.002086	0.006364	0.006348	0.002233	0.003277	0.003215
Difference:		0.004278	0.004262	0.000147	0.001190	0.001129
Weight %:		**38.87**	**38.73**	**1.33**	**10.82**	**10.26**

Table 4.7: Results obtained by measuring the change of the mean squared error (MSE) while training the MLP.

packet transmission can be highly misleading. In a similar way to nature, where honeybees evaluate a finding based on their past experience and knowledge, what is required is a way to evaluate the performance based on quality measurements taken from a number of previous transmissions. Therefore, the latest ten Q_{ds} values along with the corresponding times are collected and stored in the history matrix described in section 4.4.2. Applying regression analysis on these data, the source is able to detect whether there has been an improvement or deterioration over time. Pearson's correlation coefficient (Read & Cressie, 1988) is used for this purpose. Rewritten to fit BeeIP's design, the Pearson's r is calculated by:

$$r = \frac{\sum_{i=1}^{k}(t_i - \mu t)(Q_i^{ds} - \mu Q^{ds})}{\sqrt{\sum_{i=1}^{k}(t_i - \mu t)^2}\sqrt{\sum_{i=1}^{k}(Q_i^{ds} - \mu Q^{ds})^2}} \tag{4.4}$$

where t_i is the time of receiving Q_i^{ds} value, μt is the mean of the time column values, and k is the number of flights collected.

The idea of using regression analysis is to catch any strong positive or negative correlation between the two variables: time and Q^{ds}. As real honeybees do in nature, their artificial counterparts must possess a mechanism to detect whether the path they are using remains worthy. In addition, following the principle of honeybee dancing, when they have enough evidence and a clear understanding of the quality of the path, they start the process of recruitment. In the protocol's context, this is translated to collecting enough data from previous flights. If the correlation is a strong negative, then the signal which is given to the recruits is also negative, affecting the path's foraging capacity. The two thresholds for catching the strong correlations are set empirically to -0.8 and 0.8. The protocol's behaviour with numbers between -0.5/0.5 and -0.7/0.7 is found to judge the quality of the paths in an unfair manner, giving penalties and bonuses to paths with no genuine changes, and thus it is unable to narrow down to those cases where a path is close to break. Nonetheless, depending on the implementation these thresholds can be changed, altering the sensitivity of both the monitoring and foraging activities. The closer to -1 and 1 these thresholds get, the less sensitive the protocol becomes in terms of applying penalties and bonuses. Additionally, the smaller the threshold numbers the less accurate the detection. Mimicking nature, the artificial honeybees may dance either because of a strong negative or positive correlation is found. If no assumptions can be made (the thresholds are not reached), then no dance is released and in turn, no adjustments are made to the foraging capacity of the path.

4.6.4 Artificial honeybee dance

Artificial honeybee dance is the technique by which the foraging capacity of a path is changed according to a new evaluation, based on both previous and new quality measurements. There are two different cases when dance is performed, depending on the ack packet being received. It is triggered either by an ack_scout or an ack_forager.

As previously described, every time a new path is found between a source and a destination, new routing information is stored in the routing repository area of the source. Initially, the foraging capacity of the path, the number of foragers that are allowed to be recruited as well as the number of foragers already using the path, are set to zero. However, the first changes according to the advertising of the ack_scout. Since there is no previous knowledge, the ack_scout dances enthusiastically just because a new path is found. As mentioned in chapter 3, real scouting and foraging depends on the energy demand in the hive. If the demand is high, the first performing scout that brings fresh knowledge of a source of food recruits more foragers, in order to satisfy the overall need of energy. Therefore the initial foraging capacity fc_{init} of a path from source s to destination d is set by:

$$fc_{init} = F + D_d \tag{4.5}$$

where F is a constant number of foragers, and D_d the number of packets stored in the BeeIPQueue with destination d (demand).

A value for the constant F has been empirically set to 20 foragers. Having in mind the mechanism for detecting any possible improvement and deterioration over time, the initial number of foragers needs to be at least equal to the size of the history matrix, as it is discussed in section 4.4.2. In principle, this will allow the routing mechanism to operate. However, the nature of the network is such, that packets can often be dropped or lost during the communication

Algorithm 5 Emulating the honeybee dance

Input: A ack_scout or an ack_forager packet *pkt*

```
 1: procedure EmulateDance($pkt$)
 2:     $rData \leftarrow$ getRouteData($pkt.PID$)
 3:     if $rData$ is NULL then
 4:         return
 5:     end if
 6:     // Add last link's information, i.e., the selection metric's to the sum
 7:     // as well as the last link's quality result to the overall path quality.
 8:     editAckPacket($pkt$)
 9:     updateRouteData($rData$,$pkt$)
10:     if $pkt.Type$ is ack_scout then
11:         $adj \leftarrow$ FIRST_RECRUITS + demandFor($pkt.Src$)
12:         $rData.ForagersIn \leftarrow adj$
13:         return
14:     else
15:         // Is an ack_forager, consider prior knowledge:
16:         if $rData.MatrixSize$ = MATRIX_SIZE then
17:             resetMatrix($rData.Matrix$)
18:         end if
19:         pushToMatrix($rData.Matrix$, $pkt.TS$, $rData.PathQuality$)
20:         if $rData.MatrixSize <$ MATRIX_SIZE then
21:             $rho \leftarrow$ PearsonRho(rData.Matrix)
22:             if $rho \leq$ RHO_MINUS then
23:                 $adj \leftarrow rData.ForagersIn - RHO\_ADJ$
24:                 $rData.ForagersIn \leftarrow$ min($adj$, MAX_RECRUITS)
25:             else if $rho \geq$ RHO_PLUS then
26:                 $adj \leftarrow rData.ForagersIn + RHO\_ADJ$
27:                 $rData.ForagersIn \leftarrow$ max($adj$, 0)
28:             end if
29:         end if
30:     end if
31: end procedure
```

session. Therefore, the initial number of foragers, no matter what the demand for the particular destination can be, is preferred to be greater than the size of the history matrix. Experiments have shown that an initial value of double the size of the history matrix has no negative effect to the decision making, as it ensures that the matrix will be eventually populated, whereas at the same time it keeps the foraging capacity within reasonable limits so that penalties (if they need to be applied)[4] will make critical sense to the potential of the path as a good candidate in the future.

Although formula 4.5 ensures that new paths will get a balanced starting number of recruited foragers, the first path is always in favour as it is expected to satisfy the demand at the BeeIPQueue. Nevertheless, the situation may change in the long run, as paths are used by foragers based on their selection metric.

In addition to ack_scouts, ack_foragers also perform the artificial honeybee dance. The foraging capacity of a path is also changing according to the evaluation and the report or the performance by an ack_forager. What interests the mechanism is the two strong correlations between time and Q_{ds}, therefore two thresholds are set as described previously. Adaptation is achieved by making small changes to the foraging capacity of a path, according to the threshold which is exceeded each time. In real honeybees, because of the physical limited number of insects in each hive, the number of recruits responding to a new dance tends to be fixed. Similarly in the routing protocol, the adjustments that are made to the foraging capacity of each path are either positive or negative constant numbers. Table A.1 of Appendix A summarizes the default values of the artificial honeybee dance in BeeIP, and all other default values of the BeeIP implementation used in this thesis. The artificial honeybee dance is illustrated in Algorithm 5.

4.7 Data packet forwarding

Keeping the foraging capacity of each path up-to-date, a list of working and potentially good candidates can be created. However, this is one half of the mechanism to successfully forward data packets. The next half is the selection of the best candidate within the list. Depending on the behaviour of the routing protocol that one may want to achieve, different selection metrics can be applied. Tradi-

[4]Remember that penalties as well as bonuses are small constant numbers which makes the protocol to map the signaller-limited nature of the real colony (see section 3.3.1 for details).

tional metrics are related to the number of hops in a path (shortest paths), the transmission speed of its links (fastest paths), the expected transmission count, the energy cost, the remaining energy, etc. (De Couto, 2004; Cao *et al.*, 2007; Balaji *et al.*, 2011).

4.7.1 Appropriate path metric

Understanding the importance of the metric and its influence on the behaviour of the protocol is very significant. This is better conveyed when looking at the following example scenarios.

Scenario 1. Speed of transmission does matter: A team of robots is sent to investigate an area where a disaster took place. They are all connected via a mobile ad hoc network operating with BeeIP. An aerobot is flying high enough to make use of its on-board cameras and take shots of the unknown and probably dangerous terrain. The resulting shots reveal that there is a cliff in front, which cannot be detected by the fellow robotic rover that accelerates towards it. Moreover, other robots in the topology exchange interesting data with each other and keep the network busy. Clearly, in such situation the aerobot will need to communicate and send a stopping signal to the moribund robotic rover as fast as possible. The most appropriate path needs to be the one with the smallest transmission delays between the source and the destination.

Scenario 2. Saving energy does matter: Backing up data is scheduled at an off-peak hour, where the network is expected to be less busy. Large blobs of data are required to be transferred from node A to node B, making sure that less energy is consumed. Furthermore, a condition is set to exclude those nodes whose remaining energy level is below a threshold, and thus using them may get them shut-down and go off-line. In this example, speed is not an issue. Rather, the goal is to use the least power possible. The most appropriate path needs to consider the energy cost between each node, and the remaining energy of their batteries.

The design of BeeIP allows the selection metric to be implemented according to the desired behaviour of the protocol. This is achieved by considering it as a separate measurement, the outcome of which is applied to a populated list of alternative candidates. In the version presented in this thesis, a selection metric relevant to the speed of the transmission is used. At each intermediate node i, the selection metric m for the link between i and its adjacent node j is calculated as:

$$m_{ij} = txd_{ij} + qd_j \tag{4.6}$$

where txd_{ij} is the transmission delay a packet experiences when transmitted from j to i, and qd_j is the queuing delay a packet experiences while being buffered at node j.

At the source node, the overall path selection metric is expressed as the summation of m's which are collected during a flight back. Finally, the path selection metric is used in order to choose the most appropriate path from a list of interesting candidates. In order to participate in the list, a path needs to be acknowledged (acknowledgement flag $= 1$) and have at least 1 recruited forager waiting to use the path (foragers waiting ≥ 1). Algorithm 6 illustrates the selection of the appropriate path at the source node.

At the intermediate nodes, the selection is done by considering the direction and the path's identification number. Remember that every node which participates in a path has up-to-date information about the path and the two nodes used to forward packets, towards either the source or the destination. Finally, at the destination node the selection is done in a first-in-first-out fashion. Clearly, the complexity of the next hop selection is small because it is not based on any searching algorithm. This approach not only palliates the retransmission delays, but it also keeps the size of the packet small, allowing other useful routing information to be accommodated, if necessary.

4.8 Detecting link failures

Despite the constant monitoring and evaluation of the paths, where agents can detect disturbances and prevent loss of connection by changing automatically to another alternative, links between the nodes can still break rather unexpectedly. An adaptive routing protocol ought to have a mechanism to detect when a link has been broken, and update its routing knowledge. Various mechanisms have been proposed and used for this purpose, such as the local repair using HELLO messages in AODV and AntHocNet, or absence of broadcast packets from a

Algorithm 6 Find the appropriate identification number at the hive

Input: A destination *dst*
Output: A unique path identification number or NO_PID

```
1: // Start with something big cause the selection metric is delay.
2: $next\_best\_metric \leftarrow BROKEN\_LINK\_TIMEOUT$
3: for all $rData \in$ ROUTINGREPOSITORYTODEST($dst$) do
4:     if $rData.ACK = 1$ and
           $rData.ForagersIn = 0$ and
           $rData.ForagersOut = 0$ then
5:         $rData.ACK = 0$
6:     end if
7:     $lastReturnedTS \leftarrow$ CURRENT_TIME $- rData.TSLastReturned$
8:     if $lastReturnedTS \geq$ BROKEN_LINK_TIMEOUT and
           $rData.ACK = 1$ then
9:         $rData.ACK = 0$
10:    end if
11:    // $R$ is a list of optimal routes with equal selection metric.
12:    if $rData.ACK = 1$ and $rData.ForagersIn > 0$ then
13:        if $rData.Matric = next\_best\_metric$ then
14:            add $rData$ to $R$
15:            $next\_best\_metric \leftarrow rData.Matric$
16:        else if $rData.Matric < next\_best\_metric$ then
17:            empty $R$
18:            add $rData$ to $R$
19:            $next\_best\_metric \leftarrow rData.Matric$
20:        end if
21:    end if
22: end for
23: if SIZEOF($R$) $> 0$ then
24:    $opRoute \leftarrow$ RANDOM($R$)
25:    $opRoute.ForagersIn \leftarrow opRoute.ForagersIn - 1$
26:    $opRoute.ForagersOut \leftarrow opRoute.ForagersOut + 1$
27:    return $opRoute.PID$
28: end if
29: return NO_PID
30:                          ▷ Will queue data packet and trigger new scouting process.
```

neighbouring node, such as in DSDV.

Since BeeIP is designed to evaluate routing based on “path” level instead of “link” level, link breakage within a path is detected when no foragers return back to the source node within a period of time. In such a case, the source node sets the path’s foraging capacity to 0, and marks the path as unacknowledged. The first ensures that no future foragers will be given the identification number of a broken path, whereas the latter allows the path to become available again, if a forager eventually comes back. Furthermore, in order to get rid of very old unacknowledged paths, a timer is used for housekeeping. This simple mechanism ensures that control overhead remains low because no special messages need to be exchanged just to confirm nodes’ existence. In terms of the intermediate and the destination nodes, a timer pruning is also triggered and unused routing information is removed. Two working examples of packet data flow are included in Appendix D. In figure D.1 a successful scouting as well as the transmission of artificial foragers is shown, whereas figure D.2 illustrates the detection of a broken path and the initialization of a new scouting.

4.9 Chapter summary and conclusions

In this chapter, BeeIP has been presented. BeeIP is a reactive routing protocol for mobile ad hoc networks which focuses on the constant monitoring of available routing paths and the evaluation of their performance. Its design consists of three components (i.e., Entrance, BeeIPHive and BeeIPQueue) which are collaborating in order to map several behavioural features found in the world of real honeybees.

Similarly to nature, new routing paths are discovered by initializing an artificial scouting. Scouts are sent either at the beginning of each communication session, or whenever the source has no more routing information to a destination. Moreover, routing maintenance runs for the entire session. Its aim is to alter the foraging capacity of each path, by evaluating it based on recent knowledge regarding its quality. The foraging capacity is defined as the number of future foragers that are allowed to be recruited and use a particular path. These numbers as well as the numbers of foragers who have been already sent out, control the population of the artificial hive.

Monitoring a path is also a technique inspired by the foraging behaviour of real honeybees. Both scout and forager packets that return back to the source collect low-level information from the nodes that they visit. Then, the information is used in order to measure the quality of the intermediate links and, in turn, the quality of the overall path at the source node. The latter affects the foraging capacity as well as the potential of the path in being selected in the future.

Finally, detection failure in BeeIP is a matter of monitoring the flow of incoming foragers at the source of a communication session. When a link break occurs, the path becomes unusable and therefore foragers cannot return back to their artificial hive. Timers are used to detect these events and the path is automatically marked as unacknowledged. Unacknowledged paths are pruned from all nodes.

This chapter presented a number of novel features regarding the designing of bee-inspired routing protocols. The proposed model of monitoring and evaluating the available paths combines a number of important network factors. This results in a list of routing solutions, each one having a different foraging capacity. This not only allows the protocol to select the optimal path from a list of potential good solutions, but also offers the ability to utilize the available options in a multi-path fashion. The optimal path is selected by a separate selection metric, which represents the artificial hive's demand towards a certain commodity (i.e., find the fastest path in terms of delays). Additionally, BeeIP is able to use previously collected knowledge in order to detect changes to the performance of the paths. This is a new advantageous feature in terms of adaptability, as it allows the protocol to complete recruitment cycles by considering up-to-date information.

The next couple of chapters are dedicated to the evaluation of BeeIP. Chapter 5 presents the detailed methodology which is used in order to run the comparison experiments correctly and get valuable results. Chapter 6 is used to present the results along with a discussion of their outcome.

Chapter 5

Methodology

5.1 Introduction

In order to evaluate the proposed network protocol's design which has been presented in chapter 4, scientific experiments need to be conducted. In this chapter, the methodology for these experiments is given. BeeIP is compared to four state-of-the-art routing protocol extensively used in the literature, each one being considered as a representative of a routing approach. Namely, AODV (Perkins *et al.*, 1999), DSDV (Perins & Bhagwat, 1994), DSR (Johnson & Maltz, 1996), and the multi-path extension to AODV, AOMDV (Marina & Das, 2001). The reason why these four protocols were chosen for the comparison is discussed in section 5.4.1. The results of the experiments presented in this thesis are obtained through the use of a wireless ad hoc network simulator (discussed in 5.2). Five different sets of simulation scenarios are designed to highlight several important aspects of the protocols under a variety of network conditions. These scenarios are presented in section 5.5.

5.2 Network simulation

Simulation is an important part of network research, not only for the study of communication protocols, but also for the study of any system that is either very costly to build, or strict in terms of flexibility. Network simulators are special software designed to provide network researchers with a virtual and highly configurable environment, where they can build and test any kind of network without worrying about repeatability (Martinez *et al.*, 2011; Köksal, 2008; Duflos *et al.*, 2006). Hence, network researchers can concentrate on their study and test their developments easily. A typical network simulator provides models, not only for well-known physical devices such as routers, switches, nodes and access points, but also for communication protocols which offer realistic representations of wireless or wired communication across the virtual network.

For a typical network simulator to operate productively, a virtual network has to be created by choosing carefully the underlying infrastructure and network elements. That is the number of nodes, communication links, bridging devices, stack of network protocols, etc. The choices depend on those particular characteristics and network conditions a researcher wants to simulate. Next, directly related to the aims and objectives of an experiment, network scenario or scenarios have to be designed. A simulation scenario sets the rules and boundaries within which the experiment lies, and therefore allows a researcher to test and monitor the new development. Furthermore, when the simulation finishes, the results are obtained by looking at the outcomes carefully and selecting the statistics that are required in order to make a valuable conclusion about the network under test. In most cases, this step also involves the visualization of the previously executed simulation.

Network simulators try to model the real world networks. They are expected to be as realistic as possible, independent of the area of the network or the number of network elements being used. They need to provide a good balance between features and different technologies, be efficient, extensible and easy to use. In order to fulfil those requirements and achieve accurate results, the simulated networks need to be well and carefully modelled. Only then the network simulation can be close enough to reality and allow the researcher to get a meaningful insight into the results, and understand how changes affect them.

5.2.1 A brief survey of network simulation software

There are several network simulation software found in the literature (Di Caro, 2003), which mainly belong to one of the following categories; of discrete-event and trace-driven simulations. The trace-driven simulation takes as input a time-ordered record of events (a trace) from a real system and simulates the scenario in its environment. This makes it easy to validate the results and reduces the factor of randomness, as the implementation of models tend to be highly detailed to match the real system. However, the need for detailed simulation of the system increases the complexity of the simulator. Another disadvantage is the lack of representativeness. That is, traces of one system may not correspond to others, which affects the evaluation in general. A trace-driven simulator has a big cost in terms of repeating an experiment after making changes to the model. In such a case, a new trace is required from the real system, increasing the time to conduct an experiment. A discrete-event simulator on the other hand, changes its state variables in correspondence to events that are happening at discrete points in time. These events occur as a consequence of activities and delays. During a simulation, active elements may compete for system resources, possibly waiting in queues in order to get access to an available resource. Most of the networks simulators that are used in the area of wireless ad hoc networks belong to the second category, the discrete-event simulators. Examples of such simulators are the OPNET (OPNET Technologies, Inc., 2012), GloMoSim/QualNet (GloMoSim, 2012; QualNet, 2012), OMNeT++ (OMNeT++ Community, 2012), and ns-2 (ns-2, 2012).

OPNET (Optimized Network Engineering Tools) Modeler (OPNET Technologies, Inc., 2012) simulator is a commercial tool from OPNET Technologies Inc. for modelling and simulating communication networks, network devices and protocols. Due to its long development, OPNET is considered a mature, state-of-the-art software in network simulation and is heavily used in the industry. Another factor of its success is the well designed graphical user interface. Although initially intended to aid companies in order to diagnose and re-organize their networks, OPNET is used in researching new algorithms. One of its strengths is its ability to manage networks of large sizes by allowing most of the deployment to be made through a hierarchical graphical user interface. An OPNET node contains a set of transmission and reception modules, representing a protocol layer or physical resource, to ensure its connection to communication link. Interactions between modules are handled by exchanging messages. The simulator's users are able to configure the applications which are installed on a node, and set nodes and links to fail or recover during simulation at specified times. In general, it is a powerful and user friendly network simulator with a large library of models and a good documentation. However, as in most proprietary software, OPNET does not support external tools and its complexity is such that makes a specific component hard to develop (Duflos *et al.*, 2006).

GloMoSim (GloMoSim, 2012) is a scalable simulation library designed at UCLA

Computing Laboratory to support studies of large-scale network models (up to millions of nodes) using parallel and/or distributed execution on a diverse set of parallel computers of both distributed and shared memory. It is a very powerful simulation tool, built according to the OSI layered approach for both wired and wireless networks. The library includes a variety of network protocols for all network layers, from FTP, HTTP and CBR (application layer) to AODV, DSR, DSDV (network layer) to CSMA and IEEE 802.11 (MAC layer). However, GloMoSim is not free to the public and has a very poor documentation (no user manual is available). QualNet (QualNet, 2012) on the other hand, is a commercial product from Scalable Network Technologies which is derived from GloMoSim. QualNet has its own additional features such as an expanded library of protocols and tools to realistically design 3D environments.

OMNeT++ (Objective Modular Network Testbed in C++) (OMNeT++ Community, 2012) is a well-designed discrete event simulation environment. The principle author is Andras Varga from Technical University of Budapest, and there are some occasional contributors. OMNeT++ is an open-source project, available to the public. It is a discrete-event simulator with a modular architecture, where in fact the basic entity is a module. Modules can be atomic, which captures an actual behaviour, or they can be composed of sub-modules. Each module can communicate with others by sending and receiving messages (events) through its gates which are linked via connections. Additionally, a module is able to generate, read or react to messages. In terms of disadvantages, OMNeT++'s limited number of out-of-the-box protocols, still-evolving documentation, poor analysis of typical performance measures (Begg *et al.*, 2006), and incomplete mobility extensions (Hogie *et al.*, 2005), decelerates the simulator's usability.

Another very well-known and mature discrete-event simulator is ns-2 (ns-2, 2012). The Network Simulator (ns) began as a variant of the REAL network simulator in 1989. In 1995, its development was supported by DARPA through the Virtual InterNetwork Testbed (VINT) project. Currently the development is carried out by Information Sciences Institute in California and is supported through DARPA and NSF (ns-2, 2012). It follows a modular architecture and is built according to the OSI layered model, similarly to GloMoSim. Its engine code is written in C++, whereas the scenario scripts are written in OTcl (object-oriented version of Tcl). It is well suited for both wired and wireless networks (ad hoc, local and satellite), and for simulations of queuing and routing algorithms, transport protocols, congestion control, and multicast related work. It comes fully equipped with protocols, models, algorithms and accessory tools, and it is an open source project. Due to its open source characteristic, rich documentation and support from its community, ns-2 is the most popular simulator used in the research field of mobile ad hoc networks. Thus, in terms of scientific acceptance, number of tools/modules and cost, ns-2 is considered ideal choice. However, ns-2 is rather complex software and debugging is very hard due to its dual C++/OTcl nature. Also, ns-2 has a very simplistic graphical user interface and rather limited tools in terms of visualization. Finally, once a simulation is finished in ns-2, the results need to be extracted from rather large trace files, which contain the events that took place during the duration of the simulation. This assumes that the ns-2 user knows how to parse and analyse the resulting trace files, an operation which usually involves the ability to write using scripting languages and tools, such as awk.

After reviewing the available tools, ns-2 has been chosen as the network simulator to use during the research work presented in this thesis. ns-2 is the most frequently used simulator in the literature, and a mature as well as respected software in the area. It meets all of the requirements for developing a new routing protocol comparing with others. In terms of designing, ns-2 comes with a rich variety of built-in models, not only for network elements such as nodes and links, but also for protocols of all relevant OSI layers. It has an excellent level of acceptance amongst the scientific community, and thus, the source code of the protocols shipped with the simulator's package can be considered accurate and error free. It is open source and has a modular software design, which allows cross-layering to be applied (see below in section 5.2.3 for examples). Furthermore, it has good documentation and a very active community that provides various external tools and reusable scripts. In terms of comparisons, ns-2 has been used as the test bed simulation for a number of network protocol comparisons such as Tuteja et al. (2010) as well as Sharman and Bhatia (2011).

5.2.2 Overview of mobile networks in ns-2

ns-2 (Issariyakul & Hossain, 2009) is a discrete-event network simulator which targets research regarding transport, routing, and multicast protocols for both wired and wireless (local and satellite) environments. When designing a simulation scenario, a researcher has to consider modelling the physical environment, as well as the characteristics of the required communication. The terrain size, the number of nodes and their mobility (in case of mobile networks), need to be set up. In terms of the network characteristics, ns-2 is shipped with a variety of protocols and allows any possible microscopic details (i.e., the OSI chain of protocols inside the node) to be modelled.

The modelling of mobile networks in ns-2 consists of several components, which simulate important aspects of wireless networking such as signal propagation, node movement and energy consumption. Mobile nodes are a special type of network element which have the ability to receive and transmit packets over a communication channel, and change position periodically according to boundaries defined by the scenario's terrain size. The inner architecture of a mobile node involves a link and a mac layer module that simulate the data link layer and the MAC layer protocol respectively, a network interface module to simulate the physical layer protocol, and a radio propagation as well as an antenna module to simulate the wireless adaptor device. In terms of energy consumption, the model is independently built and linked to the mobile node as one of its attributes. The initial value, which is the level of energy the node has at the beginning of the simulation, is set by the user. Additionally, the energy usage for every packet transmission or reception, as well as the node's idle and sleep states can also be set.

A routing protocol module is built and linked to the mobile node. Routing agents, as they are called in ns-2 terminology, are responsible for providing both routing and packet forwarding at the network layer of the mobile node's architecture. There is a number of well-known protocols for mobile nodes available with ns-2, such as AODV, DSR, DSDV, TORA (Park & Corson, 2001), PUMA (Vaishampayan & Garcia-Luna-Aceves, 2004) and others. Their source code has been thoroughly revised making sure that they are free of implementation errors.

For the transport layer, both TCP and UDP protocols can be modelled. In terms of TCP, two special agents can be linked to a mobile node in order to make it act either as the source of a communication session (TCP source), or the destination (TCP sink). Each agent can be separately configured to match the appropriate traffic scenario. For example, the ns-2 user can set the TCP segment's size to examine how the new development handles segmentation, or what is the effect of the packet size to the routing provided by a new routing protocol.

The application layer is modelled by the use of several known traffic generators, examples of which are the FTP, ping, telnet and CBR. The latter, combined with the UDP transport protocol, is frequently used in research because it is able to constantly transmit data at different predefined sending rates and packet sizes. Another very common combination is FTP using TCP at the transport layer.

The transmission of a packet in ns-2 is simulated by the radio propagation models. These models are responsible for predicting the received signal power of each packet. Accepting and dropping packets is done according to a receiving threshold found at the physical layer of each mobile node. If a received packet's signal power is below the receiving threshold, the packet is marked as error and is dropped by the MAC layer. ns-2 supports three propagation models. Namely, the Free Space model, the Two-Ray Ground Reflection model, and the Shadowing model (Eenennaam, 2009). The Free Space model assumes the ideal propagation condition, where there is only one clear line-of-sight path between the transmitter and receiver. It basically represents the communication range as a circle around the transmitter. If a receiver is within the circle, it receives all packets. Otherwise, it loses all packets. On the other hand, the Two-Ray Ground Reflection model considers both the direct path and a ground reflection path. The model is able to give a more accurate prediction at a long distance than the Free Space model. It is designed to show a faster power loss than the first propagation model, as the distance increases. Furthermore, the Two-Ray Ground Reflection model works similar to the Free Space for short distances. In order to achieve the dual behaviour, the Two-Raw Ground Reflection uses a cross-over distance threshold. Both Free Space and Two-Ray Ground Reflection models are deterministic and assume that the communication range is an ideal circle shape phenomenon. In reality though, due to the signal fluctuation because of objects obstructing the propagation path and multi-path propagation effects (fading), the calculation of the received signal power is a complex task. The Shadowing model tries to tackle this problem by assuming that the average received signal power decreases logarithmically with distance. In contrast to the two deterministic models discussed before, this model uses a path loss exponent to reflect the variation of the received power at a certain distance. This path loss exponent is empirically determined by field measurement. The Shadowing model extends the ideal circular propagation to a richer statistic model, where nodes can probabilistically communicate with each other, when they are near the edge of the communication range (Eltahir, 2007).

After a ns-2 simulation is finished, trace files are created. Depending on both the configuration of the simulation, and the implementation of the protocols, the trace files contain time-ordered information about simulation events, such as packet transmissions, mobile node position change and energy consumption. These results can be either visualized with an external software tool, or analysed in order to make observations and conclusion statements about the new development.

5.2.3 Cross-layering in ns-2

Cross-layering can be achieved in ns-2. This is because the simulator is open-source, and thus, full access to the source code is given, and also due to its available documentation regarding its design (Issariyakul & Hossain, 2009). Its modular architecture allows new agents to connect, and, providing the correct linkage is available, they can communicate with others in a straight-forward manner, no matter which layer they belong to.

However, implementation of cross-layering capabilities requires significant modifications to the ns-2 source code. According to Shannon and the "40-20-40" rule (Shannon, 1998), when creating a new development, 40 percent of the time and effort is recommended to be spent on defining a problem, designing a corresponding model, and devising a set of experiments to be performed. Another 20 percent should be used to program the conceptual elements obtained during the first step, and the remaining 40 percent should be utilized in verifying and validating the model, experimenting with designed inputs, and analysing the results. Although this approach is in no way a strict one, it illustrates the significant amount of time and effort being spent on the implementation of a new model. Hence, it gives an idea of the important role in terms of programming flexibility the network simulator plays.

This led the community to seek solutions, and propose ns-2 extensions to handle cross-layering with no significant cost. All these extensions are great enhancements for the network simulator and ease the implementation of new protocols. One such example is MIRACLE, the Multi-Interface Cross-Layer Extension (Baldo *et al.*, 2007) of ns-2. Miracle enhances ns-2 by providing an efficient and embedded engine for handling cross-layer messages and, at the same time, enabling the coexistence of multiple modules within each layer of the protocol stack. In addition, MIRACLE supports multiple transmissions rates, modulation and encoding schemes as defined in the IEEE 802.11b/g standards, and a realistic interface model which calculates the signal to interface and noise ratio (SINR) for each connection. Another example of ns-2 extension to handle cross-layering, is the Cross-Layer Framework for ns-2 (Varatharajan *et al.*, 2012). In this framework, a cross-layer manager is responsible for implementing inter-layer control and signalling. Also, adaptation modules are responsible for implementing cross-layer optimisation mechanisms between agents, by generating explicit congestion, loss, or delay messages.

Regardless of the fact that a cross-layer extension can reduce the implementation cost, in BeeIP no special cross-layering extension is used. There are two reasons behind this choice. First, half of the cross-layering requirements in BeeIP involve messages being exchanged between the application layer and the network layer. Namely, the current node speed, and the remaining energy level of the battery. In ns-2, both of them are considered attributes of models linked to the mobile node, and due to the simulator's modular architecture, they can be accessed as soon as there is access to the mobile node. For the next half, MAC queue size, transmission delay and signal power, cross-layering is achieved by writing special linkage code[1]. Secondly, the lack of cross-layering extension offered an in-depth understanding about the connectivity of the OSI layers and their representation in the network simulator. Studying well-used protocol models which have a great scientific acceptance by the community, has been another source of inspiration for the research work of BeeIP.

5.3 Towards BeeIP experiments and results

During the course of this research work, the design of the BeeIP routing protocol changed and improved many times, especially in order to support multi-path routing and to enhance the path monitoring functionality with a more accurate weighting system. The methodology of obtaining the results of the comparison between BeeIP and the four well-known routing protocols AODV, DSR and DSDV,

[1]A similar approach can be used in real applications. Linkage code or software can be used to fetch the appropriate values to the routing protocol by accessing the appropriate resources. For instance, the ACPI driver of the Linux kernel keeps track of the remaining battery capacity in special state files.

and the multi-path extension to AODV, AOMDV, is described as a series of steps of three major phases. Chronologically, the methodology phases were followed as they are presented in this section. Nevertheless, during the research period, each phase went through a number of iterations in order to reach the required level performance and results.

5.3.1 Phase 1: towards the design and implementation

The first step in the designing of the routing protocol was to conduct a systematic literature review in the two areas of interest. Namely, the wireless network communication and biologically inspired protocols, as well as the real honeybees and their behaviours in nature. One interesting observation was the comprehensive study of ants and ant-inspired protocols in the area of networking, and the lack of alternatives in terms of bee-inspired work. During the literature review, a large number of comparisons between protocols have been studied, giving an insight into the de facto comparison strategies and network simulations, which are used by the community. The ns-2 network simulator was chosen. BeeIP was then carefully designed to adopt real honeybee behaviours and, in particular, their foraging strategy as the mechanism of discovering and maintaining routing paths between of wireless sources and destinations in a mobile ad hoc network. Implementing BeeIP's design was achieved by extending the ns-2 source code. Like other routing agents such as AODV or DSDV, the BeeIP agent designed in such a way so it could be linked to mobile nodes of the simulator and allow them to operate according to its routing approach. Studying the ns-2 architecture and understanding how its models are connected to each other, cross-layering was achieved between different layer protocols. As a network layer protocol, BeeIP is able to retrieve values from both data link and physical layers. Furthermore, the first version of the BeeIP protocol (Giagkos & Wilson, 2010) used an empirically found weighting system (as discussed in section 4.6.2). However, the need to discover a more accurate set of weights led to the second phase described below.

5.3.2 Phase 2: towards the weighting system

In order to get an accurate weighting system, which can be used to express the relative importance of each low-level parameter of BeeIP path monitoring functionality in formula 4.1, a separate light weight routing protocol was written in ns-2, as part of this work. The limited routing protocol, named Weighting System protocol or simply WSYS, offers the mobile nodes very limited functionality. In fact, its only purpose is to produce data sets that describe packet transmission, under several conditions.

In more detail, WSYS agents depend on user configuration in order to build a routing table. However, once the routing tables are built, a WSYS agent knows how to generate and forward forager and ack_forager packets, i.e., packets that are able to carry real data as payload and monitor the performance of the paths they traverse, on the way back to the source. After each packet transmission and monitoring activity, the WSYS agent dumps the information to a log file. The idea of the WSYS protocol is to be utilized in such a way, that during the simulation, the five low-level parameters which are used to evaluate the performance of the links will get a variety of possible values, and affect the performance of the links in as many different ways as possible.

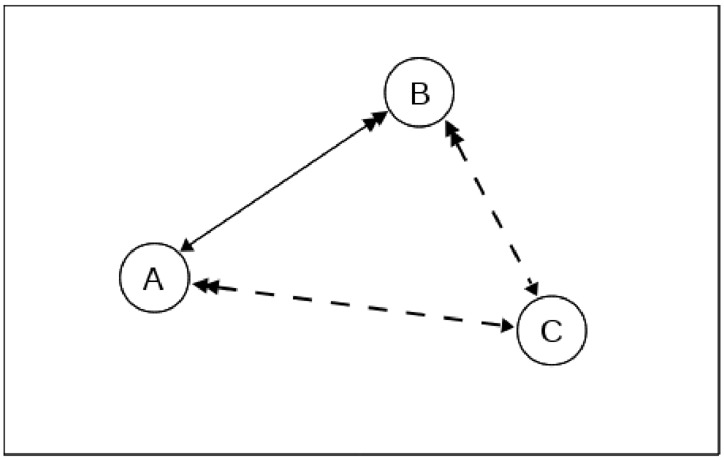


Figure 5.1: Topology and traffic model used by the limited routing protocols WSYS, in order to generate enough data and train the MLP towards the understanding of the relative important between the path monitoring low-level parameters in BeeIP. Here, the double arrows indicate the direction of data from the source to the destination. The solid line between node A and B means that the traffic between the two nodes is what is being traced by WSYS. Node C sends data to both A and B, interfering with their communication session, and producing difficulties, such as packet loss. The dashed line traffic is not traced by WSYS.

Therefore, the simulation scenario used for the data collection purpose has the following set-up. Three nodes, A, B, and C are allowed to move in a terrain of $300 \times 200\ m^2$ for 40 minutes. The small terrain size ensures that even though the transmission links will be affected by the distance of the nodes, they will not break, and the information regarding the links performance will still be written in the log file. Moreover, mobile nodes start with initial energy levels of 1800 Joules. Coupled with the long duration of the simulation, this setting ensures that the energy level parameter will take several values of the whole range, and will affect the transmissions. Similarly, nodes are constantly moving around in random speeds, from 1 m/s to 20 m/s, in order to allow the parameter of speed to affect the communication. In order to trigger the queue size parameter and the transmission delay, the routing tables provided by hand instructed the nodes to transmit data in pairs: A to B, C to A, and C to B, while the communication link between A and B is monitored. Nevertheless, the sending rate of which C sends packets is higher than the others. This ensures that both A and B will have their buffers busy, as C is constantly and rapidly transmitting packets to both of them, interfering with their own connection. The topology is illustrated in figure 5.1.

After the data collection was finished, each pattern in the data set had the form of five inputs and one output. The data was then fed to an artificial neural network and, in particular, a multilayer perceptron (MLP). The Fast Artificial Neural Network (FANN) (Nissen, 2003) library was used. Using the change of the mean squared error (MSE) technique to measure the relative importance of each input parameter (as discussed in section 4.6.2), an accurate set of weights was ready to be used in the BeeIP routing protocol. Usually, training an MLP repeats until stopping criteria are met. However, the purpose of this experiment was to find the relative importance of each parameter, where according to Sung the input that plays the most important role is the one with the highest change of MSE (Sung, 1998b). Therefore, training for each missing input was repeated in an exhaustive fashion in order to minimise the MSE scores as much as possible, and then to compare them with the MSE score achieved when all inputs were present. Table 4.6 of chapter 4.6.2 summarizes the MLP training parameters that lead to the weighting system presented in table 4.7. In terms of testing and validating the MLP training however, data sets of 5000 and 20000 patterns have also been used. Supplementary results can be found in Appendix E. The findings illustrate similar relevant importance of inputs which validates the success of the methodology adopted by Sung (1998b).

5.3.3 Phase 3: towards the comparison to other protocols

In the literature, the common methodology in order to compare a routing protocol to other representative protocols of the area, can be seen as a sequence of the following actions. Initially, once the implementation of the new protocol has reached the point where it can produce variable results, the protocols that will be used for the comparison need to be chosen. Each one ought to represent a specific approach so they can cover as much as possible of the de facto standards applied in the field. In BeeIP comparisons, these protocols are the AODV, DSR, DSDV and AOMDV. There are several reasons behind these choices, a discussion of which is given in 5.4.1.

Next, the performance evaluation of the protocols needs to be considered. In order to judge the merit of a routing protocol, one needs to think of the performance metrics with which to measure its suitability and performance. These metrics should be independent of any given routing protocol and designed to highlight ideal aspects of the network such as, the scalability, robustness, optimality, self-healing and self-configuration. In Corson and Macker (1993), the MANET working group lists several protocol evaluation and performance issues, and a number of performance metrics that researchers need to consider in order to test their protocols. Therefore, to compare BeeIP to other protocols a rich selection of performance metrics is used, to cover as much as possible in a both qualitative and quantitative approach. The performance metrics are presented in section 5.4.4.

Next in the sequence is the design of the experiments. Coupled with the performance metrics discussed above, the experiments need to be organized in such a way, that they trigger a variety of network conditions. This tests the performance under a rich number of circumstances, and offers a complete picture, in order to make valuable observations. Unfortunately, there is no clear and well-documented benchmark approach in designing experiments for routing protocol evaluation and comparisons. Very recent papers, such as Friginal et al. (2011) and Maqbool et al. (2011), are proposing ways of comparing protocols, but still need to be well-standardised. For the time being, many researchers have focused on evaluating the performance of routing protocols through generic measures like delay, routing overhead and packet delivery. Notwithstanding this common practice, when considering ns-2 an approach in comparing routing protocols was proposed by Broch J. et al. (1998b), from the Rice Monarch project[2] (Monarch Project, 2012). In the context of BeeIP, a similar approach to these common practices has been adopted. A particular interest is given in covering as many network conditions as possible to highlight not only the strengths but the weaknesses of the routing protocol, compared to others. An explanation of the BeeIP experiments is presented in section 5.5.

Coupled with the above, the ns-2 network simulator set-up plays an important role in producing efficient results. Again, common practices of configuring the simulations are followed in BeeIP, the details of which are given in section 5.4.3. Depending on the corresponding experiment, some configuration details change. However the base set-up in terms of modelling the physical and data link layers are kept constant. To conclude the methodology towards the BeeIP comparison to other protocols, the trace files produced by the simulations were analysed by the use of awk scripts, the output of which has been used to understand the observations and understand the merits of the protocol and illustrate them by drawing the charts, presented in chapter 6.

5.4 General design of experiments

In this section, a detailed discussion of the decisions made in order to obtain the comparison results is given.

5.4.1 Protocols used for comparison

Testing the performance of BeeIP is done by comparing it to four existing state-of-the-art routing protocols. Each protocol has its own characteristics and applies different routing strategies. What follows is a discussion of the reasons these four protocols were chosen as BeeIP's comparison counterparts.

AODV is a reactive routing protocol which is being heavily studied in the literature. AODV's performance is generally considered to set the benchmark standards in evaluating routing protocols for MANETs, as the protocol participates in most of the existing work in this particular research area. This is also due

[2]The Rice Monarch Project has made substantial extensions to the ns-2 network simulator, that enable it to accurately simulate mobile nodes connected by wireless network interfaces, including the ability to simulate multi-hop wireless ad hoc networks.

to the current experimental status of its draft by the IETF MANET group, which sets AODV under investigation for standardization. Being also a reactive protocol, BeeIP shares some common characteristics with AODV. In fact, the latter has been a source of inspiration in terms of the route request technique for finding new routes, as well as the gradually expanded broadcast of the scout packets, to manage the control overhead. Therefore, it has been decided that the results of their comparison will add valuable profit in understanding BeeIP.

DSR is also a reactive routing protocol for MANETs. It has gained considerable attention because of its ability to dramatically reduce control overhead, even under high rates of node mobility. DSR is similar to AODV in terms of route discovery, however it uses source routing in order to forward packets. Source routing reduces the computation complexity of deciding the next hop, as well as the amount of storage required to keep routing information, at each intermediate node. This has inspired BeeIP's packet forwarding approach (discussed in section 4.7), which divides the required next hop selection information to the intermediate nodes, reducing both computation and storage capacity. Additionally, DSR's draft has also an experimental status by the IETF MANET group and is used in the literature as a comparison counterpart for reactive routing protocols. Hence, the results of their comparison will enhance the understanding of BeeIP's performance.

DSDV on the other hand is a proactive routing protocol, considered to be the de facto standard in the area of proactive routing in MANETs. Although proactive routing is often seen as not the best approach for wireless ad hoc networks, especially under large number of nodes and high mobility rates (Hassan *et al.*, 2010), DSDV is heavily studied and used in protocol comparisons in the literature. Comparing BeeIP to the proactive DSDV, does not only comply to good common practice but also is expected to complete the picture of BeeIP performance differences under a variety of conditions.

AOMDV is a multi-path extension to the previously described AODV, proposed by Marina et al. (2001). Since BeeIP is a multi-path protocol, the comparison to an existing multi-path approach is expected to give extra value to the investigation. For the experiments presented in this thesis, the AOMDV implementation of ns-2 has been used.

A description of AODV, DSR, DSDV and AOMDV is found in 2.5.2.

5.4.2 Protocols not used for comparison

It is understood that due to the nature-inspired approach of this work, the comparison with another nature-inspired and, in particular, bee-inspired routing protocol such as BeeAdHoc and BeeSensor, would offer extra depth to the merits of BeeIP's design. However, due to technical difficulties and time constraints, these protocols are not included in this thesis and are instead seen as future work.

All four protocols which are used for the experiments are shipped with the official source code package of the network simulator and tested by the community. This ensures that they are error-free in terms of their implementation. Unfortunately, at the time of conducting this research, no ns-2 implementation of AntHocNet, BeeAdHoc or BeeSensor have been available to the public domain, after being approved by the protocol authors. Therefore, no proper scientific comparison would be possible and no genuine contribution would be added to the current work.

Implementing BeeIP's main competitor, BeeAdHoc, from scratch without a detailed documentation of its internal workings, constants, etc. as a guidance would never offer the fairness, accuracy and the sufficient level of confidence to conduct a direct comparison between the two. However, an indirect comparison and a discussion (theoretical and logical argumentation) about the benefits of the proposed design, is given below. It is worth keeping in mind that BeeAdHoc's published experimental results use traffic that is generated by constant bit rate over UDP and make use of the swarm packets only, without covering the TCP traffic and the use of piggybacked acknowledgement data packets to foragers. In Wedde et al. (2005b), BeeAdHoc is compared to the state-of-the-art AODV, DSR and DSDV (no Swarm Intelligent approaches are included, such as AntHocNet) using a similar configuration setup to the one used in this work and is presented in section 5.4.3.

BeeIP's mechanism to measure the quality of a path considers five different factors, whereas in BeeAdHoc the quality measurement is based only on the delay between the links and the remaining energy at the nodes in a path. In addition, BeeIP keeps recent history of previous quality findings. Combining those two features together, BeeIP is able to detect improvements or deteriorations over time and make changes to the foraging capacity (explained in section 4.2) accordingly. BeeAdHoc does not consider previous knowledge nor detects improvement or deterioration on the path's quality. Rather, each forager may dance and recruit fellow foragers according to the new quality finding only. Mimicking nature, BeeIP is designed to apply penalties to the foraging capacity which permits the protocol to react faster to quality changes. Additionally, its flexible design in terms of next hop selection metric, makes BeeIP easier to implement and fit the needs of the different deployments. In an indirect comparison, the results of Wedde et al. (2005a) and Wedde et al. (2005b) show that BeeAdHoc is able to send less control overhead than the others, but achieve less or equal packet delivery ratio to DSR, whereas BeeIP is found to achieve better packet delivery ratio than DSR.

BeeAdHoc uses source routing which is reported as a known disadvantage of the protocol that increases the control packet size as the length of the route gradually increases (Wedde *et al.*, 2005c). BeeIP on the other hand uses a different approach where the next hop at the intermediate nodes is decided based on the path's unique identification number and the direction (explained in section 4.7.1). Based on this logical argument, BeeAdHoc is considered less scalable in terms of number of nodes (longer paths in hops) as its foragers need to carry the whole path in their headers.

Another important design difference is that in BeeAdHoc each forager has a lifetime value. This value is considered when the protocol needs to send a matching forager to the packing floor in response to a request from a packer. The foragers whose lifetime has expired are not considered for matching (Farooq, 2009). Following this, in the forth chapter of Wedde et al. (2005c), Wang reports "*Another real disadvantage [of BeeAdHoc] is the higher memory use for storing every forager. Although they are really small it is more than storing every route only once*". Clearly, keeping foragers in memory is an expensive behaviour, especially when the devices in MANETs and WSNs come with limited resources. In BeeIP on the other hand, the foragers are destroyed when not used, deallocating their memory rather than keeping it; a source node keeps track of the number of foragers currently waiting (forgers in) and currently using the path (foragers out) for every stored entry of the routing repository. This design difference implies that BeeIP uses less memory than BeeAdHoc in terms of storing its control packets.

5.4.3 Base configuration and set-up

In terms of modelling the physical and data link layers of the mobile nodes, all ns-2 scenarios have been configured to match the benchmark scenarios presented in Broch et al. (1998b). In more detail, the Two-Way Ground Reflection propagation model is used, with the signal power of a transmission between two antennas to be attenuated as $1/r^2$ and $1/r^4$ (r being the distance between the antennas) for short and long distances respectively. The crossover point is typically around 1000 metres for outdoor low-gain antennas of 1.5 metres above the ground. This is the most common approach to model signal propagation in open space found in the literature. Furthermore, the IEEE 802.11 protocol is used, ensuring data transmission rates of 10Mbits/s, with an estimated transmission range of 300 metres.

At the MAC layer in particular, the IEEE 802.11 DCF (Mueller, 2007) protocol is used. This is the most popular MAC protocol for wireless networks. It consists of two algorithms, the contention based Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), and the 802.11 Request To-Send/Clear-To-Send (RTS/CTS). When combined, these algorithms allow single wireless channel to be used by multiple nodes without facing the hidden node problem (better described in chapter 2). Furthermore, the MAC interface queue is set to hold up to 50 packets.

At the transport layer the TCP protocol is used. Depending on the scenario, several TCP sources are configured to constantly transmit FTP data packets to TCP sinks over the wireless network. In order to simulate harsh networking conditions, packet sizes are randomly set to a range of 265K to 1025K. Initially intended for wired environments, TCP is the most common transport layer protocol used for data transmission. However, previous studies have shown a negative performance in wireless networks as discussed in section 2.3.4. Thus, CBR over UDP is frequently used when comparing routing protocols. In reality, the majority of the traffic is achieved by TCP. Friginal et al. (2011) mention the need of using real applications (VoIP, FTP or email, for instance) to develop realistic network configurations and increase the representativeness of results. Being a new routing protocol, BeeIP is designed to work using the current technologies and take advantage of them and, in particular, the acknowledgement mechanism of TCP communication sessions.

At the application layer, at the beginning of each simulation, the initial energy left in the node's battery is set to a random number from a range of 200 to 1500 Joules. These numbers are rather small and therefore all batteries are close to being flat. The reason behind this particular configuration is again to simulate a harsh condition, when after a long constant period, some nodes will shut-down causing transmission links to break, and in turn routing anomalies. In addition, FTP data are generated at the application layer, and sent over TCP as explained earlier.

Finally, nodes move according to the popular Random Waypoint (RWP) movement model (Camp *et al.*, 2002). In RWP, each node is moving along a line from one way point to the next, at a random speed. Also, way points are uniformly distributed over the given terrain area. Once a node reaches the target way point, it pauses for a certain time (pause time), after which it moves again towards the next way point at a different random speed. Speeds are selected from a range of 1 to 10 m/s, which is walking speed to a slow car moving in a city centre. RWP was chosen over other movement models, such as its variants Random Walk (Bai & Helmy, 2004), because of its simplicity and frequency of appearance in the literature, and specifically the benchmark paper by Broch J. et al.

5.4.4 Performance metrics

The different performance metrics for the comparison are listed and explained in this part of the chapter. Each one specializes in a different aspect of the performance, which offers a comprehensive understanding of the protocol strengths and weaknesses.

1. *Packet Delivery Ratio*
   This is the ratio between the successfully received data packets at the destination node, to the successfully sent data packets by the source node. This metric does not take into consideration the control packets, and achieving a high result is required by any routing protocol.

2. *Control Overhead*
   This is the number of routing-related packets that are sent in order to exchange routing information and maintain routing. For instance, these are the scout and ack_scout packets in BeeIP, and RREQ, RREP, RRERR and

RREP-ACK in AODV. The less control overhead required by a protocol, the better.

3. *Average End-To-End Delay*
   This is the difference between the time a packet is received by the destination node and the time it was originally sent by the source node. This includes all the intermediate delays a packet has to experience during the transmission, e.g., propagation delay, transmission delay, queuing delays, process delays, etc. A small average end-to-end delay means faster data transmission.

4. *Average Path Duration*
   This is the average time a path stays active and constantly used for data transmission, before it breaks. Routing protocols that manage to keep path durations high tend to be more robust and require less route discovery processes to be initiated, thus, less control overhead.

5. *Average Throughput*
   This is the average of the throughputs as calculated for each session between a TCP source and a TCP sink. The throughput is defined as the number of bits that have been delivered to a node for the duration of a session. The greater the average throughput, the better.

6. *Network Life*
   This metric is related to the energy-efficiency aspect of the performance. It is the average remaining battery capacity of the nodes after the simulation has finished. At the end of the simulations, the most energy-efficient protocol is the one with the highest remaining network life.

7. *Route Requests*
   This metric is used only for the three reactive counterpart protocols, BeeIP, AODV and AOMDV, which share similar route request mechanisms. It gives an idea of the effectiveness of the routing solutions that are found. It is the number of new scouting and route request processes during the simulation. The smaller the number, the more effective the solutions have been during the simulation, and more space has been available for real data transmission. This is because the number of route requests is directly related to the control overhead. Additionally, a small number of route requests in connection with

better throughput, indicates that multiple paths have been discovered during a single route request, and used to transmit data.

5.5 Design of simulation scenarios

In this section, a justification regarding the simulation scenarios used for the comparison between BeeIP and the other four routing protocols (AODV, DSR, DSDV and AOMDV) is included. There are five distinct sets of scenarios, which are carefully designed in order to cover as much as possible of the network conditions that occur in real networking. Namely, they cover alterations in node mobility (i.e., changing pause times and node speeds), communication (i.e., number of nodes participating in the topology, number of sources and destinations), as well as terrain alterations (i.e., changing of terrain size). Once run by the simulator, each scenario produces data which is checked against the performance metrics presented in the previous section.

5.5.1 Pause time scenarios

A pause time is the period for which a node stays in a fixed position, after which it moves again towards the next destination point at a random speed. Thus, the pause time affects the mobility of the nodes which, in turn, affects the network topology and the transmissions within. As the pause time increases, the mobility of the nodes decreases, making the topology less dynamic. Altering the pause time is a common practice when evaluating the performance of a routing protocol. The observations made from the obtained results offer an understanding about the protocol's robustness as well as self-healing and self-configuration. Since the pause time is one of the values which affect the topology, it plays a significant role in the data transmissions between the nodes and the stability of the communication.

In order to understand the degree to which the protocol's behaviour is affected, a wide range of pause time values need to be examined. Ideally, the range must cover both static, and highly dynamic networks. For this reason, BeeIP's pause time scenarios repeated with 0, 75, 150, 300 and 600 seconds. The latter is set to be the duration of the simulation. Coupled with the random speed selection of each node (1m/s to 10m/s), the progressively increased values cover the range of movements, starting from nodes with fixed positions (600 seconds) to nodes which never stop moving around the terrain (0 seconds). To make sure that there will be more than one path available between sources and destinations, the number of nodes is set to 100 and the terrain size to 3000 × 1000m^2. This ensures that multiple connection will exist due to the density of the topology.

5.5.2 Terrain size scenarios

Changing the terrain size, is another important scenario which is used to evaluate the protocol's behaviour. When increased, the average communication path length between sources and destination nodes is also increased, in terms of both hops and physical distance. Additionally, increasing the size of the terrain decreases the nodes' density in the topology. In consequence, the scenario is made more difficult. For instance, during a transmission session within a sparser network environments, fewer alternatives are generally available between sources and destinations, affecting dramatically the number of successful data deliveries. In some worse cases, sparse networks offer limited connectivity (increased packet loss) or no connectivity at all.

In BeeIP, four different terrain sizes are used with the same number of nodes operating within. Namely, 1800 × 600m^2, 2400 × 800m^2, 3000 × 1000m^2, and 3600 × 1200m^2, with 100 nodes exploring the areas. The scenarios are designed in this way, so the obtained results will highlight the scalability of the protocols and give an idea of how they perform in both dense and sparse network environments.

5.5.3 Network traffic load scenarios

Traffic between nodes is achieved by several FTP/TCP communication sessions, between sources and destinations. For this scenario, the number of active sources and data sessions is changed. Active sources are the nodes which have an active TCP source agent linked to them, and require data packets to be sent to a specific destination (to a TCP sink agent, linked to destination node). Furthermore, each active source may participate in more than one data session, or act as a destination (TCP sink) for one or more of the data session during the simulation. The number of data sessions also varies. Both these parameters constitute the traffic model of the simulation, and affect the performance in the two following ways. As the number of active sources increases, the traffic within the network topology also increases, keeping more nodes busy in packet forwarding. Also, when the number of data sessions is increased, then the active sources are expected to participate in more than one data session, causing possible bottle necks and congestion. Therefore, robustness as well as adaptability are examined by this scenario.

BeeIP's network traffic load scenarios involve altering the number of active sources and data sessions, starting from 10/15 (sources/connections) to 60/80, with intermediate values of 20/25, 30/45, 40/60, 50/80. These values are chosen to help build a solid understanding of how good or bad the protocol performs, compared to others, under stressful situations. Low (10-20), medium (30-40), and high (50-60) active FTP/TCP source nodes are randomly chosen between 100, in large enough area able to create a dense topology ($3000 \times 1000m^2$), ensuring that more than one path will be available between a source and a destination.

5.5.4 Mobile node speed scenarios

Another way of examining the behaviour of the protocols in terms of how easily they adapt to dynamic networks, is by making changes to the node mobility model. The obvious effect is that the higher the speed of the nodes, the higher their mobility. The latter leads to more frequent changes of the network topology, and thus, difficult circumstances to occur which require the ability to self-configure and adapt to changes. Similarly to the values of the previous scenarios, the node speed has to be widely chosen in order to represent the whole range, from low to high dynamic networks. Hence, for the BeeIP experiments, nodes are allowed to reach four maximum yet realistic speeds. Starting from maximum 5m/s for low mobility, to 10m/s for normal, 15m/s high, to 20m/s very high (running vehicle with 72 km per hour). Coupled with a fixed pausing time, in a terrain of 3000 $\times$ $1000m^2$, these values ensure a rich collection of network conditions where the mobility of the nodes dramatically affect the performance.

5.5.5 Number of mobile nodes scenarios

The scalability of a routing protocol is also examined by considering the number of nodes which participate in a network topology. The size of the network in terms of node, affects the performance in several ways. Initially, an increased number of nodes increases the number of alternatives between a source and the destination of a communication session. It also leads to longer paths, by increasing the number of hops a packet has to take towards its final receiver. Depending on the number of data sessions started, the generated load for a large number of nodes is more likely to be distributed across different parts of the network, leading to less congestion points and increased throughput. This is less likely to happen in a small network, where traffic from different session occupies the same number of nodes, leading to bottle necks and low packet delivery ratio. Finally, altering the number of nodes is directly related to the network density. A sparse topology causes packets to be dropped or missed by nodes at the edge of the sender's transmission range, whereas in dense topologies links are less likely to break for the opposite reason.

The number of mobile nodes scenarios in BeeIP are designed to examine the cases where 50, 100, 150 and 200 nodes are participating in the same wireless network. For each number of nodes, almost one third of the total population is used as active sources. By this, a proportional amount of traffic is generated for each case. Namely, 15 active sources for 50 nodes, 30 for 100, 50 for 150 and 60 for 200. The terrain size is kept constant, $3000 \times 1000m^2$, so that the obtained results will reveal how BeeIP behaves under different topology densities, and how it is compared to the other protocols.

5.6 Chapter summary

In this chapter, the methodology used for comparing BeeIP to the state-of-the-art AODV, DSR, DSDV and AOMDV routing protocols is presented. Using a network simulator not only speeds up the process of researching, but also provides the means to test and analytically evaluate the conceptual models under investigation. A brief summary of the most frequently used network simulators is given, where positive and negative characteristics of each existing software are discussed. A justification for choosing ns-2 in this research is also given, taking into consideration its scientific acceptance in the field, its rich variety of built-in protocol models, open-source availability, and good documentation and helpful community. An overview of both ns-2 and its strength in cross-layering, show the simulator's potential to produce valid and accurate results.

The steps that are taken towards the BeeIP design, implementation, and evaluation through an extensive set of comparisons, constitute the methodology of this research. Following the de facto standards, which are found in the related literature, the performance of BeeIP compared to the other protocols is evaluated against a variety of performance metrics. The five sets of experimental scenarios are presented and discussed, in order to provide a good understanding before the actual results are given, in chapter 6.

Chapter 6

Evaluation study

6.1 Introduction

This chapter presents the results of the extensive comparison between BeeIP and the four well-known routing protocols in the area: the three state-of-the-art AODV, DSR, and DSDV, as well as the multi-path version of AODV, AOMDV. In order to build a complete understanding of the strengths and weaknesses of the proposed protocol, a variety of network simulation experiments have been conducted. Namely, the protocols are evaluated using several performance metrics such as control overhead, packet delivery ratio and average and-to-end delay, under five distinct simulation scenarios. These scenarios are designed to explore the behaviour of the protocols and include changes in the pause times, terrain size, network traffic, speed of the nodes and number of nodes in the topology. The methodology which is used in order to design, execute and obtain the results is presented in chapter 5.

6.2 Experimental Results

The base network simulation set-up is given in 5.4.3 and is the same for all the simulation runs. However, depending on the scenario, several characteristics of the network change in order to match the required condition under which the protocols are tested. These changes are reported at the beginning of each study and are followed by a graphical representation of the results, as well as an evaluation discussion. A summary of all the experiment settings is also found in table 6.1.

Setting: \ *Scenario*:	**Pause times**	**Terrain sizes**	**Network traffic**	**Node speeds**	**No of nodes**
Sim duration:	600 s	600 s	600 s	600 s	600 s
No of nodes:	100	100	100	100	50, 100, 150, 200
Packet sizes:	265-1025 K	265-1025 K	265-1025 K	265-1025 K	265-1025 K
Starting energy:	200-1500 J	200-1500 J	200-1500 J	200-1500 J	200-1500 J
Terrain sizes:	3000×1000 m^2	1800×600, 2400×800, 3000×1000, 3600×1200 m^2	3000×1000 m^2	3000×1000 m^2	3000×1000 m^2
Pause times:	0, 75, 150, 300, 600 s	75 s	75 s	75 s	75 s
Node speeds:	1-10 m/s	1-10 m/s	1-10 m/s	5, 10, 15, 20 m/s	1-10 m/s
Sources (nodes):	40	40	10, 20, 30, 40, 50, 60	40	15 (50), 30 (100), 50 (150), 60 (200)

Table 6.1: Summary of all the experiment settings.

The experiments for each scenario are repeated 10 times, and the average results obtained from each experiment are illustrated in figures and reasoned according to the mechanisms applied by each protocol.

6.2.1 Studying pause times

The first set of experiments examines how the protocol behaves in terms of changing the pause times during the simulations. The pause time affects the mobility of the nodes which in turn affects the network topology. For these experiments, the pause times of 0s up to 600s are used, with intermediate values of 75s, 150s, and 300s. The simulations last for 600 seconds. A pause time of 0s means that the nodes are constantly moving. A pause time of 600s means that the nodes have fixed positions and the network topology is static. The number of nodes is set to 100, the terrain area is $3000 \times 1000m^2$, and each node speed is set randomly from 1 m/s to 10 m/s. In terms of data traffic, 40 nodes act as TCP sources, constantly transmitting data to TCP sinks, forming up to 60 data sessions in total. The results for each individual performance metric is shown below.

Packet delivery radio

Figure 6.1 shows the packet delivery ratio (%) as a function of the varying pause times. The error bars show the standard error from the mean. The first observation is that BeeIP shows the best packet delivery ratio for highly dynamic networks and is rather insensitive to the pause time variations, whereas AODV's performance is slightly decreased as the network loses its dynamic characteristic. Being able to find more paths than a single one, AOMDV achieves better packet delivery ratio than AODV. The result of this experiment agrees with the results obtained in Biradar et al. (2010), where AOMDV is shown to achieve better packet delivery ratio than the single-path AODV. On the other hand, due to its aggressive caching, DSR is able to perform better under less stressful situations, i.e., when the topology is less dynamic (pause time increases). For small pause times, its lack of removing inactive and out-of-date entries from the routing caches leads to packets being sent to broken links, causing increased packet loss. Nevertheless, all the reactive protocols are much more effective that DSDV, which due to its proactive nature performs poorly, especially when the pause times are small.

In order to understand the statistical significance of the difference of their

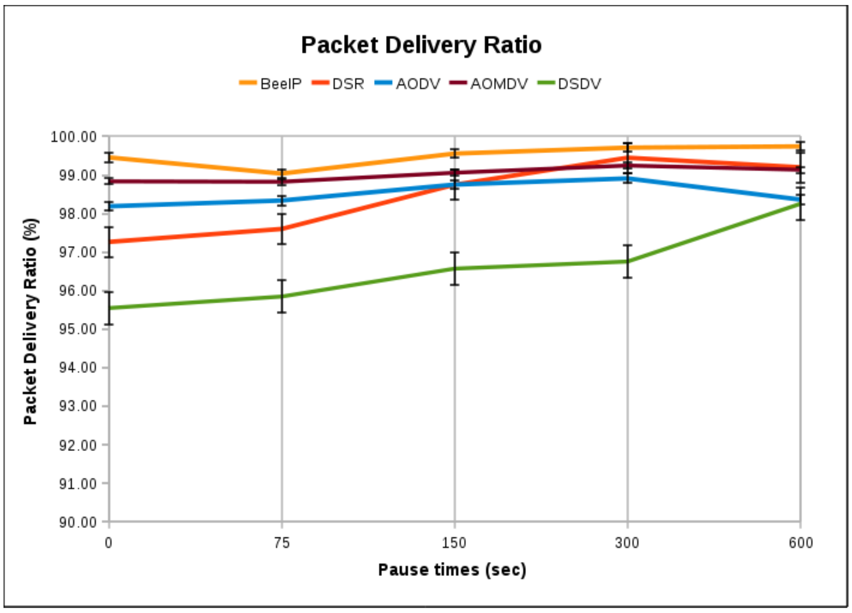


Figure 6.1: Packet delivery ratio as a function of pause times. Here, BeeIP is shown to outperform all other protocols, and manage to keep a balanced PDR for all pause times. Due to the stale cache entries problem, DSR is performing worst when the network is highly dynamic. Rather expected, DSDV on the other hand has a low PDR due to its proactive nature.

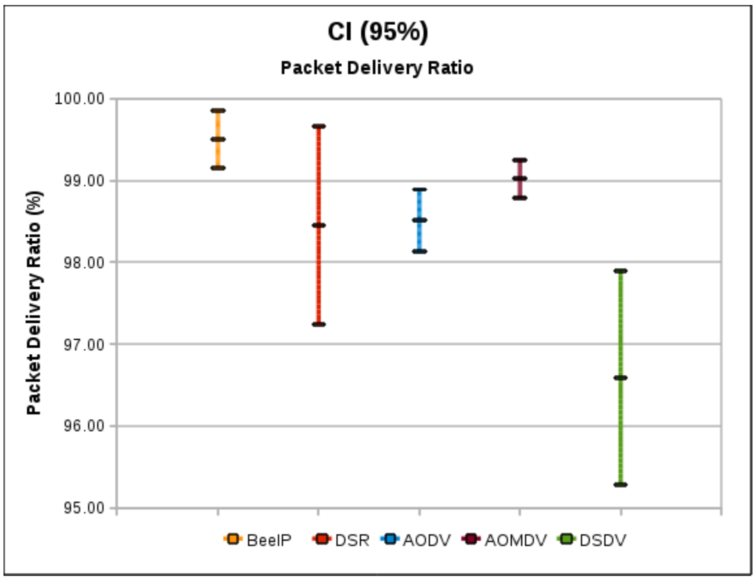


Figure 6.2: 95th confidence interval of PDR as a function of pause times. BeeIP is able to achieve a significantly better result than AODV and DSDV for all pause times.

performance, the 95th confidence interval has been measured and shown in figure 6.2. The 95% confidence level shows that for the 95% of the times, the protocol's average packet delivery ratio will lie within the upper and lower limits. It is clearly proven that under the same experimental conditions, BeeIP's performance in terms of packet delivery ratio is significantly better than both AODV and DSDV, while its 95% confidence upper limit is greater than all other protocols.

Control Overhead

Looking at the control overhead caused by the five routing protocols in figure 6.3, it is understood that as nodes lose their mobility, the number of control packets required to maintain routing is also reduced. The lack of a local link repair mechanism in DSR allows it to produce a significantly less control packets during the simulations. This is reinforced by figure 6.4 where the 95th confidence interval is also shown.

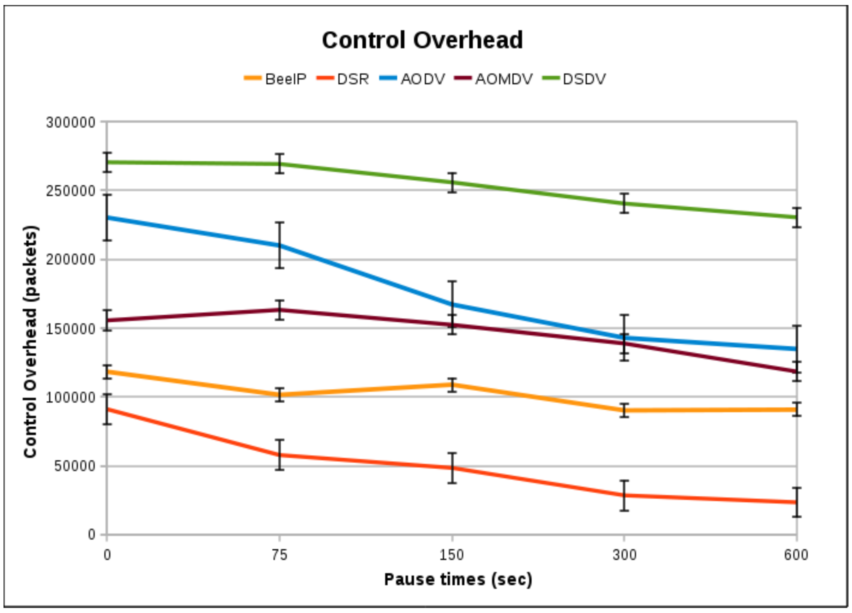


Figure 6.3: Control overhead as a function of pause times. Here, BeeIP is shown to outperform AODV, AOMDV and DSDV, and produce a balanced control overhead for a variety of pause times. Its ability to use multiple paths allows the protocol to achieve better results for constantly moving nodes.

Between BeeIP and AODV there are two noticeable observations. BeeIP significantly outperforms AODV as well as manages to keep a balanced behaviour for broadcasting control packets, whereas AODV's route requests and replies are gradually decreased as nodes tend to become immobilised. This is an indication of the pay-off the multi-path feature BeeIP is giving to the performance. A single scouting process is able to find more than one routing path between a TCP source and a TCP sink. Although this increases the number of ack_scouts that return to the source, in the long run it allows the source to work with a number of alternatives, and switch between them in case that one of the paths is broken. Similar results are shown for AOMDV, which is also able to find alternative paths using a single route request process. Being able to find alternative paths, AOMDV starts less route discovery processes and thus incurs less control overhead than AODV. However, the mechanism of how the alternative paths are used is more less the same for the two protocols. That is, an alternative path is used only when the previous working path is broken. In BeeIP on the other hand, alternative paths are allowed to be used concurrently (depending on the number of available foragers), allowing the protocol to monitor and evaluate them while they are all in use. DSR is shown to outperform all other protocols, due to its lack of local repair mechanism, which reduces the control overhead dramatically. The significance of the difference between BeeIP, AODV, and DSDV is show in figure 6.4.

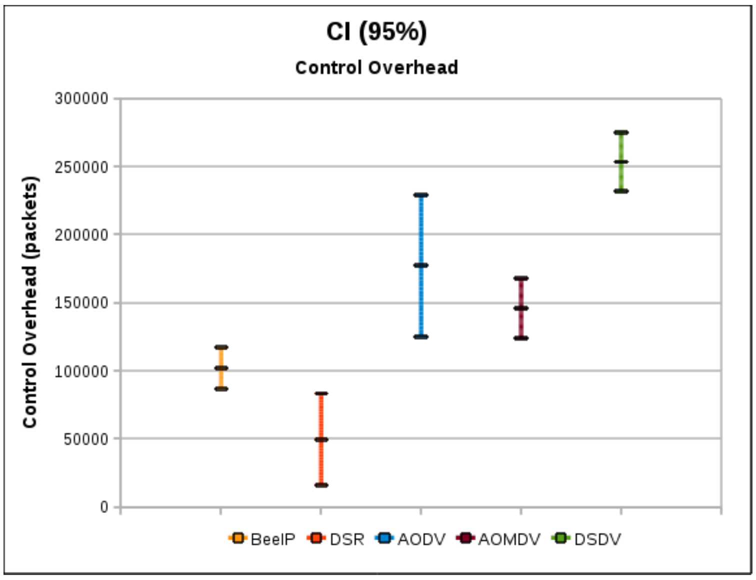


Figure 6.4: 95th confidence interval of control overhead as a function of pause times. The significant improvement in control overhead is shown, as BeeIP's upper limit is lower than AODV, AOMDV, and DSDV's which produce more routing packets in order to discover and maintain routing.

Finally, the DSDV protocol follows expected proactive behaviour. Exchanging route messages periodically increases the control overhead. It is also interesting to notice that BeeIP's, AOMDV and DSDV's standard error bars are quite short, which shows that the amount of control overhead produced by them is constant no matter the variation of the pause time.

Average end-to-end delay

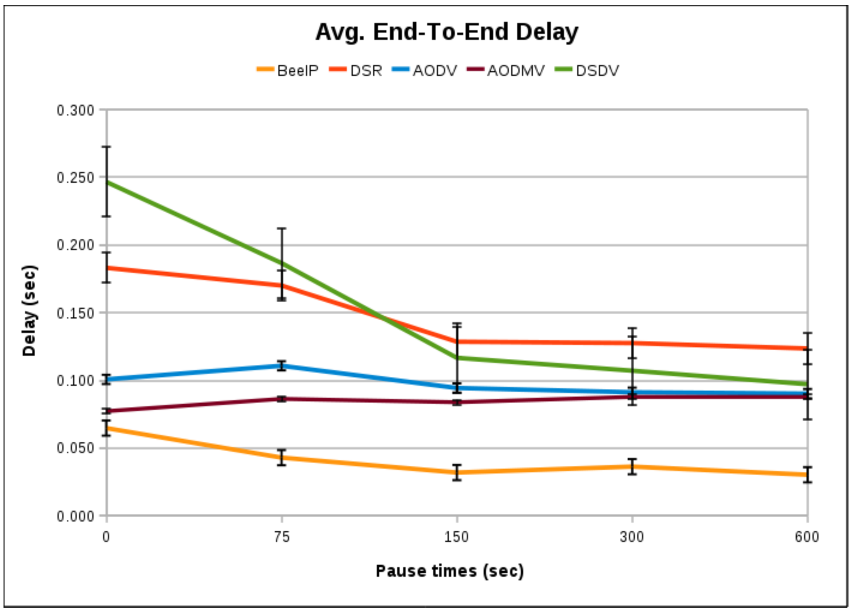


Figure 6.5: The average end-to-end delay as a function of pause times. BeeIP is shown to outperform all other protocols, due to its minimalistic approach in packet switching between the intermediate nodes of a path. Additionally, using more than one paths concurrently allows the protocol to overcome traffic congestions and spread the packet load better throughout the topology, reducing the end-to-end delay.

The average end-to-end delay is presented in figures 6.5 and 6.6. Here, the strength of combining traditional routing tables with the minimalistic approach of source routing developed in BeeIP is shown. BeeIP is able to outperform all the other protocols under the various different pause times. Additionally, its small error from the mean is another indication that the protocol is not sensitive to the increase of pause times. This is due to the fact that BeeIP's buffering is always kept to a minimum, which is achieved for two reasons. Firstly, being multi-path the protocol uses more than one path to transmit packets for each session. Rather than occupying the buffers of the same nodes, the load is spread over the topology. Secondly, BeeIP has balanced control overhead and fewer packets flood the network and the MAC interface queue of each of the nodes. Moreover, the performance of both DSR and DSDV is noticeably worse than BeeIP, AODV and AOMDV under the situations of small pause times. It is seen that up to 150s, their end-to-end delay is gradually decreased.

In figure 6.6, the statistical significance of BeeIP's improvement is shown. The results show that for the 95% of the time, the average end-to-end delay achieved by the protocol lies between the upper and lower limits, which are less than what is achieved by the other protocols. By looking at the length of the CI95 bars of the protocols, it is also noticeable that BeeIP, AODV, and AOMDV are less sensitive to the changes of pause times, compared to DSR and DSDV.

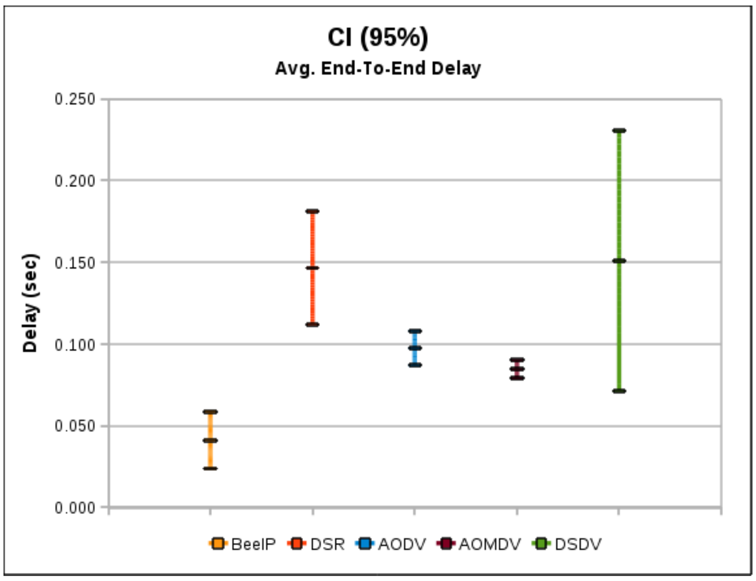


Figure 6.6: 95th confidence interval of the average end-to-end delay as a function of pause times. Here, BeeIP is shown to significantly outperform the other four protocols, in terms of end-to-end delay. This is due to its ability to use spread the traffic over multiple paths and overcome traffic congestions and bottlenecks.

Average path duration

The aggressive scouting behaviour of BeeIP, comes with a cost. Observing the experimental results in figure 6.7, the average path duration produced by BeeIP is rather ambiguous as the pause times vary, and is less than its three reactive counterparts. Unlike AODV and AOMDV, where a path's future is compromised only by the links which may be found to be broken by the local HELLO messages technique, in BeeIP a path can be dropped either because the foragers have not used it for some time, or it has stopped having sufficient foraging capacity, or both.

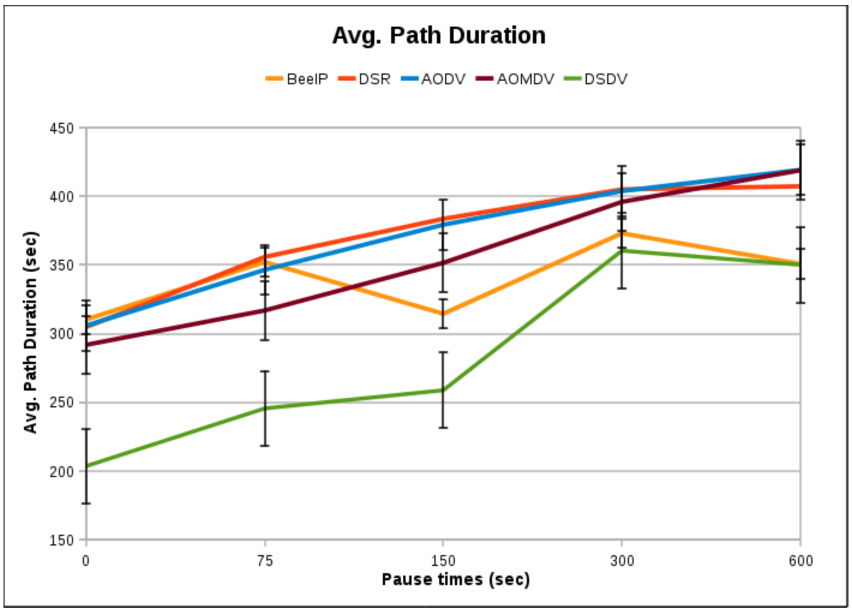


Figure 6.7: The average path duration as a function of pause times. Here, BeeIP is shown to have unbalanced and weak average path duration compared to AODV, AOMDV and DSR. This is due to the aggressive scouting of the protocol.

Although it does not make any significant difference as shown in figure 6.8, it is considered be to a feature of the protocol which contradicts to the behaviour of real honeybees. In nature, when the demand of food is high and the working paths are found to gradually deteriorate over time, honeybees initialize new scouting processes to discover new interesting sites. This is a wise behaviour for the real world, though in wireless networks the number of new paths to be discovered is limited, and there is always the possibility that a scouting process will end up with the same result.

An interesting observation regarding AODV and AOMDV, is that when the network is highly dynamic (0s to 150s pause time), AOMDV suffers more in terms of path duration. This is explained by considering the design of AOMDV and the way it handles multiple paths between sources and destinations. Although a

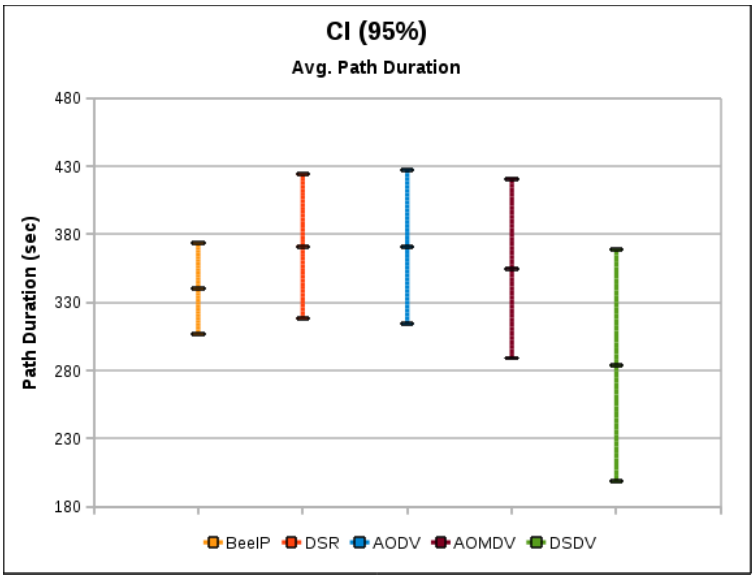


Figure 6.8: 95th confidence interval of the average path duration as a function of pause times. Here, no statistical significance can be seen regarding the improvement of BeeIP. However, the weakness of BeeIP for applying an aggressive foraging is highlighted, as although reactive in nature and multi-path, the average path duration is lower than the other reactive counterparts, namely AODV, AOMDV and DSR.

single route request may return information for multiple paths, the information is stored in the routing tables and is not used as long as the links which constitute the current path are not broken. Hence, due to high mobility, alternatives may be already broken when they are selected as the next available solutions. To reduce the magnitude of this problem, AOMDV is using time-out values and extra local repair HELLO messages as AODV does for the single-path maintenance (Marina & Das, 2006).

Although this technique slightly increases the control overhead of AOMDV, it reduces the need of extra route discovery processes, which will be shown later in this comparison. Moreover, as the network becomes static, both AOMDV and AODV achieve similar results. That happens because AOMDV is using its multi-path ability less frequently when no links break due to mobility.

Average throughput

For the average throughput (as seen in figure 6.9), BeeIP is a competent reactive counterpart despite its low average path duration, explained earlier. This is rea-

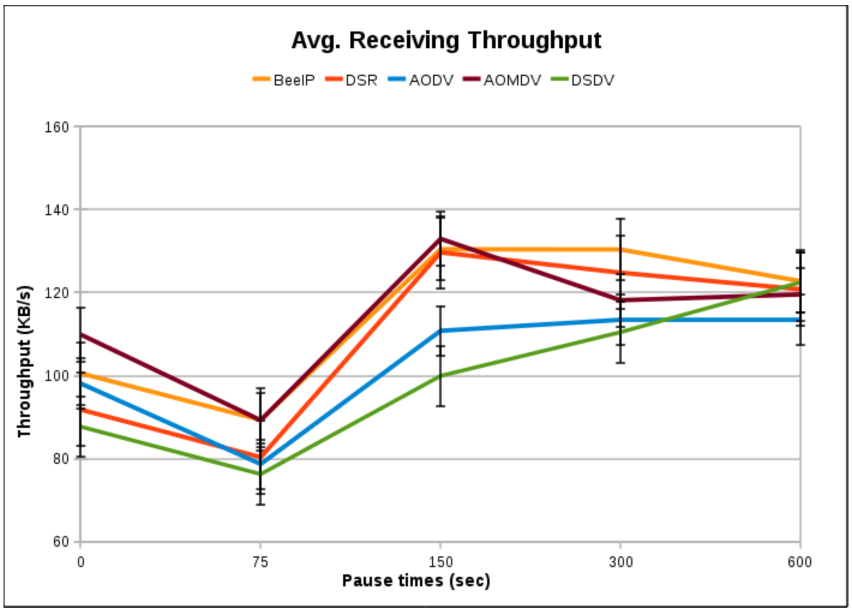


Figure 6.9: Average throughput as a function of different pause times. Here, BeeIP is shown to be a competent reactive counterpart, when compared to the multi-path AOMDV, as well as DSR which is generally strong in reducing the control overhead and increasing throughput.

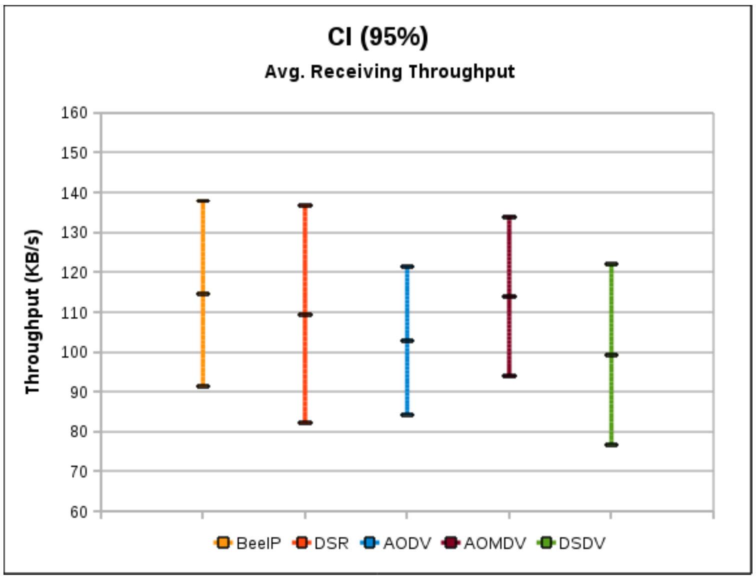


Figure 6.10: 95th confidence interval of the average throughput as a function of pause times. The average limits of BeeIP, AOMDV and DSR is another illustration of the similar throughputs achieved by BeeIP, AOMDV and DSR.

soned by considering again the multi-path ability of the protocol, which allows it to utilize several path alternatives that might be found by a single scouting process. Compared to AODV, the difference in average throughput is scaling up to double for pause times 150s to 600s, compared to more dynamic 0s and 75s. Not surprisingly, the less dynamic the network, the better the performance of DSDV, even when it sacrifices a significant amount of bandwidth and buffering capacity for the periodical control packet transmissions.

Considering the 95^{th} confidence interval in figure 6.10, no clear assumption can be made. However, looking at the average limits, the results enhance the previous discussion on the ability of BeeIP to achieve similar throughput to DSR and AOMDV.

Network life

In terms of the average remaining battery at the end of the simulations, there is a slight difference which can be spotted in figure 6.11. All reactive protocols (BeeIP, DSR, AODV and AOMDV) are found to compete with no significant changes (see figure 6.12), for all pause times. On the contrary, DSDV can be considered a less

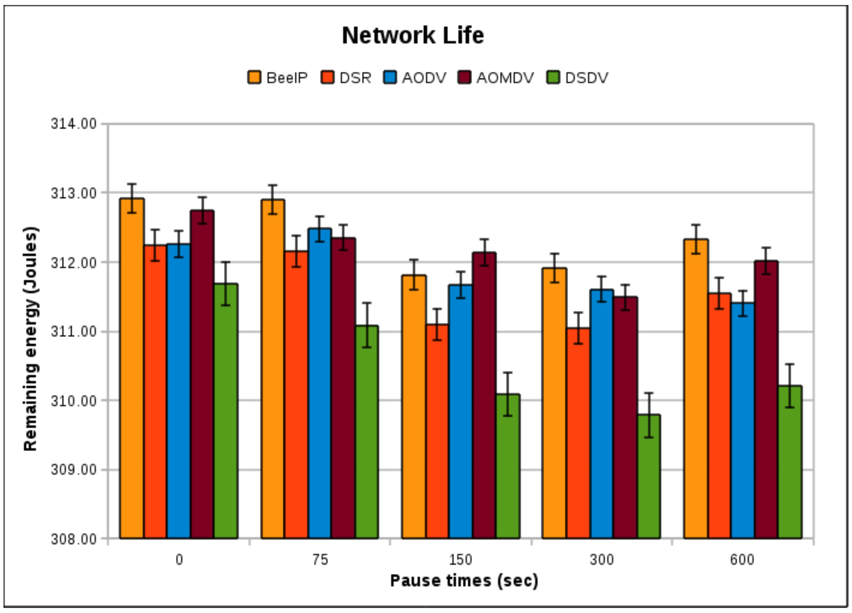


Figure 6.11: Network life as a function of different pause times. All reactive protocols are shown to compete in terms of remaining energy. As expected, DSDV is found to have used more energy, as a result of its proactive route discovery mechanism.

energy-efficient protocol. This is explained by the increased number of control packets that are required to maintain routing paths, due to its proactive behaviour. In conjunction with the average throughput which is illustrated in figure 6.9, it is noticed that as DSDV's throughput performance speeds up for pause times 150s, 300s and 600s, the energy drain is increased.

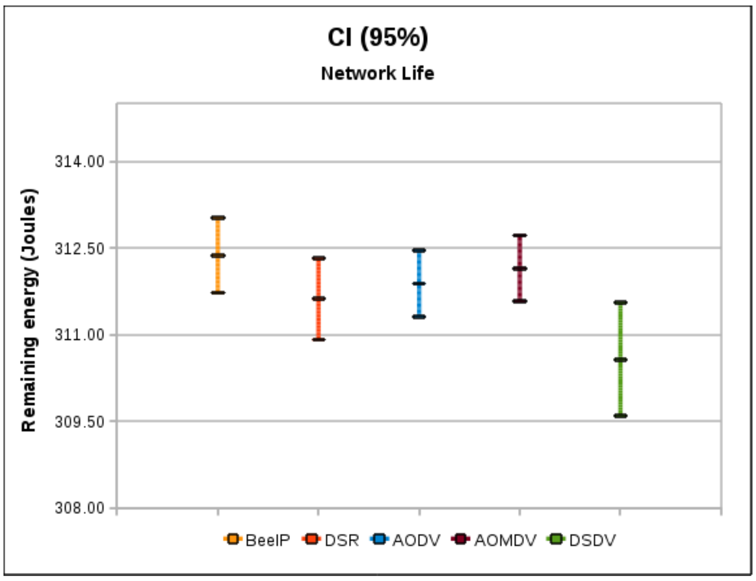


Figure 6.12: 95th confidence interval of the network life as a function of pause times. The only assumption that can be made here, is that BeeIP is significantly better than DSDV in terms of remaining energy after the simulation has finished, and a strong competent amongst the other reactive routing protocols, AODV, AOMDV and DSR.

Route requests (BeeIP, AODV and AOMDV only)

In addition to the above tests, BeeIP, AODV, and AOMDV have been studied in a bit more detail due to their common route request mechanisms. Table 6.2 below shows the average number of route requests (or scouting processes in the context of bee-inspired routing) initialized during each simulation, when varying the pause time. The difference in route requests is in favour of BeeIP, which also explains the difference in the control overhead the protocols produce, as shown in figure 6.3. DSR is not participating in this comparison mainly because of its difference in route request process. Namely, the route cache technique being used for constructing RREPs. In DSR, if an intermediate node receiving a RREQ has a route to the destination node in its cache, the it replies to the source node by sending a RREP with the entire path from the source to the destination node.

Protocol: \ *Pause times(s):*	0	75	150	300	600
BeeIP	4796	3296	4169	3995	4589
AODV	5511	4595	5805	4950	5925
AOMDV	5117	4004	5478	4646	5802

Table 6.2: Number of route requests during 600s of simulation, when studying pause times. For all different situations, BeeIP is shown to have started less route discoveries than AODV and its multi-path extension AOMDV.

6.2.2 Studying terrain size

The scalability in terms of terrain sizes of BeeIP compared to the other protocols is examined by this set of experiments. Here, the pause time is kept to 75 seconds for all 100 nodes, which are allowed to move according to RWP in random speeds from 1 m/s up to 10 m/s. What varies is the terrain size. In particular, results of simulations with 1800 × 600 m^2, 2400 × 800m^2, 3000 × 1000m^2 and 3600 × 1200m^2 are obtained. As previously, 40 TCP sources are constantly transmitting data packets to TCP sinks via 60 data sessions, for 600 seconds.

Packet delivery ratio

In the results found in figure 6.13, one can see that BeeIP, AODV and AOMDV have similar behaviour with BeeIP outperforming the other two, which have slightly different packet delivery ratio during the simulations for different network areas. All protocols outperform DSDV which struggles as the density of the topology decreases. Facing the problem of the stale entries in nodes' cache, DSR also shows a decreasing performance in terms of packet delivery ratio, as the distance between the nodes exceeds the transmission range and causes frequent broken links.

Control overhead

Nevertheless, the effect is not the same for the control overhead. It is to be expected that as the network area increases in size, broken links occur more often triggering new route discovery processes. In figure 6.14 the control overhead

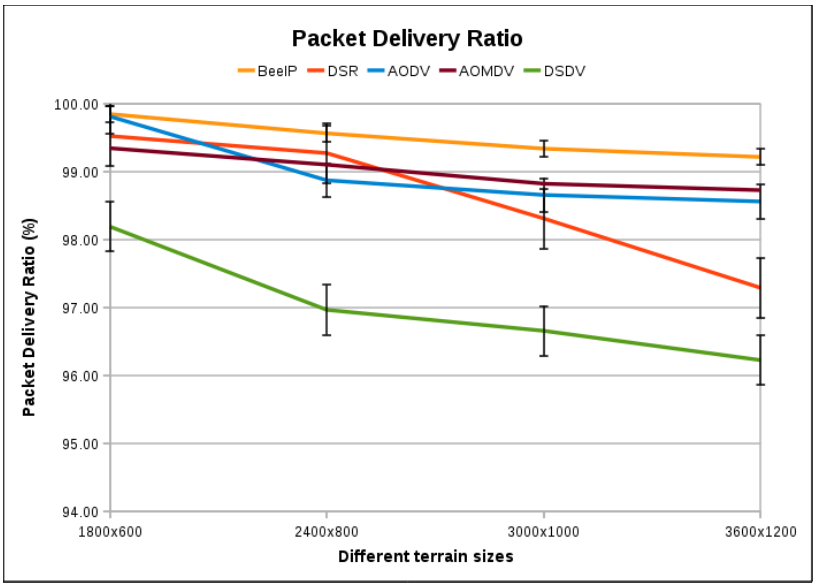


Figure 6.13: Packet delivery ratio as a function of different terrain sizes. BeeIP is shown to outperform all other routing protocols, achieving better packet delivery ratio for all network sizes. Here, as the number of nodes remains the same, the density of the topology is under investigation.

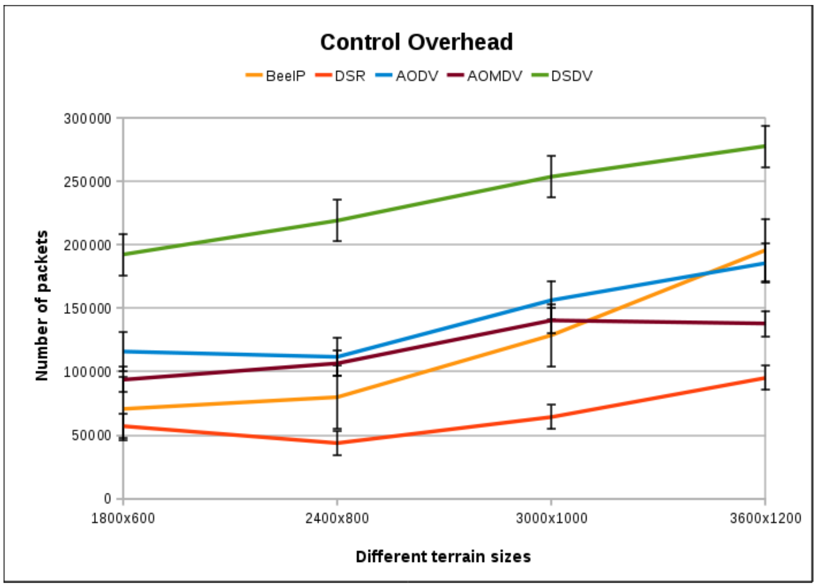


Figure 6.14: Control overhead as a function of different terrain sizes. Here, BeeIP is shown to produce less routing packets than both AODV and AOMDV for dense network topologies. When the density of the topology is reduced, BeeIP is shown to be outperformed, due to its aggressive scouting.

caused during the simulations is illustrated. The aggressive behaviour of BeeIP is again shown, especially within terrain size of $3600 \times 1200m^2$. In this case, the increase in terms of control overhead is expected, as paths tend to be judged as having low quality. In order to understand it better, it is important to bring to mind what the effect of a long distance between two nodes is. It dramatically affects the transmission signal strength of the packet. This is also shown in the previous chapter 4 and, in particular, 4.6.2, where the investigation towards the understanding of the relevant important of each low-level parameter is presented. Therefore, BeeIP shows better results within network areas of high density, than operating within sparse terrain environments.

Although BeeIP's performance gets worst as the terrain size increases, for the first three terrain sizes (up to $3000 \times 1000m^2$) its performance is better than AODV and AOMDV. In turn, they are all outperformed by DSR, which although shows an increasing behaviour as the terrain also increases, it still manages to keep the number of control packets low. Last, DSDV shows a poor behaviour compared to the others, as it is negatively affected as the terrain size increases.

Average end-to-end delay

In terms of this performance metric, a deteriorating performance for DSR is shown in figure 6.15, where the average end-to-end delay for each protocol is presented. Although according to the standard error bars there is no significant difference between the protocols up to $2400 \times 800m^2$ terrain size, packets routed by DSR as well as AODV tend to face higher delays than BeeIP. This is another indication that the multi-path distribution of packets that BeeIP applies pays off, in conjunction with the current selection metric used by the protocol (paths with lowest transmission and queuing delays are preferred). It is important to notice that although AOMDV is also multi-path protocol, the way it switches between alternative paths differs from BeeIP. AOMDV starts utilizing an alternative path when the initial is found to be broken, whereas BeeIP switches between them depending on the number of foragers that are allowed to be recruited to each one. Finally, the small standard error bar of BeeIP indicates a balanced behaviour.

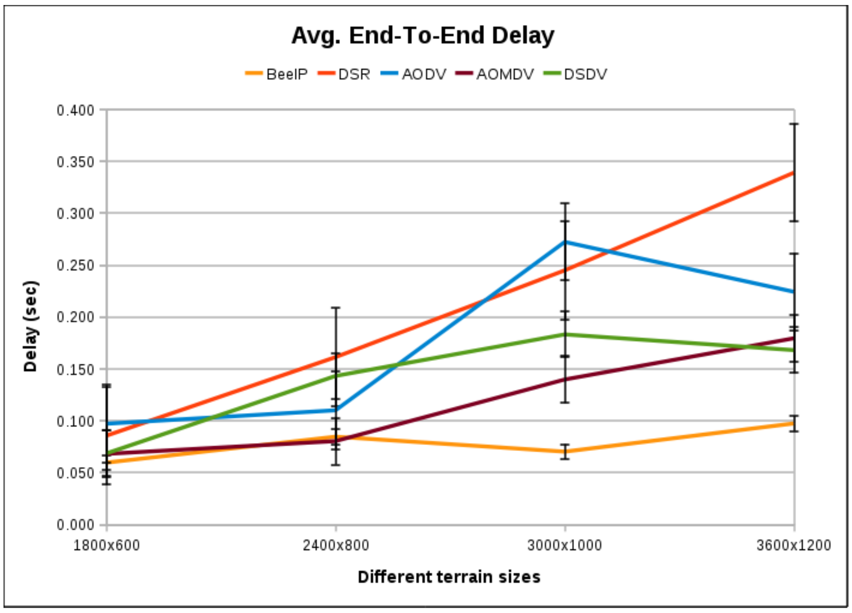


Figure 6.15: Average end-to-end delay as a function of different terrain sizes. Here, is shown that up to $2800 \times 800m^2$ terrain sizes, Both BeeIP and AOMDV achieve equal performance, outperforming all other protocols. However, BeeIP is able to keep an equally good end-to-end delay for larger terrain sizes where the network topologies become less dense.

Average path duration

The average path duration is illustrated in figure 6.16. The standard error bars show that up to $2400 \times 800m^2$ there is no significant difference. However, BeeIP is clearly outperformed by DSR, AODV and AOMDV as the network size increases. This is an expected behaviour of the protocol. For the largest area all three reactive protocols achieve similar results, whereas BeeIP swaps between alternative links reducing the average duration.

Average throughput

The average throughput is kept high for BeeIP, as illustrated in figure 6.17. Due to already increased packet delivery radio and low end-to-end delay, BeeIP is able to keep the average throughput between the data sessions high. A common assumption to all protocols is that as the density gets lower, the average throughput is negatively affected.

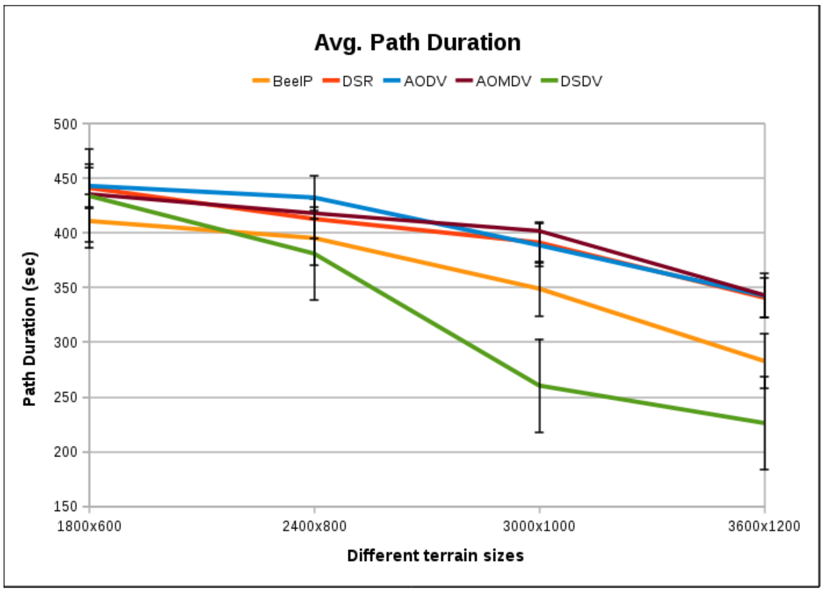


Figure 6.16: Average path duration as a function of different terrain sizes. Here, BeeIP is shown to be outperformed by all three reactive routing protocols, which manage to score a better average path duration for all terrain sizes, at the end of the simulations.

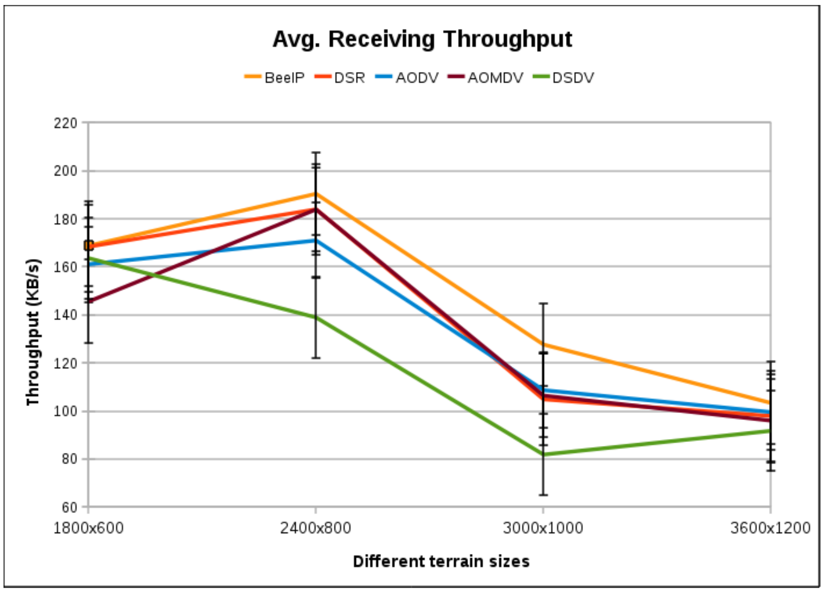


Figure 6.17: Average throughput as a function of different terrain sizes. Here, BeeIP is shown to outperform all other routing protocols, achieving the highest throughput for all terrain sizes.

Network life

When considering energy consumption, there is again no significant difference seen between the five protocols in figure 6.18. There is a slight variation which is reported in favour of BeeIP and AOMDV during the last two simulation experiments (areas $3000 \times 1000\ m^2$ and $3600 \times 1200 m^2$), but no clear assumptions can be made.

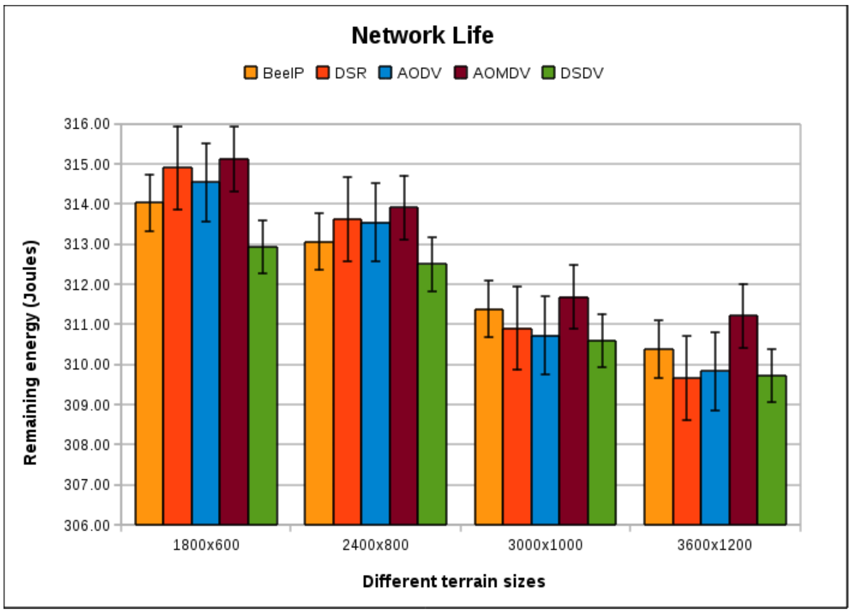


Figure 6.18: Network life as a function of different terrain sizes. Similarly to other comparison results, no significant difference is shown between the protocols as none of them is designed to be energy-efficient.

6.2.3 Studying network traffic

The effect of varying the number of active sources and data sessions to the performance of the protocols, is examined with this set of experiments. The base scenario is used, that is, 100 nodes moving according to RWP with random speeds between 1m/s to 10m/s is used in a $3000 \times 1000 m^2$ terrain for 600 seconds. The pause time is fixed to 75 seconds. The number of active sources and data connections are gradually increased, in order to investigate BeeIP's robustness and compare it to other protocols. TCP sources and sinks are again picked up randomly, starting from 10/15 (sources/connections) to 60/80, with intermediate values of 20/25, 30/45, 40/60 and 50/80. These values are designed to help build a solid understanding of how good or bad each protocol performs, under stressful situation of low (10-20), medium (30-40), and high (50-60) traffic loads.

Packet delivery ratio

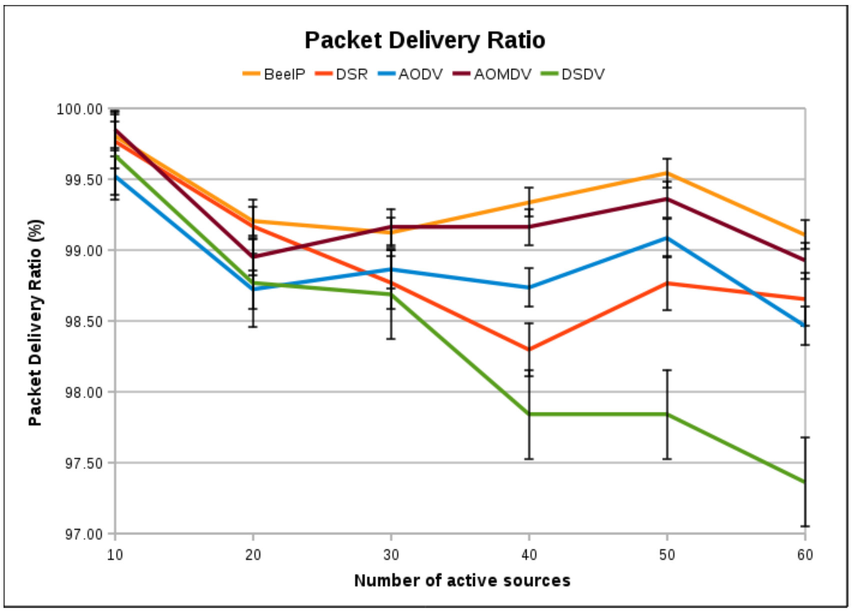


Figure 6.19: Packet delivery ratio as a function of different number of active sources. Here, BeeIP is found to perform better than the other protocols, achieving a better PDR especially when the traffic is high (40-60 active sources).

An initial observation that can be made from figure 6.19 is that the delivery ratio achieved by all five protocols is between 97% (DSDV with 60 active sources) and 99.8% (BeeIP and AOMDV with 10 active sources). Initially, this might be considered too high and unrealistic compared to other similarly configured experiments which are found in the literature (Arun Kumar B. R. *et al.*, 2008; Rahman & Zukarnain, 2009). However, the traffic model used in the majority of those comparisons is constant bitrate (CBR) that uses UDP instead of TCP (refer to section 2.3.4 for details). Depending on the configuration, CBR has significantly higher and constant sending rate which causes congestion points (bottle necks), interference, and eventually increased packet loss. Furthermore, being an unreliable protocol UDP does not guarantee that the packet is successfully accepted by a destination, as TCP does by utilizing packet acknowledgements. Rather, UDP just sends the packet to its way from the source to the destination. So, TCP by its nature will increase the control overhead[1] and thus will negatively affect the performance of the protocols. It is that mechanism, amongst other issues (refer to chapter 2 for a detailed discussion), that researchers prefer to use UDP as the transport layer protocol. Nevertheless, a comparison with both CBR and TCP traffic has been made in Mali and Barwar (2011), producing similar results to the ones presented in this thesis.

Contrariwise, BeeIP is designed to take advantage of that mechanism by piggybacking the acknowledgement packets to ack_foragers that are sent back to the source node of a communication session. To continue with the observation in terms of packet delivery ratio, BeeIP performs better than the others. The difference is clearly observed when 40 to 60 nodes act as sources. Additionally, the proactive approach is proven quite unreliable as DSDV's performance rapidly deteriorates as the number of nodes increases.

Control overhead

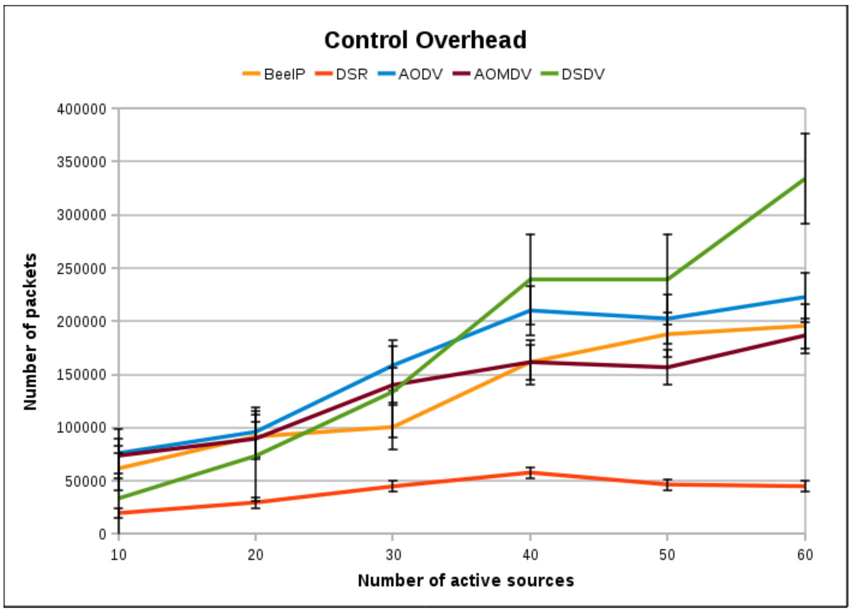


Figure 6.20: Control overhead as a function of different number of active sources. Here it is shown that DSR outperforms all other four protocols. BeeIP and AOMDV are found to follow and achieve better results than AODV.

Varying the number of active sources and data connections also affects the

[1]Here, control overhead refers to the transport layer packets being sent in order to successfully and efficiently transfer TCP segments.

control overhead. The results are included in figure 6.20, where DSR is proven the best performing protocol. BeeIP and AOMDV come second by transmitting less control packets particularly when 30 and 40 sources are active (in BeeIP) and 50 and 60 (in AOMDV). Both multi-path routing protocols seem to perform better than the single-path AODV. Moreover, the periodically and incrementally transmitted routing advertisements produced by DSDV when changes occur, are noticeable under high traffic networks.

Average end-to-end delay

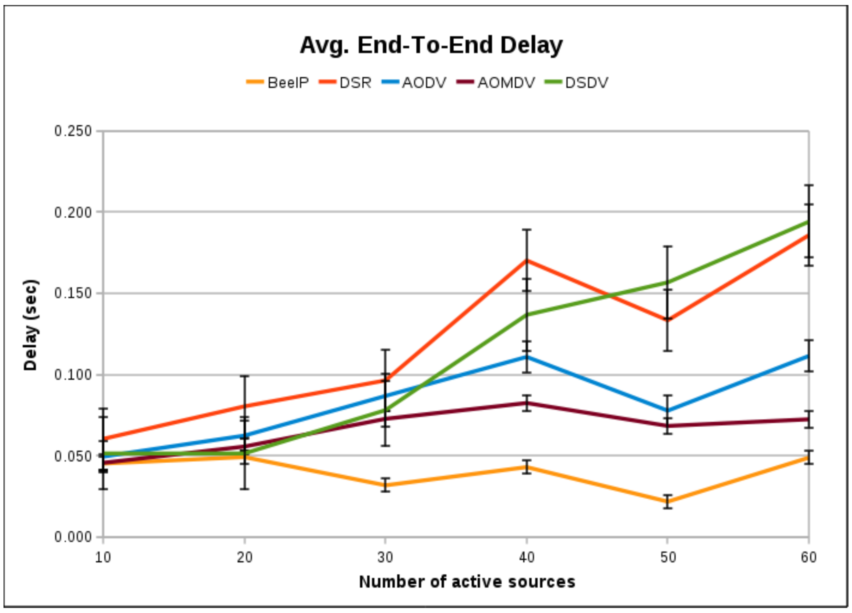


Figure 6.21: Average end-to-end delay as a function of different number of active sources. Here, BeeIP is shown to have a balanced and low average end-to-end delay. Coupled with the high packet delivery ratio, this complements the protocol's robustness under high network traffic.

The average end-to-end delay is illustrated in figure 6.21. When the network traffic is low, the observed difference between the protocols is quite small. This changes though for medium to high traffic, where BeeIP is shown to be the best performing protocol. In addition to that, the error bars of the standard error from the mean for BeeIP, AOMDV and AODV are rather short, from which a balanced end-to-end delay is concluded. The latter, together with the already high packet delivery ratio highlights the protocols robustness under several network traffic conditions.

Average path duration

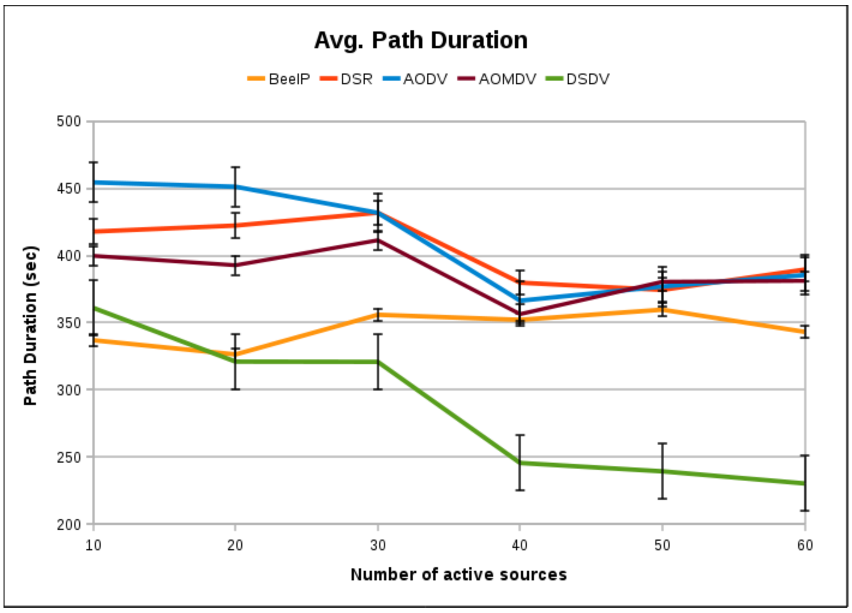


Figure 6.22: Average path duration as a function of different number of active sources. Here, the effect of low and high traffic in connection with the aggressive scouting of BeeIP is illustrated. BeeIP has the lowest average path duration when the traffic is low. However, the difference is getting smaller as the network traffic is increased.

The aggressive scouting of BeeIP appears again when the average duration of paths during the simulation is examined. This is shown in figure 6.22 when low traffic is applied to the wireless network. Under the same situation, the reactive approach makes the other three protocols, AODV, AOMDV and DSR, perform better with the first outperforming the other two. Furthermore, the difference between the reactive protocols is reduced, when the traffic is medium to high. DSDV on the other hand, is following the opposite trend. The average path duration is reduced dramatically when the traffic is increased. This is an indication of how sensitive the protocol is to stressful conditions and network bottlenecks.

Average throughput

The above verdicts are also made by looking at figure 6.23 where the average throughput is illustrated. Here, the negative affect of the high network traffic is visible to all protocols. AODV, AOMDV and DSR are less sensitive to bottle necks,

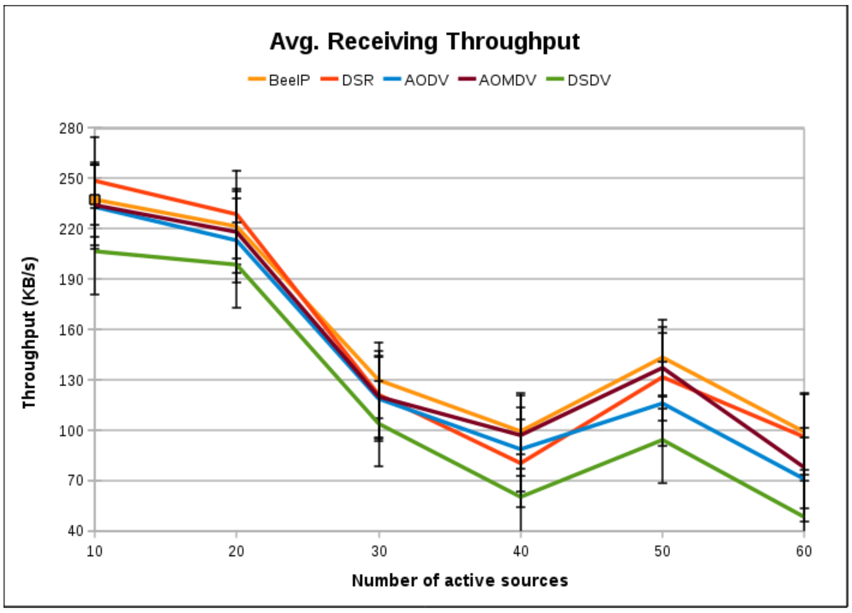


Figure 6.23: Average throughput as a function of different number of active sources. All protocols are affected by high traffic, and achieve low throughputs. BeeIP is shown to have better performance, when compared to other reactive protocols and, in particular, AOMDV which is able to discover multiple alternatives between a source and a destination.

than DSDV which has the worst performance (only ∼45KB/s average throughput within busy networks, compared to ∼98KB/s of BeeIP and ∼95KB/s of DSR). Also, BeeIP is capable of handling high traffic much better as a result of being able to use concurrently multiple paths between sources and destinations.

Moreover, between BeeIP and AOMDV (the two multi-path routing protocols) it is seen that they are both able to overcome network congestion better than the others with a slightly better throughput, under medium to high traffic (40 and 50 active sources). However, their different approach in multi-path usage pays off for BeeIP, which is found to overcome all other protocols under high traffic.

Network life

Considering network life, the results in figure 6.24 bear resemblance to those of the previous sets of experiments. No clear statistical significance can be detected between their differences, which is partially expected since none of these protocols is designed to be energy-efficient[2]. Still, it is seen that BeeIP has a constantly

[2]Although BeeIP is designed to easily accept different selection metrics, as discussed in 4.7.

decreasing behaviour, with no extreme values reported.

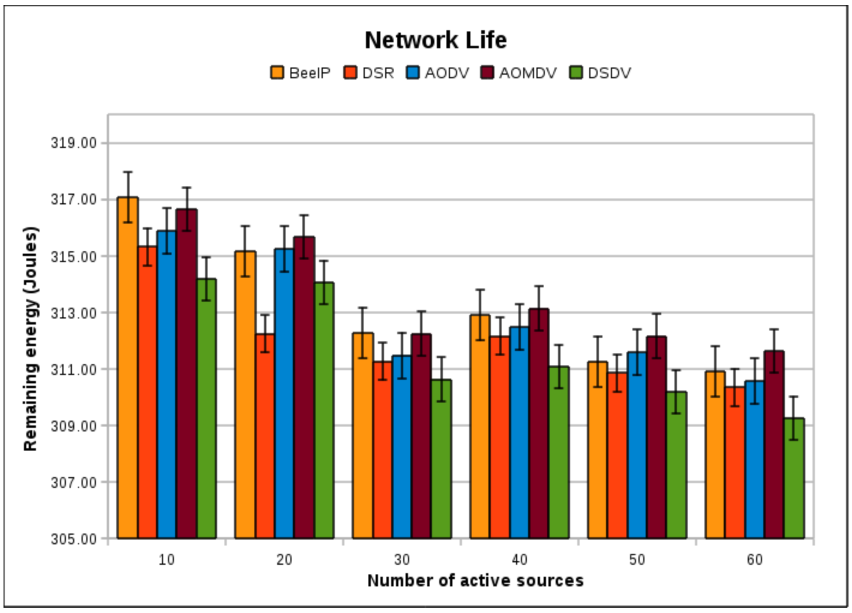


Figure 6.24: Network life as a function of different number of active sources. Similarly to previous results, there is no significant difference between the protocols' behaviour, nor any extreme values are reported.

6.2.4 Studying node speeds

The experiments included in this section study the adaptation to frequent link breaks and robustness of each protocol. The scenario is designed as follows. For 600 seconds, 100 nodes are moving according to the RWP model in a network area of $3000 \times 1000m^2$. The pause time is fixed to 75 seconds. Similarly, the number of TCP sources and connections are fixed to 40 and 60, respectively. The experiments are repeated by gradually widening the speed range of the nodes, starting from 1-5 m/s, to 1-10 m/s, 1-15 m/s and 1-20 m/s. As the maximum speed that the nodes can reach is increased, the average speed is also increased, as well as the frequency by which links break. Thus, the aim is to investigate how the protocols are able to adapt to the topological changes. It is worth mention that 1 metre per second is a good walking speed for a man, whereas 20 metres per second is the speed of a car being driven slowly, e.g., in a city centre.

Packet delivery ratio

As previously, the packet delivery ratio is the first performance metric to be examined. Figure 6.25 shows that BeeIP keeps a steady performance, slightly deteriorating as the mobility of the nodes increases. Its packet delivery ratio varies from ∼98.9% to ∼99.6% when the second best reactive protocol AOMDV fluctuates between ∼98.8% and ∼99.4%. AODV and DSR are found to have more packet loss for highly dynamic networks, amongst the reactive protocols in comparison. Moreover, the proactive DSDV is also losing in delivering packets on time, as its packet deliver ratio is dropped to ∼97.5% for speed range 1-20m/s.

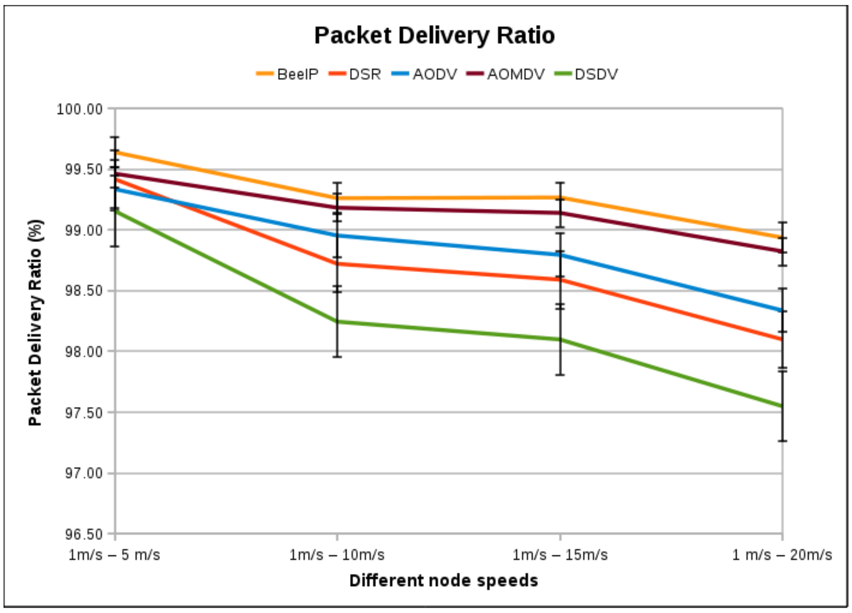


Figure 6.25: Packet delivery ratio as a function of different node speeds. Amongst the reactive protocols, BeeIP and AOMDV are shown to be less affected by the mobility, with BeeIP having a slightly better performance.

Control overhead

In terms of control overhead in figure 6.26, DSR is found to be the least sensitive protocol. In addition, BeeIP, AODV and AOMDV increase the amount of packets required to maintain routing, with BeeIP achieving a lower overhead.

The results obtained from this experiment show that the multi-path extension to AODV is more sensitive to high mobility networks, in terms of the number of control packets being produced. This agrees with the results obtained in Marina

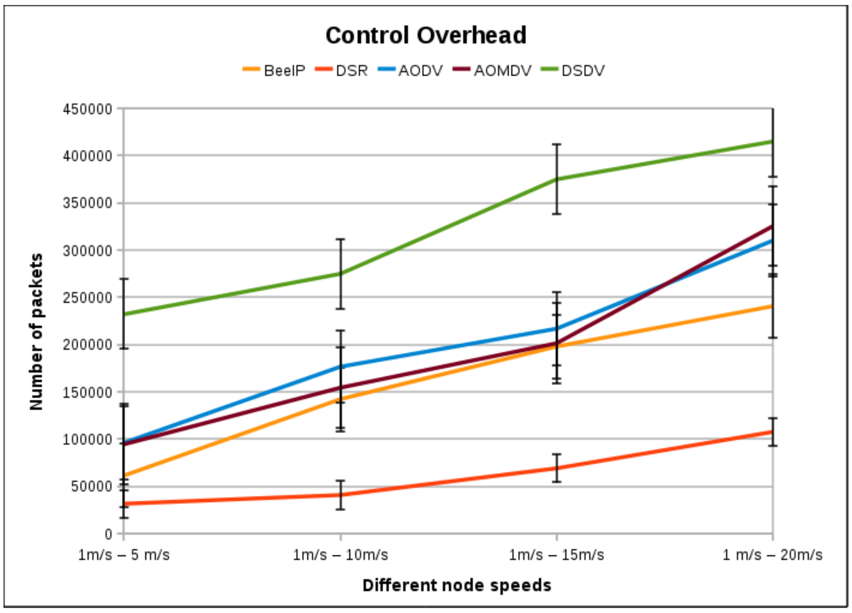


Figure 6.26: Control overhead as a function of different node speeds. BeeIP is shown to be a strong competent to other reactive protocols, namely AODV and AOMDV. In particular, BeeIP is found to less control overhead for high mobility networks, outperformed only by DSR.

and Das (2006), where the frequency of route discoveries and routing overhead is similar for AODV and AOMDV with varying mobility. Finally, increasing the average speed of the nodes affects the proactive DSDV more, as route update messages flood the network in an effort to maintain routing.

Average end-to-end delay

In terms of end-to-end delay, the results are summarized in figure 6.27 and are as expected. DSDV follows the opposite trend compared to the reactive protocols, as its ability to adapt to consistently dynamic networks is quite limited. Again, BeeIP is shown to be able to perform better than AODV, AOMDV and DSR, achieving a balanced packet delay. Keeping in mind that the path selection metric of BeeIP is set to the fastest path (see section 4.7.1), the low and balanced behaviour of BeeIP in connection to its high packet delivery ratio can be considered an indication of achieving efficient and optimal routing.

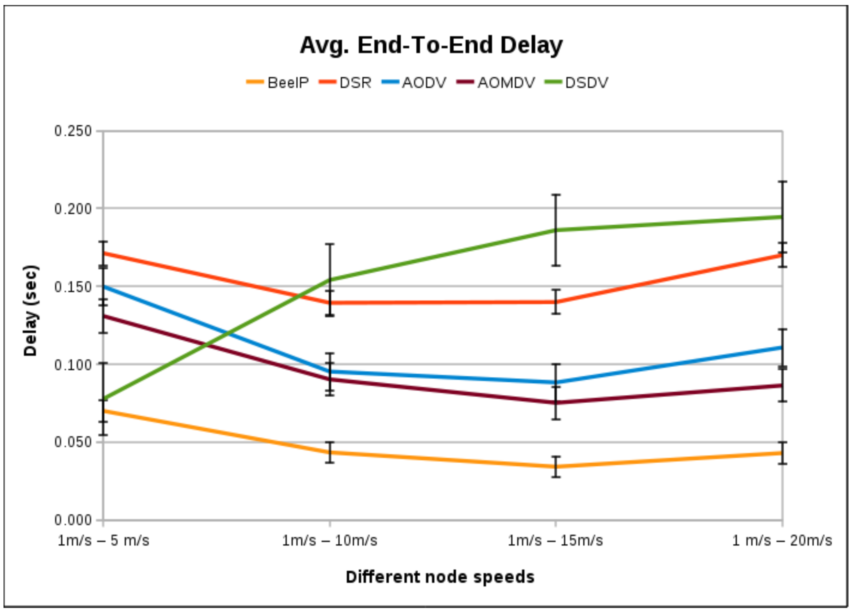


Figure 6.27: Average end-to-end delay as a function of different node speeds. BeeIP is shown to result in low and balanced average end-to-end delay over the other reactive routing protocols. DSDV follows an opposite trend as it deteriorates as the average moving speed of nodes increases.

Average path duration

The average path duration results are shown in figure 6.28. Faster nodes lead to frequent link breaks, and thus paths are cut sooner. BeeIP is in a more difficult position compared to the other three reactive protocols, because of its aggressive scouting. Although speed is less important parameter for BeeIP's monitoring mechanism (as described in 4.6.2), it affects the distance between two communicating nodes. For this reason, the signal strength of packet transmissions become weaker. This itself leads to dull artificial honeybee dances by the ack_foragers that consistently reduce each path's foraging capacity.

Average throughput

Considering the average throughput in figure 6.29, it is understood that the performance of all the protocols deteriorates as the average speed becomes higher. Due to the highly dynamic network when the node speed range is 1-20m/s, BeeIP's average throughput is less than DSR and AODV. Nevertheless, under less stressful conditions, namely when the speed ranges are 1-5m/s to 1-15m/s, BeeIP is

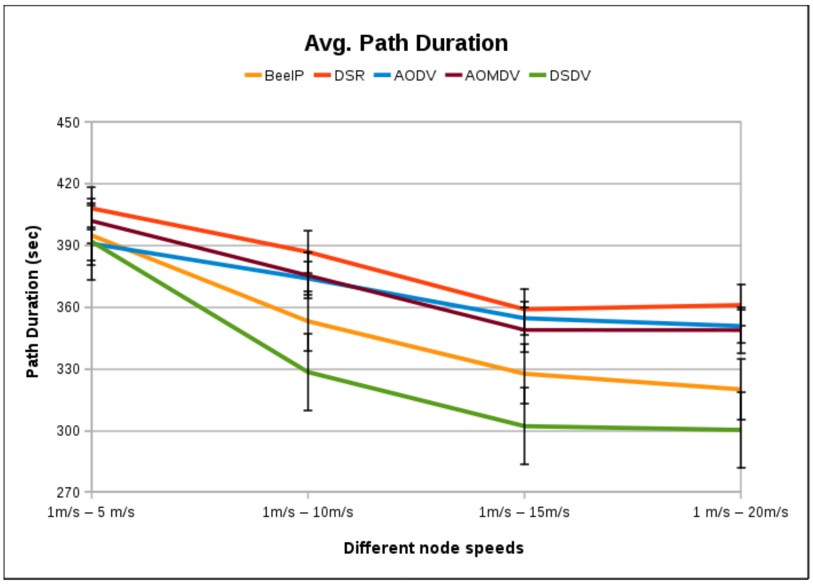


Figure 6.28: Average path duration as a function of different node speeds. The aggressive scouting is again shown to be a downside of the protocol. Although all protocols are found to deteriorate in average path duration due to frequent link breaks, BeeIP is found to be in the worst position followed only by DSDV.

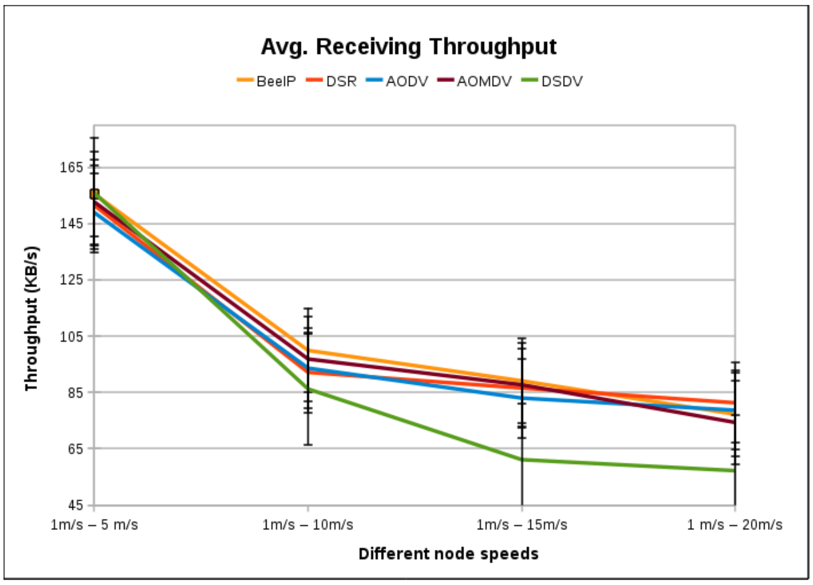


Figure 6.29: Average throughput as a function of different node speeds. BeeIP is shown to achieve better results than the other reactive protocols, AODV, AOMDV and DSR, for low to average mobility networks. Its performance is observed to be deteriorating when the average moving speed of the nodes is high (1-20m/s). Here, BeeIP is outperformed by AODV and DSR.

found to strongly compete with the other reactive protocols, achieving better performance (∼155KB/s, ∼100KB/s, ∼89KB/s), followed by AOMDV (∼152KB/s, ∼96KB/s, ∼87KB/s). To conclude, DSDV throughput is low, demonstrating its need to incrementally exchange route update messages to maintain routing, wasting a significant amount of bandwidth.

Network life

At the end of the simulations, DSDV tends to have low average remaining energy and, in particular, when the average moving speed is medium to high. The results are summarized in figure 6.30. DSDV has a deteriorating performance mainly because of the amount of control overhead being spent, as explained previously. Otherwise, all protocols seem to be equally affected since the resulting numbers are similar and no significant difference is shown by the standard error bars. To this, AOMDV is found to be an exception, as it is observed to outperform BeeIP, AODV and DSR for node moving speeds up to 1-15m/s.

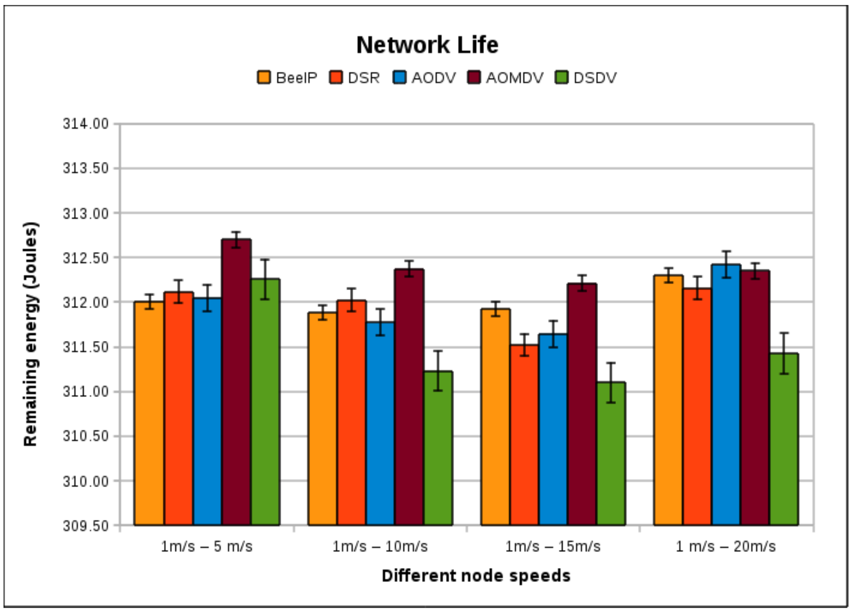


Figure 6.30: Network life as a function of different node speeds. Constantly moving nodes affect all protocols equally. As in previous experimental scenarios, no extreme values are reported.

Route requests (BeeIP, AODV and AOMDV only)

In order to investigate further the table 6.3 is produced, summarizing the total number of route requests and scouting processes started by the three protocols. The table highlights the effectiveness of allowing a BeeIP scouting process to discover more than one path between the source and the destination node, ending in collecting multiple routing alternatives. Between AODV and AOMDV, it is observed that the latter starts less route discoveries, which are able to find more than one paths. Although being multi-path like BeeIP, AOMDV still incurs extra overhead, especially when the mobility of the network is increased. Again, this is reasoned by the protocol's approach to multi-path routing, i.e., an alternative path is used only when the current one breaks. In a high mobility network paths are more likely to break before being utilized as next hops.

Protocol: \ *Node speeds(m/s):*	1-5	1-10	1-15	1-25
BeeIP	2681	3424	5117	6236
AODV	3483	5752	6033	8996
AOMDV	3164	5504	5905	8067

Table 6.3: Number of route requests during 600s of simulation, when studying node speeds.

6.2.5 Studying number of nodes

For the last set of experiments, the scalability of the protocol in terms of adapting to different network sizes is investigated. The scenario is similar to the one presented in the previous section, namely the nodes are moving in an area of 3000 $\times$ 1000m^2 for 600 seconds. The pause time is again fixed to 75 seconds. Nodes are moving around in a RWP fashion, at speeds between 1m/s to 10m/s. The simulations have been repeated for 50, 100, 150 and 200 nodes. Each time, almost one third of the total number of nodes are used as active sources. Namely, 15 active sources for 50 nodes, 30 for 100, 50 for 150 and 60 for 200 nodes.

Packet delivery ratio

Looking at the packet delivery ratio, as presented in figure 6.31, it is seen that all reactive protocols can easily adapt to the network sizes, with BeeIP outperforming

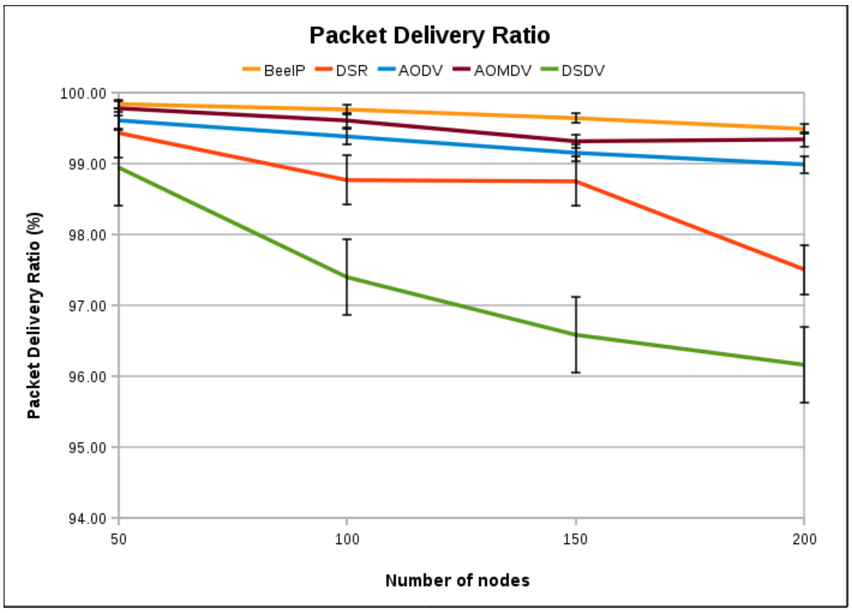


Figure 6.31: Packet delivery ratio as a function of different number of nodes.

AODV, AOMDV and DSR. The latter is becoming less strong in terms of delivery efficiency as the number of nodes increases, achieving an average of ∼97.5% for 200 nodes. On the other hand, the local repair proves to be effective in enhancing AODV as under similar conditions its average does not drop lower than ∼99%. Compared to DSDV, BeeIP is clearly the best performer as the proactive nature of the first produces much more control overhead which slows down the real data transmissions.

Control overhead

In terms of control overhead, figure 6.32 shows that BeeIP outperforms AODV, AOMDV and DSDV as they all tend to take bigger steps increasing the number of control packets as the network size is getting bigger. DSR performance is steadily low, which is to be expected, again, as it lacks a local repair mechanism. This is not the case for BeeIP though, which even though it does not use local repair as AODV or AOMDV do, it initializes more route discovery processes than DSR, and thus, produces more control overhead.

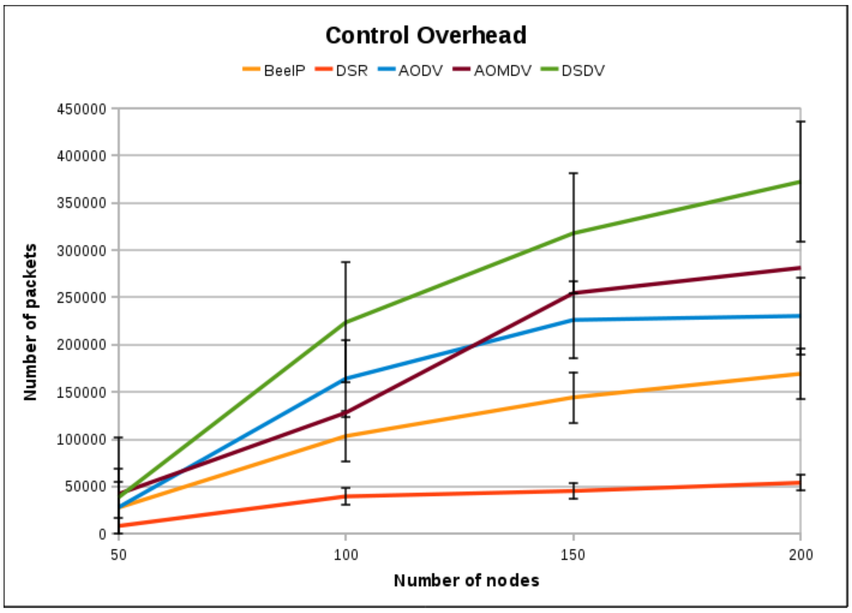


Figure 6.32: Control overhead as a function of different number of nodes. Here, BeeIP is shown to take bigger steps than DSR in terms of the number of routing packets, but it outperforms AODV, AOMDV and the proactive DSDV.

Average end-to-end delay

Results for the end-to-end packet delays are shown in figure 6.33. Although BeeIP outperforms the others, the delays for all protocols are kept quite low. This is because the TCP traffic load is also kept balanced, as explained at the beginning of this section. Moreover, from the error bars of the obtained results it can be seen than BeeIP has a constant behaviour for all network sizes. This is one of the protocol's strengths which is also confirmed by all the previous experimental results. Additionally, with a large number of nodes participating in the network, the number of hops between a source and a destination is also increased. This causes packets routed by DSR to face an increased delay, due to the fact that the source routing mechanism increases the size of the packet by adding a considerable amount of data to its header. This extra size is proportional to the length of the path to be traversed, and affects the packets transmission delay.

Average path duration

The average path duration performance metric is again highlighting BeeIP's aggressiveness in starting new scouting processes. As generally found during the

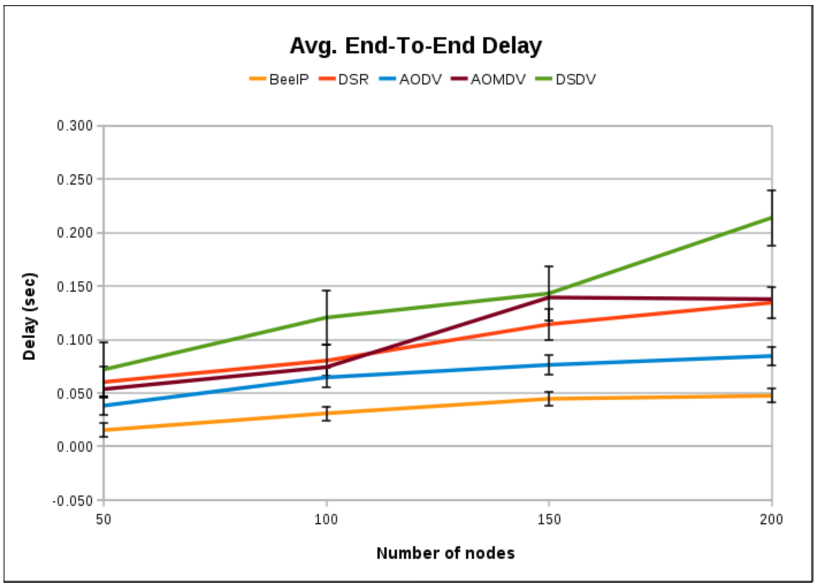


Figure 6.33: Average end-to-end delay as a function of different number of nodes. Here, BeeIP is shown to outperform all other protocols, for all different numbers of nodes in the topology. The standard error from the mean bars indicate a balanced behaviour.

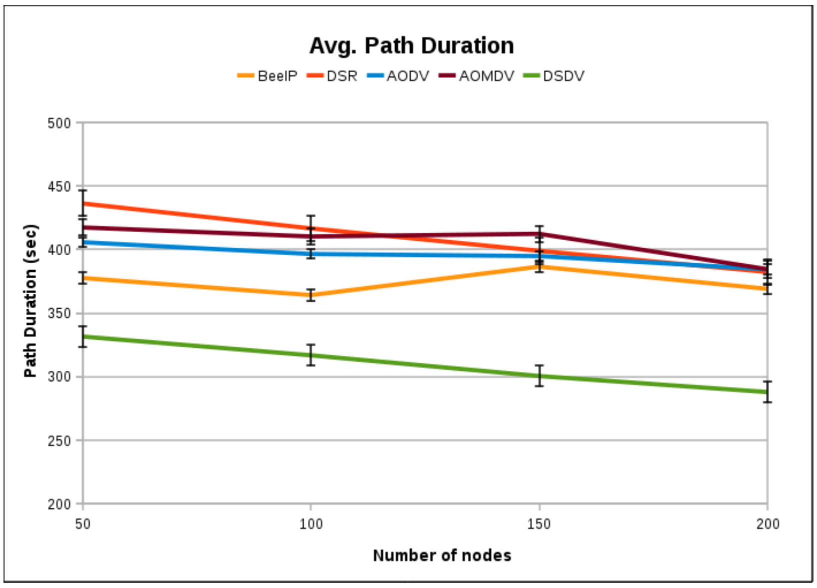


Figure 6.34: Average path duration as a function of different number of nodes. BeeIP's aggressive scouting behaviour is what keeps the average small, up to 100 nodes. However, as the number of nodes increases more alternative paths are created, which pays off as scouts can always make new discoveries.

experiments in this chapter, the average duration of which links between sources and destinations are used to constantly transmit data using BeeIP, is lower than AODV, AOMDV and DSR. This is also observed in this set of experiments, when the number of nodes is 50 and 100. However, it is found that the difference between the BeeIP and the other reactive protocols is reduced as the network increases in size. In these cases, the performance of the protocol is balanced by combining the aggressive scouting with the multi-path feature, which is a sign that the protocol mimics nicely the honeybees' behaviour in the real world. Figure 6.34 illustrates the results.

Average throughput

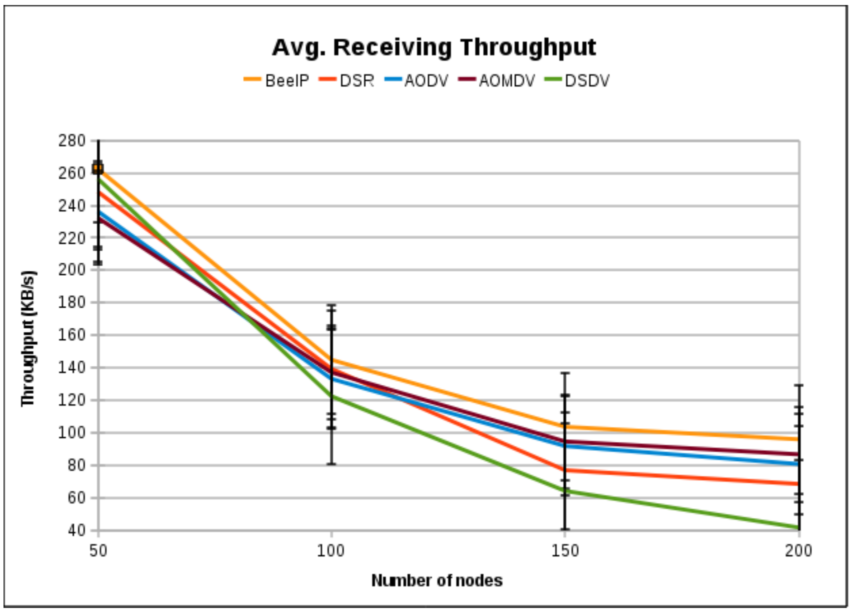


Figure 6.35: Average throughput as a function of different number of nodes. BeeIP is found to perform better than the others, keeping a constantly higher average throughput. As number of the nodes is increased, multi-path protocols are in favour.

Finally, the average throughput is illustrated in figure 6.35. The following similar impression for all protocols is made; the average throughput is deteriorating as the number of nodes increases. However, by studying the graph lines and the values obtained by the experiments, further observations can be made. Firstly, the lines representing BeeIP and AODV are downgrading by taking smaller steps than DSR, when the number of nodes changes from 100 to 150 and finally to 200.

This is an indication that the protocols are less sensitive to varying this particular network condition. Secondly, BeeIP outperforms the other protocols, especially when the network size is large, by utilizing its concurrent use of paths to spread the load to alternative paths avoiding congestion points and bottlenecks. Thirdly, by using their local repair mechanism AODV and AOMDV are able to detect broken links and update their routing table faster than DSR, which due to stale entries sends packets in out-of-date directions. The latter is also explained by the already low packet delivery ratio of DSR for 150 and 200 nodes, presented in figure 6.31.

Network life

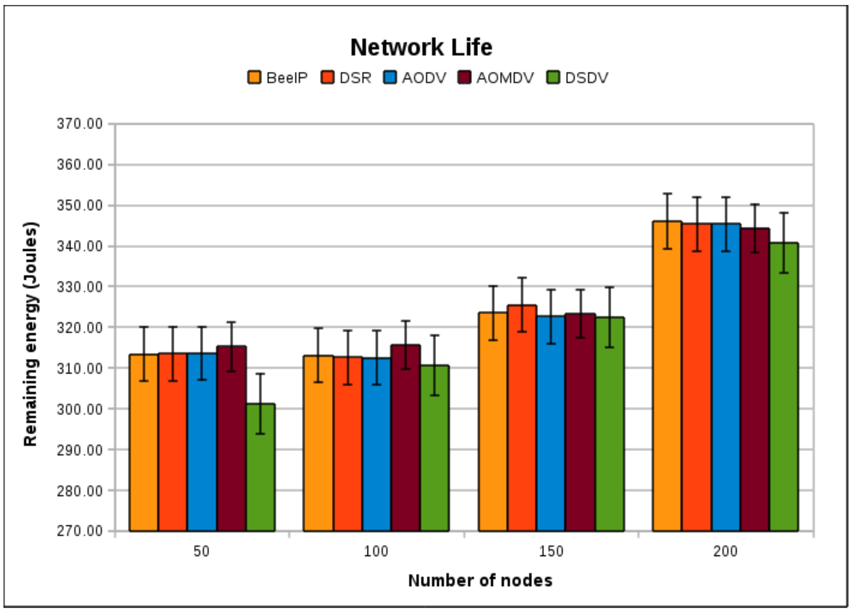


Figure 6.36: Network life as a function of different number of nodes. Here, BeeIP is shown to produce similar results to the other protocols, which is an expected behaviour.

When considering the energy consumption, similar results to the previous experiments are observed. An interesting outcome from 6.36 is that the average remaining energy is much higher for 200 nodes, than for smaller networks. In fact, keeping in mind the size of the terrain ($3000 \times 1000m^2$), 50 nodes form a topology of pretty low density. Therefore, less paths are physically available between the sources and the destinations, and fewer nodes are used to transmit packets. Hence, the battery of a particular group of nodes is drained faster, leading to a lower average remaining energy. When 200 nodes participate in the topology, more paths are physically available. This keeps more nodes busy, achieving a better distribution of the energy consumption.

6.2.6 Summary of the observations

When considering the pause times, BeeIP is found rather insensitive to changes showing the significantly better packet delivery ratio, control overhead, and end-to-end delay. Its results in connection to the average throughput are also positive, showing that BeeIP competes with DSR and is able to transmit more KB/s under low or high dynamic networks than AODV, AOMDV and DSR. A negative observation is BeeIP's average path duration, where due to its aggressive scouting behaviour the average time a link stays active is smaller compared to the other protocols. Considering the remaining network life at the end of the simulations, although no significant difference is reported, BeeIP shows slightly better results.

When changing the network terrain size, the results indicate that BeeIP, AODV and AOMDV are outperforming DSR and DSDV, in terms of packet delivery ratio and end-to-end delay. Additionally, BeeIP has a better average throughput than the others and, based on the results, it is understood that the protocol is able to operate more efficiently in a network with higher density. In terms of control overhead, a difference in favour of BeeIP compared to AODV and its multi-path extension AOMDV is reported initially ($1800 \times 600m^2$, $2400 \times 800m^2$), but it reduces as network size increases. The aggressive scouting feature is again keeping BeeIP's performance behind in terms of average path duration, compared to DSR, AODV and AOMDV, which score equally higher.

Looking at the impact of traffic on the protocol's performance, BeeIP is found to be less sensitive compared to the other protocols. It manages to achieve high packet delivery ratio, especially under high traffic networks. In terms of control overhead, all protocols' performance is gradually decreased, as the network traffic becomes higher. The experiments have shown that BeeIP manages to send less control packets for medium traffic networks, compared to AODV and DSDV, and is outperformed by DSR which has the least control overhead. BeeIP is found to be the best performing protocol in terms of average end-to-end delay for any traffic model (low, medium, and high). Also, its aggressive scouting behaviour leads to poor average path duration, especially for low traffic networks.

The speed of the nodes in the topology also affects the performance of the protocol. The experiments which have been conducted show that like AODV, BeeIP increases the number of control packets as the speed of nodes is increased. On the contrary, the packet delivery ratio deteriorates slightly. Additionally, BeeIP's average throughput is found to be less than AODV, AOMDV and DSR under highly dynamic networks, where nodes are allowed to reach a speed of 20m/s. However, under less dynamic networks, such as 1-5m/s to 1-15m/s, BeeIP is found to have a better throughput.

Finally, when the number of nodes varies, it is found that all reactive protocols can easily adapt to the network changes, with BeeIP outperforming AODV, AOMDV, DSR, and also DSDV which due to its proactive nature is dramatically affected by the network size. Again, BeeIP's average path duration is outperformed by AODV and DSR for sparse networks, however, when the number of nodes is increased, their performance match. In terms of average throughput, all protocols are downgrading, but BeeIP, AODV and AOMDV take smaller steps than DSR, under medium to high density networks, with BeeIP outperforming.

6.3 Conclusion

In this chapter, an evaluation of the proposed protocol is presented. BeeIP is compared with four well-known routing protocols in the area, namely DSR, AODV, AOMDV, and DSDV. DSR and AODV as well as its multi-path extension AOMDV are following a reactive approach in providing routing solutions. One main difference between the these protocols, is that DSR is using source routing. In source routing, each packet knows the full path to the destination and carries it in its header. DSDV on the other hand, is the de facto proactive protocol for MANETs, heavily used for routing protocol comparisons.

The experiments are separated into five sets, where different network conditions are changed. Scalability, robustness and routing efficiency are studied, the results of which are presented and discussed. To summarize, BeeIP is found to outperform the other protocols in terms of most of the performance metrics, under several conditions. A reported strength of the protocol is its ability to keep low and balanced end-to-end packet delay, even under stressful network conditions such as high traffic and mobility rates. This, in connection to the high average throughput results, is an indication that the decision of the path selection metric (selection of delays, equation 4.6), which was used for the presented comparisons, has proven to pay off as the protocol is able to select the fastest paths between the sources and the destinations of each communication session. Selecting the appropriate path is discussed in section 4.7.1.

On the downside, BeeIP shows an aggressive behaviour as it is constantly monitoring the working paths. That leads to generally smaller average path duration and increased control overhead. Considering these performance metrics, it is observed that under low mobility and/or sparse network topologies, BeeIP is outperformed by AODV, AOMDV and DSR. Furthermore, this behaviour is balanced when the network gains in number of nodes, and BeeIP is able to utilize its multi-path feature more effectively.

Chapter 7

Conclusions and future work

7.1 Overview

In this thesis, the problem of routing in mobile wireless ad hoc networks is addressed using nature inspired techniques. In particular, the problem is solved by emulating the collaborative behaviours of honeybees when engaged in food foraging and communication within their hive. Honeybees are able to find multiple paths to an interesting flower, and by applying several techniques driven by teamwork, they are able to distinguish the most profitable and efficient one, or ones, and use them according to their needs.

Mapping and modelling honeybee behaviours to an agent-based routing protocol for wireless ad hoc networks is a new research area, started with Wedde et al. (2005a) and Saleem and Farooq (2007b), where the routing problem was addressed for MANETs and WSNs respectively. The difference and novelty of the work presented in this thesis lies on three distinct aspects. Firstly, the proposed design focuses on the way artificial honeybees constantly monitor and evaluate the goodness of each path by utilizing a rich number of low-level pieces of information that describe and represent the quality of a wireless transmission of data. This follows the foraging rules of the real insects by which a forager is able to evaluate the goodness of a finding according to a variety of quality factors. Low-level parameters are used by the model (presented in section 4.6 of chapter 4) to calculate the quality of the intermediate links, and then, the overall quality of a path. Secondly, focus is given on the way communication is achieved within the artificial hive. Honeybees exchange information about foraging and paths by the means of a special dance. During a dance, a forager reports the outcome of her latest investigation: an enthusiastic dance concentrates the attention of more waiting foragers and, in turn, has a better chance to recruit new members. Coupled with this, one of the honeybees' abilities in evaluating a finding is to detect any improvement or deterioration over time. In this work, this behaviour is modelled by using a statistical tool which allows the detection of quality changes for each path as time progresses. Finally, the decision of which is the best path a packet must use to be transmitted from a source to a destination, is also modelled according to what is found in nature. It is reported in von Frisch (1967), that foragers get feedback from the workers within the hive. The feedback informs them of which of the three important commodities (i.e., water, nectar and pollen) is running low, and bias their foraging towards the specific commodity. Based on this concept, the best path selection in the proposed system is designed to be done by considering the application's requirements. Considering the application needs, a path is selected amongst a list of promising paths, discovered and evaluated by the artificial foragers. For instance, if the application requires packets to be sent as fast as possible, the selection is done based on the sending delays (queuing plus transmission delay). Thus, the selection metric is designed to be separated from the path evaluation model.

The second and third chapters of this thesis give the related literature review which is important to understand the internal mechanisms and design of the proposed work. In particular, chapter 2 gives an introduction to the field of routing in wireless ad hoc networks and provides an overview of a number of important solutions found in the literature. In addition to several well-known classifications of protocols, such as reactive, proactive and hybrid, Internet-inspired and nature-inspired, location-aware and power-aware, this chapter also introduces the idea of cross-layering. Cross-layering is a collection of techniques which are used to enhance optimality to network protocols, and it is heavily used by adaptive routing protocols which make use of information that is not available to the network layer by default. Being an adaptive routing protocol, BeeIP is designed in a cross-layer manner and therefore is able to access and utilize low-level information generated within different layers (i.e., level of the energy remaining in batteries, size of the interface queue at the MAC layer, moving speed of nodes, etc.).

The third chapter discusses all the exciting information related to real honeybee foraging and communication and builds a solid background for chapter 4, where the new routing protocol, BeeIP, is thoroughly described. Apart from the use of quality factors to evaluate the goodness of a finding, chapter 3 makes the following important points. A hive consists of a large number of similar insects (the workers), which change their roles depending on the type of work they are required to complete. This implies a powerful adaptation mechanism which eliminates the need to develop special honeybee types. Another important point is related to the decentralized and distributed approach the insects apply to their foraging. The lack of hierarchical decision making and central guidance in terms of coordinating moves, allows the honeybees able to work in harmony despite their large population or unfortunate events (e.g., an insect is damaged). The next important point made in chapter 3, is the adaptive approach met in natural foraging. Foragers are able to adjust their dancing and affect the recruitment of others. Additionally, they are able to compare different findings with prior knowledge, instead of selecting the one with, for example, the higher concentration of sugar. Finally, another important point of chapter 3 is the idea of grouping. Honeybees are able to form groups which work towards the same commodity and/or source of food. Foragers that carry pollen from the same flower adopt the same odour, which speeds up their communication and thus their foraging.

In chapter 4, BeeIP is described. The internal parts of the artificial hive (Entrance, BeeIPHive, and BeeIPQueue) as well as the types of control packets that are used by the protocol are presented. The chapter also describes the mechanism of routing maintenance (monitoring the quality of links and paths, the artificial honeybee dancing), data packet forwarding, and detecting link failures.

Chapters 5 and 6 present an extensive range of experimental tests, in which BeeIP is evaluated and compared to four state-of-the-art protocols. Five distinct sets of scenarios are used in such a way that they cover as much as possible of the network conditions that occur in real networking. They cover alterations in node mobility (i.e., changing pause times and node speeds), communication (i.e., number of nodes participating in the topology, number of sources and destinations), as well as terrain alterations (i.e., changing of terrain size). Additionally, a rich number of performance metrics are for the comparison of the five protocols including packet delivery radio, control overhead and average throughput. A summary of the observations is given in section 6.2.6.

In what follows, the initial research hypothesis and question are revisited and possible future directions of this research are discussed.

7.2 Revisiting research hypothesis and question

The initial research hypothesis, as presented in section 1.2.1, is:

Hypothesis: "*If the method used by a honeybee colony in order to evaluate the quality of a food source is a corporate consideration of a number of factors, which indicate different attributes of the food source as well as the foraging process in general (environmental conditions), then an artificial colony based on similar principles should be able to obtain analogous information, and use it to evaluate the quality of the paths between sources and destinations and control the number of transmissions that will use the paths in the future.*"

In order to provide credit to the initial research hypothesis, it is important to consider both the routing design and the observations of the experimental tests. In BeeIP, the evaluation of paths is designed to mimic the real behaviour of honeybees. That is, low-level information retrieved dynamically from other layers (media access control, physical, and application layers) is used to measure the quality of the intermediate links of a path between a source and a destination (formulas 4.1 and 4.3, in sections 4.6.1 and 4.6.3 respectively). In nature, foragers utilise their quality factors and set thresholds, which allows them to build an idea of how good or bad a foraging process with respect to a particular finding becomes over time. It is unclear what the precise values of those thresholds are (because of the different circumstances and combinations), but it has been empirically shown that they exist and they alter the behaviour of honeybees (von Frisch, 1967) while foraging or dancing back within their hive to recruit others. Combined together, the quality factors represent and describe the overall quality of the foraging process, e.g., the flying between the flower and the hive and the quality of the nectar found.

Based on this understanding, a similar approach has been taken in networks. BeeIP's quality factors are used to represent the quality of paths. The formula to measure an overall path quality takes into consideration the importance of each factor, as they all exist and play different roles in the data transmission. Then, the artificial foragers report their finding and with the use of regression analysis, improvement or deterioration over time can be detected. This methodology allows recruitment to take place, and controls the number of transmissions that will use a specific path in the future. The results of comparing BeeIP to other routing protocols show interesting and competitive results, as BeeIP is able to outperform the others in terms of packet delivery ratio and end-to-end delay. Therefore, the initial hypothesis is successfully validated.

The initial hypothesis allowed the formulation of the following research question:

Research Question: "*To what extent will a bee-inspired model that allows agents to discover and constantly measure and monitor the quality of paths in a MANET, based on low-level information obtained by the network, be able to adjust the future transmissions over those paths, in order to provide adaptive, decentralized and robust routing in comparison to other existing state-of-the-art approaches?*"

This research question is answered when considering chapters 5 and 6, where the methodology of the experimental tests and the comparison results are given and discussed. As illustrated, the proposed bee-inspired model is able to utilize agents in a decentralized way, in order to discover and constantly measure and monitor the quality of paths in a mobile ad hoc network. By modelling the special honeybee dance and the ability of the honeybees to select the most appropriate of the list of promising paths, packets are able to be transmitted over multiple routes over time. Therefore, adaptability is achieved when artificial honeybees adjust the recruitment process of each discovered path, according to the low-level up-to-date information which represents its quality.

Furthermore, the proposed design model achieves robust routing as is seen from the results of the simulation comparison between BeeIP (the implementation) and the four state-of-the-art routing protocols in the area of wireless ad hoc routing. As described in chapter 5, the simulation scenarios are designed in such a way that several realistic topologies, as well as movement and traffic patterns, are tested. What is observed is the potential of the protocol to adapt, scale up and efficiently provide routing solutions, outperforming the others in terms of packet delivery radio, end-to-end delay and average receiving throughput. The protocol can also perform better under stressful network conditions such as high traffic and mobility rates. When considering the network density, the proposed bee-inspired routing protocol is found to scale up much better within dense topologies, e.g., large number of nodes and high traffic.

7.3 Future research directions

There are several research directions which are considered in order to improve the design of the proposed bee-inspired routing protocol. Due to the time limits during the course of this research project, the completion of some aspects of the protocol has been limited. However, in this section, these aspects are identified and briefly discussed.

In chapter 4, where the design is described, it is mentioned that the artificial foragers are carrying real data and stay at the destination node as long as there are some data that need to go back to the source. In a TCP scenario, the artificial foragers are frequently utilized as the transport protocol is sending acknowledgement packets for each data packet being received. However, in an unreliable service with no acknowledgements such as UDP, there may not be a data packet available to drive the forager back unless the destination needs to send some data back to the source. A possible way to solve this problem is to apply some extra rules for such protocols that follow the honeybee swarming when a new honeybee queen and her followers are swarming in order to create a new hive. In Wedde et al. (2005a), such behaviour has been mapped to help UDP foragers travel back to the source as a group, once the difference between the incoming foragers from a destination and the outgoing foragers to the same destination reaches a threshold value. In BeeIP, this behaviour is incompatible with the principle of the protocol to constantly monitor the quality of the path. If forager packets return to the source in groups after some time interval (or another criterion which allows "idle" gaps to the traffic towards the source node), then the protocol stops being adaptive and in turn operates in a traditional best-effort mode of delivering packets to what it knows already– providing that there are foragers waiting in the routing repositories (refer to section 4.5 for the internal details). For this reason, the swarming technique has not been introduced in BeeIP. A potential way to address this issue, would be to force the foragers to return to the source, just as they would when carrying a data packet, but after shorter intervals to match a TCP behaviour. Moreover, the performance of this behaviour must be investigated in order to understand the impact it will have on the control overhead and throughput.

In section 4.6.2, a weighting system is found and presented by using the change of the mean squared error (MSE) method. Compared to sensitivity analysis (SA), and fuzzy curves (FC), MSE is the most straight-forward method to be used. However, in order to improve the weighting system and thus the optimality of the routing protocol, other methods need to be considered. In Sung (1998b), a comparison of these three methods is given and the author observes that both SA and FC methods perform very well in identifying the importance of each input, in relation to the output, even when a noisy input is present. Therefore, an interesting future research direction which will contribute to this work, would be to investigate the results of a different method to measure the inputs' ranking importance, and compare them in order to produce the optimal weighting system for the model.

In section 4.7.1, the metric of the best path selection is discussed. In the current design, the selection metric is chosen before the initialization of the ad hoc network, and stays static for as long as the network is used. An interesting direction for future research is to elaborate further the idea of dynamically switching the selection metric, according to the application needs. Cross-layering can be used for this and mark each communication session with a predefined type of behaviour. Such an investigation is expected to contribute towards the quality of service (QoS) (I.Jawhar & Wu, 2004) ability of the proposed design. What needs to be researched is the ability of the protocol to dynamically perform for each communication session and, in connection to the UDP acknowledgements discussed before, to offer an extra level of data transfer reliability at the network layer's level.

Furthermore, section 5.4.2 has pointed out an important research step that needs to be taken in the future. That is, the need to compare the proposed bee-inspired routing with other nature-inspired protocols and, in particular, bee-inspired protocols of the same area, namely BeeAdHoc. Unfortunately, due to the lack of availability of approved source code of these protocols for the ns-2 network simulator, a valid direct scientific comparison could not be achieved. Directly comparing to nature-inspired in addition to the state-of-the-art protocol is expected to give extra merit to the contributions of the new design, specifically to the field of applying swarm intelligence to computer networks.

Finally, the implementation of BeeIP for additional network simulators and also for real applications of collaborative agent-based systems (e.g., module for the Linux kernel to be utilized by Wi-Fi devices) will give an extra value to the understanding its performance. Interesting future work would also be to apply the proposed design to other types of networks, such as wireless sensor networks. Different simulators such as GloboSim/QualNet[1] or SensorSim (a sensor network extension for ns-2) (Park *et al.*, 2000) can be used to simulate sensor networks more efficiently.

7.4 Publications of the work in this thesis

The research work presented in this thesis has led to the following publications.

Paper 1. Abstract: "The field of robotics relies heavily on various technologies such as mechanical and electronic engineering, computing systems, and wireless communication. The latter plays a significant role in the area of mobile robotics by supporting remote interactions. An effective, fast, and reliable communication between homogeneous or heterogeneous robots, as well as the ability to adapt to the rapidly changing environmental conditions predicates the robots' success and completion of their tasks. In this paper we present our research position in the area of adaptive nature-inspired routing protocols for mobile ad hoc networks (MANETs). Our approach is based on the honeybee foraging behaviour and ability to find and exchange information about productive sources of food in a rapidly changed environment. We describe the research problem, present a brief review of the relative literature, and illustrate our future plan."

Paper 1. Citation: "Giagkos A., Wilson, M.S.: "A Cross-layer Design for Bee-Inspired Routing Protocols in MANETs". In Proceedings of Towards Autonomous Robotic Systems (TAROS) Conference. Kyriacou, T., et al. (eds.). TAROS 2009, pp. 25-32. Ulster University (2009)"

Paper 2. Abstract: "We introduce a new bee-inspired routing protocol for mobile ad hoc networks. Emphasis is given to the ability of bees to evaluate paths by considering several quality factors. In order to achieve similar behaviour in the networking environment, BeeIP is using cross-layering. Fetching parameters from the lower PHY and MAC layers to the core of the protocol, offers the artificial

[1]See section 5.2 for information regarding this network simulator.

bees the ability to make predictions of the link's future performance. The simulation results show that BeeIP achieves higher data delivery rates and less control overhead compared to DSDV, and has slightly better results to AODV, initializing less routing discovery processes."

Paper 2. Citation: "Giagkos, A., Wilson, M.S.: "BeeIP: Bee-Inspired protocol for routing in mobile ad-hoc networks". In Proceedings of the 11th International Conference on Simulation of Adaptive Behaviour - From Animals to Animats 11. Doncieux et al. (eds). SAB 2010, LNAI 6226, pp. 263-272, Springer-Verlag (2010)"

Appendix A

BeeIP default values

The following table A.1 lists all the default values of BeeIP's internal mechanisms, e.g., thresholds for the timers, honeybee dancing emulation, etc. As in most of other protocol's RFC reports, these variables may be set by the user's best guess, according to the network characteristics. Here, each variable is briefly explained.

NETWORK_DIAMETER is the maximum number of hops that can be found in the longest route within the network topology.

MULTI_PATH_NO is the number of alternative paths that are allowed to be found by each new scouting process. The larger the number, the longest the list of alternatives.

MAX_RECRUITS and FIRST_RECRUITS are used to define the maximum and the first number of unemployed foragers to be recruited for a path.

RHO_MINUS, RHO_PLUS and RHO_ADJ are used for the emulation of the honeybee dancing and the Pearson's r thresholds for negative and positive correlation between time and path quality. The RHO_ADJ is the constant number which is added or removed from tha foraging capacity, when either a strong positive or a strong negative correlation is found.

SCOUT_START_TTL, _RESEND, _STEP, and _TRIES are used to control the scouting process. A new scout packet starts with an initial _TTL and is resent every _RESED seconds. Each time the scout packet is resent, its TTL value is increased by a _STEP. This makes sure that the scout packets do not flood the network in order to find a destination close to the source node. However, after _TRIES number of resending a scout, the NETWORK_DIAMETER is used.

BROKEN_LINK_TIMEOUT, NEIGHBOURS_, SCOUTING_, RDATA_ are all used to control the timers' with respect to cleaning the neigbouring nodes', scouting and routing repositories.

QUEUE_PRUNE_TIMEOUT and QUEUE_MAX_LEN are used to control the BeeIP queues (this is not the MAC interface queue which is controlled by another protocol). QUEUE_MAX_LEN defines the number of packets that are allowed to stay in the queue simultaneously.

Variable Name	**Default Value**	**Value Type**
NETWORK_DIAMETER	35	integer (hops)
MULTI_PATH_NO	4	integer (paths)
MAX_RECRUITS	40	integer (pkts)
FIRST_RECRUITS	20	integer (pkts)
RHO_MINUS	-0.8	float (Pearson's r)
RHO_PLUS	0.8	float (Pearson's r)
RHO_ADJ	4	integer (pkts)
SCOUT_START_TTL	3	integer (hops)
SCOUT_TTL_RESEND	0.4	float (secs)
SCOUT_TTL_STEP	3	integer (hops)
SCOUT_MAX_TRIES	5	integer
BROKEN_LINK_TIMEOUT	3	integer (secs)
NEIGHBOURS_TIMEOUT	10	integer (secs)
SCOUTING_TIMEOUT	6	integer (secs)
RDATA_TIMEOUT	9	integer (secs)
QUEUE_PRUNE_TIMEOUT	4	integer (secs)
QUEUE_MAX_LEN	40	integer (pkts)

Table A.1: BeeIP default values.

Appendix B

Normalization to different scales

The following formula is used to convert two numbers of a scale A to B, to a different scale C to D:

$$y = C + (x - A) * (D - C)/(B - A) \tag{B.1}$$

Simple practical illustration:

If $x = 5$ in a scale of 1 to 10, convert x to a number y in a scale of 1 to 100.
Answer:

Let, $A = 1$, $B = 10$, $C = 1$, $D = 100$. Using formula B.1:

$$\begin{aligned} y &= C + (x - A) * (D - C)/(B - A) \\ &= 1 + (5 - 1) * (100 - 1)/(10 - 1) \\ &= 50 \end{aligned} \tag{B.2}$$

Notice that if $x = A$, then $y = C$, and if $x = B$, $y = D$ as required.

Appendix C

Processing BeeIP control packets

Algorithms 7, 8 and 9 illustrate the logic for processing the BeeIP control packets at any BeeIP node.

Algorithm 7 Processing a scout packet

Input: A scout packet *pkt*

Output: A “loop”, “propagate scout” or a “reply with ack_scout” result

```
1: procedure RECVSCOUT(pkt)
2:     if pkt.Originator is me then
3:         return loop
4:     end if
5:     if pkt.Dest is me then
6:         if pkt's scouting is unknown then
7:             CREATESCOUTINGDATA(pkt)
8:         else
9:             count ← 0
10:            // Lookup acknowledged routing entries of previous scoutings
11:            // with the same hive and me as the flower (multipath):
12:            for all rData ∈ RoutingRepository do
13:                if rData.ACK = 1 then
14:                    count ← count + 1
15:                end if
16:            end for
17:            if count ≥ MULTI_PATH_NO then
18:                return loop
19:            end if
20:        end if
21:        return reply with ack_scout
22:    end if
23:    // I am a path node and I might want to propagate this.
24:    if pkt.TTL > 0 then
25:        if pkt's scouting is unknown then
26:            CREATESCOUTINGDATA(pkt)
27:            return propagate scout
28:        end if
29:    end if
30:    return loop
31: end procedure
```

Algorithm 8 Processing an ack_scout packet

Input: An ack_scout packet *pkt*

Output: An “ack_scout at hive” or “propagate ack_scout” result

```
1: procedure recvAckScout($pkt$)
2:     $sData \leftarrow$ getScoutingData($pkt$)
3:     $sData.ACK \leftarrow 1$
4:     updateNeighbouringData($pkt$)
5:     if $pkt.Src$ is me then
6:         return ack_scout at hive
7:     end if
8:     return propagate ack_scout
9: end procedure
```

Algorithm 9 Processing a forager or an ack_forager packet

Input: A forager or an ack_forager packet *pkt*

Output: A forager “at path”, or “at hive”, or “at flower”

```
1: procedure recvForager($pkt$)
2:     if $pkt.Type$ is ack_forager then
3:         updateNeighbouringData($pkt$)
4:     end if
5:     $rData \leftarrow$ getRoutingData($pkt.PID$)                ▷ Path ID
6:     if $rData$ not NULL then
7:         $rData.TSLastUsed \leftarrow$ max($rData.TSLastUsed$, $pkt.TS$)
8:         $rData.ACK \leftarrow 1$
9:         if $rData.Type$ is FLOWER and $pkt.Dest$ is me then
10:            return forager at flower
11:        end if
12:        if $rData.Type$ is HIVE and $pkt.Dest$ is me then
13:            $rData.TSLastReturned \leftarrow$ CURRENT_TIME
14:            $rData.ForagersIn \leftarrow rData.ForagersIn + 1$
15:            $rData.ForagersOut \leftarrow rData.ForagersOut - 1$
16:            return forager at hive
17:        end if
18:    end if
19:    return forager at path
20: end procedure
```

Appendix D

Working examples of packet flow

Two working examples of data packet flow are presented in this Appendix. Figure D.1 illustrates the data flow between node A and node C using the path $A \leftrightarrow B \leftrightarrow C$, starting by a successful scouting process.

Figure D.2 illustrates an example where the link between A and C breaks due to intermediate node B being disconnected. A new scouting process is initialized when node A detects the broken path and an alternative path $A \leftrightarrow D \leftrightarrow C$ is discovered and used.

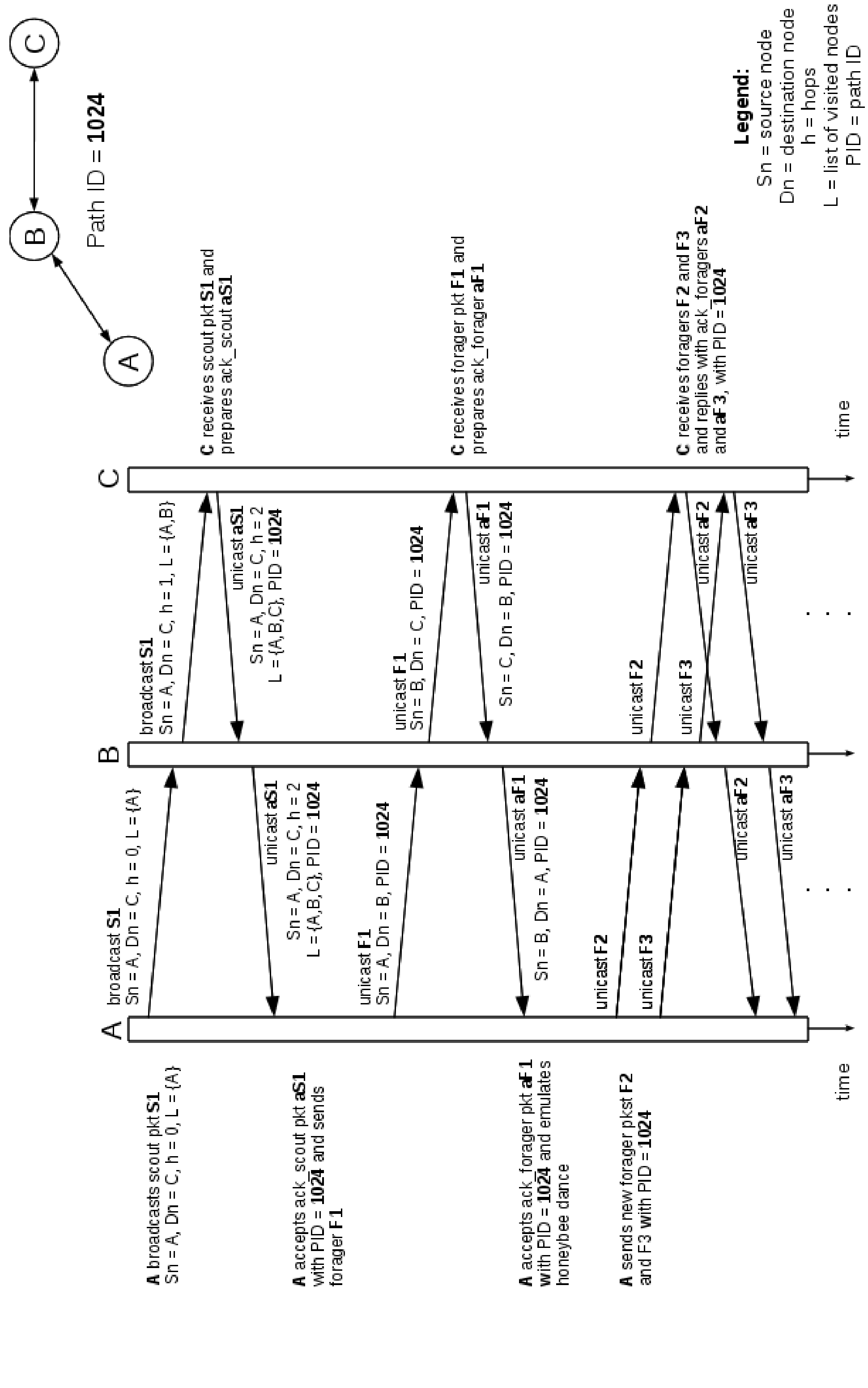

A
B
C
Path ID = 1024
A
B
C
time
time
A broadcasts scout pkt S1
Sn = A, Dn = C, h = 0, L = {A}
A accepts ack_scout pkt aS1 with PID = 1024 and sends forager F1
A accepts ack_forager pkt aF1 with PID = 1024 and emulates honeybee dance
A sends new forager pkst F2 and F3 with PID = 1024
broadcast S1
Sn = A, Dn = C, h = 0, L = {A}
unicast aS1
Sn = A, Dn = C, h = 2
L = {A,B,C}, PID = 1024
unicast F1
Sn = A, Dn = B, PID = 1024
unicast aF1
Sn = B, Dn = A, PID = 1024
unicast F2
unicast F3
unicast aF2
unicast aF3
broadcast S1
Sn = A, Dn = C, h = 1, L = {A,B}
unicast aS1
Sn = A, Dn = C, h = 2
L = {A,B,C}, PID = 1024
unicast F1
Sn = B, Dn = C, PID = 1024
unicast aF1
Sn = C, Dn = B, PID = 1024
unicast F2
unicast F3
unicast aF2
unicast aF3
C receives scout pkt S1 and prepares ack_scout aS1
C receives forager pkt F1 and prepares ack_forager aF1
C receives foragers F2 and F3 and replies with ack_foragers aF2 and aF3, with PID = 1024
Legend:
Sn = source node
Dn = destination node
h = hops
L = list of visited nodes
PID = path ID

Figure D.1: A data packet flow diagram which captures the scouting and foraging activity on a working path of 3 nodes *A*, *B* and *C* of a communication session. The source node *A* requests a route to destination node *C*.

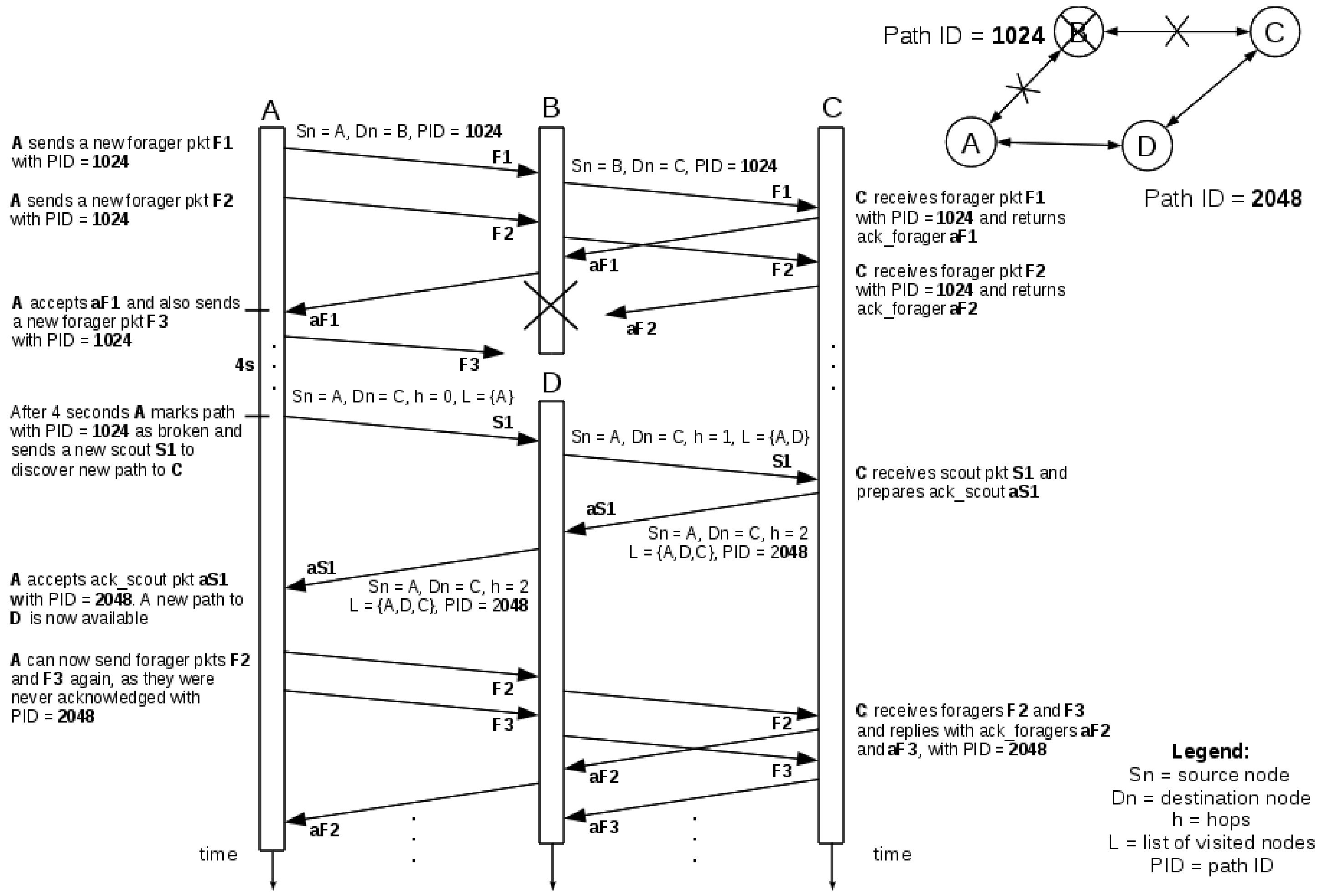

Path ID = 1024
Path ID = 2048
A
B
C
D
A sends a new forager pkt F1 with PID = 1024
A sends a new forager pkt F2 with PID = 1024
A accepts aF1 and also sends a new forager pkt F3 with PID = 1024
4s
After 4 seconds A marks path with PID = 1024 as broken and sends a new scout S1 to discover new path to C
A accepts ack_scout pkt aS1 with PID = 2048. A new path to D is now available
A can now send forager pkts F2 and F3 again, as they were never acknowledged with PID = 2048
Sn = A, Dn = B, PID = 1024
F1
F2
aF1
F3
Sn = B, Dn = C, PID = 1024
F1
aF1
F2
aF2
C receives forager pkt F1 with PID = 1024 and returns ack_forager aF1
C receives forager pkt F2 with PID = 1024 and returns ack_forager aF2
Sn = A, Dn = C, h = 0, L = {A}
S1
Sn = A, Dn = C, h = 1, L = {A,D}
S1
C receives scout pkt S1 and prepares ack_scout aS1
aS1
Sn = A, Dn = C, h = 2 L = {A,D,C}, PID = 2048
aS1
Sn = A, Dn = C, h = 2 L = {A,D,C}, PID = 2048
F2
F3
F2
F3
aF2
aF3
aF2
C receives foragers F2 and F3 and replies with ack_foragers aF2 and aF3, with PID = 2048
time
time
Legend:
Sn = source node
Dn = destination node
h = hops
L = list of visited nodes
PID = path ID

Figure D.2: A data packet flow diagram which captures the foraging activity when the path $A \rightarrow B \rightarrow C$ becomes unavailable. A new scouting is initialized and a new path $A \rightarrow D \rightarrow C$ is discovered.

Appendix E

Supplementary MLP training results (MSE change)

	All present	Missing parameters				
		S.Strength	Energy	Speed	Qu delay	Tx delay
Dataset 0:	0.001287	0.003616	0.003423	0.001363	0.001834	0.001369
Dataset 1:	0.001602	0.004472	0.004642	0.001224	0.001485	0.001840
Dataset 2:	0.000963	0.004057	0.003967	0.001066	0.001388	0.001799
Dataset 3:	0.001254	0.002248	0.002251	0.001106	0.001260	0.001495
Dataset 4:	0.002680	0.008018	0.008559	0.002972	0.003124	0.004695
Dataset 5:	0.001123	0.003472	0.003866	0.001338	0.002037	0.001670
Dataset 6:	0.002139	0.009302	0.009188	0.002821	0.004725	0.003362
Dataset 7:	0.002336	0.008906	0.009119	0.002404	0.002549	0.002765
Dataset 8:	0.002280	0.008671	0.007945	0.002796	0.005410	0.004509
Dataset 9:	0.002462	0.007472	0.007266	0.002875	0.002929	0.004226
Avg MSE:	0.001813	0.006023	0.006023	0.001997	0.002674	0.002773
Difference:		0.004211	0.004210	0.000184	0.000862	0.000960
Weight %:		**40.39**	**40.38**	**1.76**	**8.26**	**9.21**

Table E.1: Supplementary results obtained by measuring the change of the mean squared error (MSE) while training the MLP with a data set of 5000 patterns.

	All present	Missing parameters				
		S.Strength	Energy	Speed	Qu delay	Tx delay
Dataset 0:	0.000860	0.002675	0.003301	0.001043	0.001174	0.001191
Dataset 1:	0.002250	0.004819	0.005185	0.002277	0.004096	0.003420
Dataset 2:	0.001385	0.003178	0.002969	0.001498	0.001641	0.001675
Dataset 3:	0.001318	0.003393	0.003460	0.001282	0.001426	0.001970
Dataset 4:	0.001586	0.004739	0.004952	0.001891	0.001831	0.002813
Dataset 5:	0.001105	0.003117	0.003393	0.001144	0.001582	0.001753
Dataset 6:	0.001176	0.003366	0.004169	0.001212	0.001441	0.001819
Dataset 7:	0.001325	0.003165	0.003226	0.001350	0.001486	0.002250
Dataset 8:	0.000727	0.002515	0.002609	0.000928	0.000920	0.000970
Dataset 9:	0.000870	0.002422	0.002448	0.000699	0.001242	0.001234
Avg MSE:	0.001260	0.003339	0.003571	0.001332	0.001684	0.001910
Difference:		0.002079	0.002311	0.000072	0.000424	0.000649
Weight %:		**37.56**	**41.75**	**1.3**	**7.66**	**11.73**

Table E.2: Supplementary results obtained by measuring the change of the mean squared error (MSE) while training the MLP with a data set of 20000 patterns.

References

Adjih, C., Baccelli, E., & Jacquet, P. 2003a. Link state routing in wireless ad-hoc networks. *Pages 1274–1279 of: Proceedings of the 2003 IEEE conference on Military communications - Volume II.* MILCOM'03. IEEE Computer Society.

Adjih, Cedric, Clausen, Thomas, Laouiti, Anis, Mühlethaler, Paul, & Raffo, Daniele. 2003b. Securing the OLSR protocol. *Pages 25–27 of: In 2nd IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2003), Mahdia.*

Aguayo, Daniel, Bicket, John, Biswas, Sanjit, Judd, Glenn, & Morris, Robert. 2004. Link-level measurements from an 802.11b mesh network. *Pages 121–132 of: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications.* SIGCOMM '04. ACM.

Akyildiz, I.F., Su, Weilian, Sankarasubramaniam, Y., & Cayirci, E. 2002. A survey on sensor networks. *Communications Magazine, IEEE*, **40**(8), 102–114.

Alena, R.L. Alena, & Lee, C. 2005 (March). Adaptive bio-inspired wireless network routing for planetary surface exploration. *Pages 1438–1443 of: Aerospace Conference, 2005 IEEE.*

Amiri, M. 2010. *Evaluation of Lifetime Bounds of Wireless Sensor Networks.* Arxiv preprint arXiv:1011.2103 vol. abs/1011.2.

Arun Kumar B. R., Reddy, Lokanatha C., & Hiremath, Prakash S. 2008. Performance Comparison of Wireless Mobile Ad-Hoc Network Routing

Protocols. *International Journal of Computer Science and Network Security (JCSNS)*, **8**(6), 337–343.

Aschenbruck, Nils, Gerhards-Padilla, Elmar, & Martini, Peter. 2008. A survey on mobility models for performance analysis in tactical mobile networks. *Journal of Telecommunications and Information Technology*, **2**(2/2008), 54–61.

Bai, Fan, & Helmy, Ahmed. 2004. *A Survey of Mobility Modeling and Analysis in Wireless Adhoc Networks*. Kluwer Academic Publishers. Chap. 1.

Balaji, N. Prasanna, Sreenivasulu, U., N, Anjaneyulu, & Priyadarishini, N. Indira. 2011. Wireless Mobile Ad-hoc Shortest Path Routing in Delay Tolerant Networks. *International Journal of Multidisciplinary Sciences and Engineering*, **2**(6), 10–15.

Baldo, Nicola, Maguolo, Federico, Miozzo, Marco, Rossi, Michele, & Zorzi, Michele. 2007. ns2-MIRACLE: a modular framework for multi-technology and cross-layer support in network simulator 2. *Pages 1–8 of: ValueTools: Proc. of the International Conference on Performance evaluation methodologies and tools*. ICST.

Banerjee, Suman, & Misra, Archan. 2002. *Adapting Transmission Power for Optimal Energy Reliable Multi-hop Wireless Communication*. Tech. rept. WiOpt 03. Multi-hop Wireless Communication, Wireless Optimization Workshop, Sophia-Antipolis.

Bardwell, J. 2002. *Converting Signal Strength Percentage to dBm Values*. WildPackets.

Beach, Aaron, Gartrell, Mike, Panichsakul, Saroch, Chen, Li, Ching, Chao-Kai, & Han, Richard. 2008. *X-Layer: An Experimental Implementation of a Cross-Layer network Protocol Stack for Wireless Sensor Networks*. Tech. rept. CU-CS-1051-08. University of Colorado at Boulder.

Begg, L., Liu, W., Pawlikowski, K., Perera, S., & Sirisena, H. 2006 (Feb.). *Survey of Simulators of Next Generation Networks for Studying*

Service Availability and Resilience. Tech. rept. TRCOSC 05/06. Department of Computer Science & Software Engineering, University of Canterbury, Christchurch, New Zealand.

Beijar, N. 2002. *Zone Routing Protocol (ZRP).* Licentiate course on Telecommunications Technology, Ad-Hoc Networking.

Bellman, R. 1958. On a Routing Problem. *Quarterly of Applied Mathematics*, **16**, 87–90.

Bharghavan, Vaduvur, Demers, Alan, Shenker, Scott, & Zhang, Lixia. 1994. MACAW: A media access protocol for wireless LAN's. *SIGCOMM Comput. Commun. Rev.*, **24**(4), 212–225.

Biradar, S. R., Majumder, Koushik, Sarkar, Subir Kumar, & C, Puttamadappa. 2010. Performance Evaluation and Comparison of AODV and AOMDV. *International Journal on Computer Science and Engineering (IJCSE)*, **2**(2), 373–377.

Blake, S, & Pullin, A. 2007. The Effect of Node Density on Routing Protocol Performance in Mobile Ad-Hoc Networks. *The Convergence of Telecommunications, Networking and Broadcasting*, Jun.

Board, IEEE-SA Standards. 2003. Part 11 : Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications High-speed Physical Layer in the 5 GHz Band. *Control*, **1999**(June).

Bonabeau, Eric, Dorigo, Marco, & Theraulaz, Guy. 2000. Inspiration for Optimization from social insect behaviour. *Nature*, **406**(Jul.), 39–42.

Boppana, Rejandra V., & Konduru, Satyadeva P. 2001. *An Adaptive Distance Vector Routing Algorithm for Mobile, Ad Hoc Networks.* IEEE InfoCom.

Borgia, Eleonora, Conti, Marco, & Delmastro, Franca. 2004. *MobileMAN: Design, Integration, and Experimentation of Cross-Layer Mobile Multihop Ad Hoc Networks.* IEEE.

Boukerche, Azzedine. 2009. *Algorithms and Protocols for Wireless and Mobile Ad hoc Networks.* Wiley.

Breed, Michael D., Robinson, Gene E., & Robert E. Page, Jr. 1990. Division of Labor during Honey Bee Colony Defense. *Behavioral Ecology and Sociobiology*, **27**(6), 395–401.

Brenner, By Pablo. 1997. A Technical Tutorial on the IEEE 802.11 Protocol Director of Engineering. *Portal*, 147–152.

Broch, Josh, Maltz, David A., Johnson, David B., Hu, Yih-Chun, & Jetcheva, Jorjeta. 1998a. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Pages 85–97 of: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. MobiCom '98. ACM.

Broch, Josh, Maltz, David A., Johnson, David B., Hu, Yih-Chun, & Jetcheva, Jorjeta. 1998b. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Pages 85–97 of: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. MobiCom '98. ACM.

Camp, Tracy, Boleng, Jeff, & Davies, Vanessa. 2002. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communication & Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, **2**(5), 483–502.

Cao, Lijuan, Dahlberg, T., & Wang, Yu. 2007 (Apr.). Performance Evaluation of Energy Efficient Ad Hoc Routing Protocols. *Pages 306–313 of: Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa.*

Caro, Gianni Di, & Dorigo, M. 1998. *AntNet: Distributed Stigmergetic Control for Communications Networks*. Vol. 9. Journal of Artificial Intelligence Research (JAIR). Pages 317–365.

Chakeres, I., & Perkins, C. 2012. *Dynamic MANET On-demand (DYMO) Routing*. IETF Internet Draft http://tools.ietf.org/id/draft-ietf-manet-dymo-22.txt.

Chauhan, R.K., & Chopra, Ashish. 2010. Power Optimization In Mobile Ad Hoc Network. *Global Journal of Computer Science and Technology*, **10 Issue 4**(Jun.), 92–96.

Chen, Lijun, Low, Steven H., Chiang, Mung, & Doyle, John C. 2006. *Cross-layer Congestion Control, Routing and Scheduling Design in Ad Hoc Wireless Networks*. IEEE.

Chen, T., & Gerla, M. 1998. *Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks*. IEEE ICC '98. Pages 171–175.

Chen, Yebin, Shu, Jian, Zhang, Sheng, Liu, Linlan, & Sun, Limin. 2009 (May). Data Fusion in Wireless Sensor Networks. *Pages 504–509 of: Second International Symposium on Commerce and Security (ISECS '09)*, vol. 2.

Chetoui, Yassine, Bouabdallah, Nizar, & Othman, Jalel Ben. 2007. Resolving the Unfairness Limitations of the IEEE 802.11 DCF. *IJCSNS International Journal of Computer Science and Network Security*, **7**(1), 298–303.

Chiang, C. C. 1997. *Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel*. Proc IEEE SICON '97. Pages 197–211.

Chlamtac, I., Conti, M., & Liu, J. 2003. Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, **1**(1), 13–64.

Clausen, T., & Jacquet, P. 2003. *Optimized Link State Routing Protocol (OLSR)*. RFC 3626.

Conti, M., Maselli, G., Turi, G., & Giordano, S. 2004. Cross-layering in mobile ad hoc network design. *Computer*, **37**(2), 48–51.

Corson, S., & Macker, J. 1993. *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*. RFC 2501.

Dargie, W., & Poellabauer, C. 2010. *Fundamentals of wireless sensor networks: theory and practice*. John Wiley and Sons.

De, Swades, Das, Sajal K., Wu, Hongyi, & Qiao, Chunming. 2002. Trigger-based distributed QoS routing in mobile ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, **6**(3), 22–35.

Decastro, Leandro N., Zuben, Fernando J. Von, Pub, Idea Group, & Castro, Leandro N. De. 2004. *Recent Developments In Biologically Inspired Computing*. IGI Publishing.

Deepa, S., & Nawaz, D.M. Kadhar. 2010. MANET Routing Protocols with Varying Densities and Dynamic Mobility Patterns. *IJCS Special Issue on Mobile Ad-Hoc Networks*, 124–128.

Dhillon, S.S., Arbona, X., & Van Mieghem, P. 2007. Ant Routing in Mobile Ad Hoc Networks. *Page 67 of: Third International Conference on Networking and Services, 2007. ICNS*. IEEE.

Di Caro, G. A. 2004 (Nov.). *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. Ph.D. thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles.

Di Caro, Gianni. 2003 (Dec.). *Analysis of simulation environments for mobile ad hoc networks*. Tech. rept. 24-03. IDSIA, Lugano, (Switzerland).

Di Caro, Gianni, Ducatella, Frederick, & Gambardella, Luca Maria. 2005. *Special Issue on Self-organisation in Mobile Networking: AntHocNet: an adaptive nature-insipred algorithm for routing in mobile ad hoc networks*. AEIT.

Dorigo, Marco, & Stutzle, Thomas. 2004. *Ant Colony Optimization*. MIT Press, Cambridge.

Dornhaus, Anna, Klügl, Franziska, Oechslein, Christoph, Puppe, Frank, & Chittka, Lars. 2006. Benefits of recruitment in honey bees: effects of ecology and colony size in an individual-based model. *Behavioral Ecology*, **17**(3), 336–344.

Dousse, O., Thiran, P., & Hasler, M. 2002. Connectivity in ad-hoc and hybrid networks. *Pages 1079–1088 of: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2.

Dousse, O., Baccelli, F., & Thiran, P. 2003 (Mar.). Impact of interferences on connectivity in ad hoc networks. *Pages 1724–1733 of: INFOCOM*

2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 3.

Ducatelle, Frederick. 2007. *Adaptive Routing in Ad Hoc Wireless Multi-hop Networks.* Ph.D. thesis, Università della Svizzera italiana Faculty of Informatics.

Ducatelle, Frederick, Di Caro, Gianni, & Gambardella, Luca Maria. 2005. 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), 169–184.

Duflos, S., Grand, G. L., Diallo, A. A., Chaudet, C., Hecker, A., Balducelli, C., Flentge, F., Schwaegerl, C., & Seifert, O. 2006 (Sept.). *List of Available and Suitable Simulation Components.* Tech. rept. D1.3.2. Ecole Nationale Supieure des Tommunications (ENST).

Eenennaam, E M Van. 2009. A Survey of Propagation Models used in Vehicular Ad hoc Network (VANET) Research. *Technical Report*, **67**(1), 132–147.

El-Sayed, H., Bazan, O., Qureshi, U., & Jaseemuddin, M. 2005 (Sept.). Performance Evaluation of TCP in Mobile Ad-Hoc Networks. *In: Proceedings of the Second International Conference on Innovations in Information Technology (IIT'05).* IIT '05.

Eltahir, I.K. 2007 (Aug.). The Impact of Different Radio Propagation Models for Mobile Ad hoc NETworks (MANET) in Urban Area Environment. *Page 30 of: The 2nd International Conference on Wireless Broadband and Ultra Wideband Communications, 2007. AusWireless 2007.*

Farooq, Muddassar. 2006. *From the Wisdom of the Hive to Intelligent Routing in Telecommunication Networks: A Step towards Intelligent Network Management through Natural Engineering.* Ph.D. thesis, der Universität Dortmund.

Farooq, Muddassar. 2009. *Bee-Inspired Protocol Engineering.* Springer.

Fleischer, Mark. 2005. *Foundations of Swarm Intelligence: From Principles to Practice.* Swarming Network Enabled C4ISR.

Forouzan, Behrouz A. 2005. *TCP/IP Protocol Suite.* third edn. McGraw-Hill.

Free, J. B. 1977. *The Social Organization of Honeybees.* Edward Arnold.

Friginal, Jesús, de Andrés, David, Ruiz, Juan-Carlos, & Gil, Pedro. 2011. Towards benchmarking routing protocols in wireless mesh networks. *Ad Hoc Netw.*, **9**(8), 1374–1388.

Fu, Zhenghua, Zerfos, Petros, Luo, Haiyun, Lu, Songwu, Zhang, Lixia, & Gerla‘, Mario. 2003. The Impact of Multihop Wireless Channel on TCP Throughput and Loss. *IEEE INFOCOM*, Mar.

Ge, Y., Lamont, L., & Villasenor, L. 2004 (Aug.). Improving Scalability of Heterogeneous Wireless Networks with Hierarchical OLSR. *In: In the OLSR Interop and Workshop.*

Gerla, Mario, Tang, Ken, & Bagrodia, Rajive. 1999. TCP Performance in Wireless Multi-hop Networks. *Pages 41–50 of: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications.* WMCSA '99. IEEE Computer Society.

Gharavi, H., & Ban, K. 2004. *Dynamic Adjustment Packet Control for Video Communications over Ad-hoc Networks.* IEEE.

Giagkos, A., & Wilson, M. S. 2009 (Aug.). A Cross-layer Design for Bee-Inspired Routing Protocols in MANETs. *Pages 25–32 of: Proc. TAROS Towards Autonomous Robotic Systems (TAROS'09).*

Giagkos, A., & Wilson, M. S. 2010. BeeIP: Bee-Inspired Protocol for Routing in Mobile Ad-Hoc Networks. *Pages 263–272 of:* Doncieux et al. (ed), *Proc. SAB Simulation of Adaptive Behavior, From Animals to Animats 11 (SAB'10).* LNAI, vol. 6226/2010. Springer-Verlag.

GloMoSim. 2012. *Global Mobile Information Systems Simulation Library.* Online: http://pcl.cs.ucla.edu/projects/glomosim/. (accessed 16/04/2012).

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley.

Gould, James L. 1975. Communication of distance information by honey bees. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, **104**, 161–173.

GOULD, JAMES L., HENEREY, MICHAEL, & MACLEOD, MICHAEL C. 1970. Communication of Direction by the Honey Bee. *Science*, **169**(3945), 544–554.

GUNES, MESUT, SORGES, UDO, & BOUAZIZI, IMED. 2002. *ARA - The Ant-Colony Based Routing Algorithms for MANETs.* IWAHN.

GURNEY, KEVIN. 1997. *An introduction to Neural networks.* CRC PRESS.

HANBALI, AHMAD AL, ALTMAN, EITAN, & NAIN, PHILIPPE. 2005. A survey of TCP over ad hoc networks. *IEEE Communications Surveys and Tutorials*, **7**(1-4), 22–36.

HASSAN, YASSER KAMAL, EL-AZIZ, MOHAMED HASHIM ABD, & EL-RADI, AHMED SAFWAT ABD. 2010. Performance Evaluation of Mobility Speed over MANET Routing Protocols. *International Journal of Network Security*, **11**(3), 128–138.

HAYKIN, S. 1994. *Neural Networks. A comprehensive foundation.* Prentice Hall.

HAYKIN, SIMON. 2005. Cognitive radio: brain-empowered wireless communications. *IEEE Journal on Selected Areas in Communications*, **23**(2), 201–220.

HAYZELDEN, ALEX L. G., & BIGHAM, JOHN. 1999. Agent technology in communications systems: an overview. *Knowl. Eng. Rev.*, **14**(4), 341–375.

HE, C. 2002. *Destination-Sequenced Distance Vector (DSDV) Protocol.* Networking Laboratory, Helsinki University of Technology.

HEDRICK, CHARLES. 1988. *Routing Information Protocol.* RFC 1058.

HEINZELMAN, W.B., CHANDRAKASAN, A.P., & BALAKRISHNAN, H. 2002. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on Wireless Communications*, **1**(4), 660–670.

HOFMANN-WELLENHOF, B., LICHTENEGGER, H., & COLLINS, J. 2001. *Global Positioning System: Theory and Practice.* 5th edn. Springer.

Hogie, J. L., Bouvry, P., & Guinand, F. 2005. An Overview of MANETs Simulation. *Pages 81–101 of: Proc. of 1st International Workshop on Methods and Tools for Coordinating Concurrent, Distributed and Mobile Systems (MTCoord).* LNCS, Elsevier.

Ibnkahla, Mohamed (ed). 2009. *Adaptation and Cross Layer Design in Wireless Networks.* CRC Press.

I.Jawhar, & Wu, J. 2004. *Quality of Service Routing in Mobile Ad Hoc Networks.* Resource Management and Wireless Networking, Kluwer Academic Publishers.

Issariyakul, Teerawat, & Hossain, Ekram. 2009. *Introduction to Network Simulator NS2.*

Johnson, D. B., & Maltz, D. A. 1996. *Dynamic Source Routing in Ad-Hoc Wireless Networks.* Mobile Computing, T. Imielinski and H. Korth, Eds., Kluwer. Pages 153–81.

Jubin, J., & Tornow, J.D. 1987. The DARPA packet radio network protocols. *Proceedings of the IEEE*, **75**(1), 21–32.

Jun, Jangeun, Peddabachagari, Pushkin, & Sichitiu, Mihail. 2003. Theoretical Maximum Throughput of IEEE 802.11 and its Applications. *Pages 249+ of: NCA '03: Proceedings of the Second IEEE International Symposium on Network Computing and Applications.* IEEE Computer Society.

Karaboga, Dervis, & Akay, Bahriye. 2009. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*, **31**(1), 61–85.

Karn, P. 1990 (Sept.). MACA: A new channel access method for packet radio. *Pages 134–140 of: Proceedings of the 9th Computer Networking Conference.*

Kim, Kyu-Han, & Shin, Kang G. 2006. On accurate measurement of link quality in multi-hop wireless mesh networks. *Pages 38–49 of: Proceedings of the 12th annual international conference on Mobile computing and networking.* MobiCom '06. ACM.

Kleinrock, L., & Stevens, K. 1971. *Fisheye: A Lenslike Computer Display Transformation.* Technical report, UCLA, Computer Science Department.

Kleinrock, L., & Tobagi, F. A. 1975. Packet switching in radio channels: Part i - carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Transactions on Communications*, **23**(2), 1400–1416.

Ko, Young-Bae, & Vaidya, Nitin H. 2000. Location-aided routing (LAR) in mobile ad hoc networks. *Wirel. Netw.*, **6**(4), 307–321.

Köksal, Murat Miran. 2008. *A Survey of Network Simulators Supporting Wireless Networks.* Middle East Technical University Ankara, Turkey.

Kostoulas, Dionysios, Aldunate, Roberto, Mora, Feniosky Pena, & Lakhera, Sanyogita. 2008. A nature-inspired decentralized trust model to reduce information unreliability in complex disaster relief operations. *Advanced Engineering Informatics*, **22**(1), 45 – 58.

Kotz, David, Newport, Calvin, Gray, Robert S., Liu, Jason, Yuan, Yougu, & Elliott, Chip. 2004. Experimental evaluation of wireless simulation assumptions. *Pages 78–82 of: MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems.* ACM.

Kozat, Ulas C., Koutsopoulos, Iordanis, & Tassiulas, Leandros. 2004. *A Framework for Cross-Layer Design of Energy-efficient Communication with QoS Provisioning in Multi-hop Wireless Networks.* IEEE INFOCOM 2004.

Kuan, C.-M., & Hornik, K. 1991. Convergence of learning algorithms with constant learning rates. *IEEE Transactions on Neural Networks*, **2**(5), 484–489.

Kumar, Yugal, & Kumar, Dharmender. 2011. Article: Parametric Analysis of Nature Inspired Optimization Techniques. *International Journal of Computer Applications*, **32**(3), 42–49.

Lau, Clifford G. Y. 1992. *Neural networks: theoretical foundations and analysis.* IEEE Press.

Layuan, L., Peiyan, Y., & Chunlin, L. 2007. Performance evaluation and simulations of routing protocols in ad hoc networks. *Computer Communications*, **30**(8), 1890–1998.

Lie, Changling, & Kaiser, Jrg. 2005. *A Survey of Mobile Ad Hoc network Routing Protocols.* Tech. rept. Series Nr. 2003-08. University of Ulm.

Lin, Yinghua, & Cunningham, G.A., III. 1995. A new approach to fuzzy-neural system modeling. *IEEE Transactions on Fuzzy Systems*, **3**(2), 190–198.

Lougheed, K., & Rekhter, Y. 1990 (Jun.). *A border gateway protocol.* RFC 1163.

Lucic, P., & Teodorović, D. 2001. Bee system: modeling combinatorial optimization transportation engineering problems by swarm intelligence. *In Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis.*, 441–445.

Lucic, P., & Teodorović, D. 2002. Transportation modeling: an artificial life approach. *Pages 216–223 of: In Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence, 2002. (ICTAI 2002).*

Lucic, P., & Teodorovic, D. 2003. Computing with Bees: Attacking Complex Transportation Engineering Problems. *International Journal on Artificial Intelligence Tools*, **12**(3), 375–394.

Maleki, Morteza, Dantu, Karthik, & Pedram, Massoud. 2002. Power-aware Source Routing Protocol for Mobile Ad Hoc Networks. *Pages 72–75 of: Proceedings of the 2002 International Symposium on Low Power Electronics and Design.*

Mali, Baldev Ram, & Barwar, N.C. 2011. Effect of Mobility on Performance of MANET Routing Protocols under Different Traffic Patterns. *Special issues on IP Multimedia Communications*, Oct., 19–24.

Maqbool, Bilal, M.A.Peer, & S.M.K.Quadri. 2011. Towards the Benchmarking of Routing Protocols for Adhoc Wireless Networks. *International Journal of Computer Science and Technology*, **2**(1), 28–35.

Marina, Mahesh K., & Das, Samir R. 2006. Ad hoc on-demand multipath distance vector routing. *Wireless Communications and Mobile Computing*, **6**(7), 969–988.

MARINA, M.K., & DAS, S.R. 2001 (Nov.). On-demand multipath distance vector routing in ad hoc networks. *Pages 14–23 of: Ninth International Conference on Network Protocols, 2001.*

MARTINEZ, FRANCISCO J., TOH, CHAI KEONG, CANO, JUAN-CARLOS, CALAFATE, CARLOS T., & MANZONI, PIETRO. 2011. A survey and comparative study of simulators for vehicular ad hoc networks (VANETs). *Wireless Communications and Mobile Computing*, **11**(7), 813–828.

MARWAHA, SHIVANAJAY, THAM, CHEN KHONG, & SRINIVASAN, DIPTI. 2002. *Mobile Agents based Routing Protocol for Mobile Ad Hoc Networks.* IEEE.

MCCLELLAND, JAMES L., & RUMELHART, DAVID E. 1986. *Explorations in the Microstructure of Cognition: Foundations.* Series: Parallel Distributed Processing, vol. 1. MIT Press.

MINAR, NELSON, KRAMER, KWINDLA HULTMAN, & MAES, PATTIE. 1999. *Cooperating Mobile Agents for Dynamic Network Routing.* Springer-Verlag. Chap. 12.

MUELLER, ALEXANDER. 2007. *Efficient Wireless MAC Protocols- Analyzing Quality of Service in Wireless Networks.* VDM Verlag.

NGUYEN, DANG, & MINET, PASCALE. 2007. Scalability of the OLSR Protocol with the Fish Eye Extension. *Pages 88–94 of: Proceedings of the Sixth International Conference on Networking.* ICN '07. IEEE Computer Society.

NISSEN, STEFFEN. 2003. *Implementation of a fast artificial neural network library (fann).* Technical report.

NS-2. 2012. *The Network Simulator - ns-2.* Online: http://www.isi.edu/nsnam/ns/. (accessed 16/04/2012).

OGUNJEMILUA, K, DAVIES, J.N., GROUT, V, & PICKING, R. 2009. *An Investigation into Signal Strength of 802.11n WLAN.* Centre for Applied Internet Research, Glyndŵr University dan University of Wales, Wrexham.

OLAGUE, GUSTAVO, PUENTE, CESAR, & EVOVISIN, PROYECTO. 2006. C.: The honeybee search algorithm for three-dimensional reconstruction. *Pages 427–437 of: of Lecture Notes in Computer Science.* Springer.

OMNeT++ Community. 2012. *OMNeT++*. Online: http://www.omnetpp.org/. (accessed 16/04/2012).

OPNET Technologies, Inc. 2012. *OPNET Modeler*. Online: http://www.opnet.com/solutions/network_rd/modeler.html. (accessed 16/04/2012).

OPNET Technologies, Inc. 2012. *Optimized Network Engineering Tools*. Online: http://www.opnet.com/. (accessed 16/04/2012).

Parissidis, G., Lenders, V., May, M., & Plattner, B. 2006. Multi-path routing protocols in wireless mobile ad hoc networks: A quantitative comparison. *Pages 313–326 of: NEW2AN 2006*. LNCS 4003.

Park, Sung, Savvides, Andreas, & Srivastava, Mani B. 2000. SensorSim: a simulation framework for sensor networks. *Pages 104–111 of: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*. ACM.

Park, V., & Corson, S. 2001. *Temporally-Ordered Routing Algorithm (TORA)*. Online: http://tools.ietf.org/html/draft-ietf-manet-tora-spec-04/. Internet Draft, IETF MANET Working Group, (accessed 16/04/2012).

Pei, G., Gerla, M., & Chen, T. 2000. *Fisheye State Routing in Mobile Ad Hoc Networks*. ICDCS Workshop on Wireless Networks and Mobile Computing.

Perez-Neira, Ana I., & Campalans, Marc Realp. 2009. *Cross-Layer Resource Allocation in Wireless Communications*. Elsevier.

Perins, C. E., & Bhagwat, P. 1994. *Highly dynamic destination-sequenced distance vector (DSDV) for mobile computers*. ACM SIGCOMM '94.

Perkins, C. E., Royer, E. M., & Das, S. R. 1999. *Ad hoc on demand distance vector routing*. RFC 3561.

Postel, Jon. 1981 (Sep.). *Internet Protocol - DARPA Inernet Programm, Protocol Specification*. RFC 791.

Prowler. 2012. *Prowler: Probabilistic Wireless Network Simulator*. Online: http://www.isis.vanderbilt.edu/projects/nest/prowler/. (accessed 05/04/2012).

QualNet. 2012. *QualNet Scalable Network Technologies.* Online: http://www.scalable-networks.com/content/products/qualnet/. (accessed 16/04/2012).

Radha, Sankararajan, & Shanmugavel, Subbaiah. 2001. Effect of exponential and normal distributed speed and position for mobiles in ad hoc networks. *Modeling and Design of Wireless Networks*, **4531**(1), 293–300.

Rahman, Abdul Hadi Abd, & Zukarnain, Zuriati Ahmad. 2009. Performance Comparison of AODV, DSDV and I-DSDV Routing Protocols in Mobile Ad Hoc Networks. *European Journal of Scientific Research*, **31**(4), 566–576.

Randles, M., Lamb, D., & Taleb-Bendiab, A. 2009 (Dec.). Experiments with Honeybee Foraging Inspired Load Balancing. *Pages 240 –247 of: Second International Conference on Developments in eSystems Engineering (DESE).*

Read, Timothy R.C., & Cressie, Noel A.C. 1988. *Goodness-of-fit statistics for discrete multivariate data.* Series in Statistics. Springer.

Ram Murthy, C. Siva, & Manoj, B. S. 2004. *Ad Hoc Wireless Networks Architectures and Protocols.* Prentice Hall.

De Couto, Douglas S. J. 2004 (Jun.). *High-Throughput Routing for Multi-Hop Wireless Networks.* Ph.D. thesis, Massachusetts Institute of Technology.

Monarch Project. 2012. *The Rice University Monarch Project: Mobile Networking Architectures.* Online: http://www.monarch.cs.rice.edu/. (accessed 16/04/2012).

Rosen, E., Viswanathan, A., & Callon, R. 2001. *Multiprotocol Label Switching Architecture.* RFC 3031.

Roth, Martin, & Wicker, Stephen. 2003. *Termite: Ad-Hoc Networking with Stigmergy.* IEEE.

Roth, Martin, & Wicker, Stephen. 2005. *Termite: A Swarm Intelligent Routing Algorithm for Mobile Wireless Ad-Hoc Networks.* Cornell University.

Russell, Stuart, & Norvig, Peter. 2003. *Artificial Intelligence: A Modern Approach*. 2nd edition edn. Prentice-Hall, Englewood Cliffs, NJ.

Saleem, M., & Farooq, M. 2007a (Sept.). A framework for empirical evaluation of nature inspired routing protocols for wireless sensor networks. *Pages 751–758 of: IEEE Congress on Evolutionary Computation, 2007. CEC 2007.*

Saleem, Muhammad, & Farooq, Muddassar. 2007b. *BeeSensor: A Bee-Inspired Power Aware Routing Protocol for Wireless Sensor Networks.* Springer-Verlag.

Saltelli, A., Chan, K., & Scott, E. M. 2000. *Sensitivity Analysis.* Wiley.

Sarkar, T K, Medouri, A, & Salazar-Palma, M. 2003. A survey of various propagation models for mobile communication. *IEEE Antennas and Propagation Magazine*, **45**(3), 51–82.

Sass, Paul. 1999. Communications networks for the force XXI digitized battlefield. *Mob. Netw. Appl.*, **4**(3), 139–155.

Schoonderwoerd, Ruud, Bruten, Janet L., Holland, Owen E., & Rothkrantz, Leon J. M. 1996. Ant-based load balancing in telecommunications networks. *Adapt. Behav.*, **5**(2), 169–207.

Seeley, T D, & Mikheyev, A S. 2003. Reproductive decisions by honey bee colonies: tuning investment in male production in relation to success in energy acquisition. *Insectes Sociaux*, **50**(2), 134–138.

Seeley, T. D., & Towne, W. F. 1992. Tactics of dance choice in honey bees: do foragers compare dances? *Behavioral Ecology and Sociobiology*, **30**, 59–69.

Seeley, Thomas D. 1995. *The wisdom of the hive: The Social Physiology of Honey Bee Colonies.* Harvard University Press.

Selvakennedy, S., & Sinnappan, S. 2005 (Nov.). The time-controlled clustering algorithm for optimized data dissemination in wireless sensor network. *Pages 2 pp. –510 of: The IEEE Conference on Local Computer Networks, 2005. 30th Anniversary.*

Selvakennedy, S., Sinnappan, S., & Shang, Yi. 2006. *T-ANT: A Nature-Inspired Data Gathering Protocol for Wireless Sensor Networks.* IEEE.

Shah, R. C., & Rabaey, J.M. 2002. *Energy aware routing for low energy ad hoc sensor networks.* In Proceeding of the IEEE Wireless Communications and Networking Conference, Orlando, FL. Pages 350–355.

Shannon, Robert E. 1998. Introduction to the art and science of simulation. *Pages 7–14 of: Proceedings of the 30th conference on Winter simulation.* WSC '98. IEEE Computer Society Press.

Sharman, A. Kumar, & Bhatia, N. 2011. Behavioral study of MANET routing protocols by using NS-2. *IJCEM International Journal of Computational Engineering & Management*, **12**(Apr.), 100–104.

Singh, Tanu Preet, Singh, R. K., & Vats, Jayant. 2011. Article: Routing Protocols in Ad Hoc Networks: A Review. *International Journal of Computer Applications*, **25**(4), 30–35. Published by Foundation of Computer Science, New York, USA.

Srivastava, V., & Motani, M. 2005. Cross-layer design: a survey and the road ahead. *Communications Magazine, IEEE*, **43**(12), 112–119.

Stuedi, P., & Alonso, G. 2007. Wireless ad hoc VoIP. *In: in MNCNA 2007: Proceedings of the Workshop on Middleware for Next-Generation Converge Networks and Applications.* ACM.

Subramanian, D., Druschel, P., & Chen, J. 1997. Ants and reinforcement learning: A case study in routing in dynamic networks. *Pages 832–838 of: In Proceedings of IJCAI-97, International Joint Conference on Artificial Intelligence.*

Suhonen, J., Kuorilehto, M., Jannikainen, M., & Hamalainen, T. D. 2006. *Cost-aware dynamic routing protocol for wireless sensor networks design and prototype experiments.* In Proceeding of the 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Helsinki, Finland. Pages 1–5.

Sung, A.H. 1998a. Ranking importance of input parameters of neural networks. *Expert Systems with Applications*, **15**(34), 405–411.

Sung, A.H. 1998b (May). Ranking input importance in neural network modeling of engineering problems. *Pages 316–321 of: The 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence*, vol. 1.

Swingler, K. 1996. *Applying Neural Networks: A Practical Guide.* Academic Press Limited.

Takai, Mineo, Martin, Jay, & Bagrodia, Rajive. 2001. Effects of Wireless Physical Layer Modeling in Mobile Ad Hoc Networks. *Pages 87–94 of: in MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing.* ACM Press.

Taneja, Kavita, & Patel, R B. 1972. An Overview of Mobile Ad hoc Networks: Challenges and Future. *Inst Of Computer Tech Business Management, Dept Of Computer Eng, Engineering College Mullana Haryana, India*, 1–4.

Taneja, Sunil, & Kush, Ashwani. 2010. A Survey of Routing Protocols in Mobile Ad Hoc networks. *International Journal of Innovation, Management and Technology.*

Tantubay, Neeraj, Gautam, Dinesh Ratan, & Dhariwal, Mukesh Kumar. 2011. A Review of Power Conservation in Wireless Mobile Adhoc Network (MANET). *IJCSI*, **8 Issue 4**(1).

Teodorović, Dusan, & Orco, Mauro Dell. 2005. Bee colony optimization - a cooperative learning approach to complex transportation problems. *Pages 51–60 of: In Proceedings of 10th EWGT Meeting and 16th Mini EURO Conference.*

Teodorović, Dusan, Davidović, Tatjana, & Šelmić, Milica. 2011. Bee Colony Optimization: The Applications Survey. *ACM Transactions on Computational Logic.* (under publiacion).

Theraulaz, G., & Bonabeau, E. 1999. A brief history of stigmergy. *Artificial Life, Special Issue on Stigmergy*, **5**(2), 97–116.

Toh, C. K. 2002. *Ad Hoc Mobile Wireless Networks: Protocols and systems.* Prentice-Hall. Chap. 6, pages 80–95.

TRENN, S. 2008. Multilayer Perceptrons: Approximation Order and Necessary Number of Hidden Units. *IEEE Transactions on Neural Networks*, **19**(5), 836 –844.

TUTEJA, ASMA, GUJRAL, RAJNEESH, & THALIA, SUNIL. 2010 (Jun.). Comparative Performance Analysis of DSDV, AODV and DSR Routing Protocols in MANET Using NS2. *Pages 330–333 of: Conference on Advances in Computer Engineering (ACE), 2010 International.*

UPADHAYAYA, SHUCHITA, & GANDHI, CHARU. 2011. Node Disjoint Multipath Routing Considering Link and Node Stability protocol: A characteristic Evaluation. *IJCSI International Journal of Computer Science Issues*, **7, Issue 1**(2), 18–25.

VAISHAMPAYAN, R., & GARCIA-LUNA-ACEVES, J.J. 2004 (Oct.). Efficient and robust multicast routing in mobile ad hoc networks. *Pages 304–313 of: IEEE International Conference on Mobile Ad-hoc and Sensor Systems.*

VAN HOESEL, LODEWIJK, NIEBERG, TIM, WU, JIAN, & HAVINGA, PAUL J. M. 2004. *Prolonging the Lifetime of Wireless Sensor Networks By Cross-Layer Interaction.* IEEE.

VARATHARAJAN, SARVESH, JABBAR, ABDUL, MOHAMMAD, ASIFUDDIN, NAIDU, RAMYA, ROHRER, JUSTIN P., UPADHYAY, PIYUSH, & STERBENZ, JAMES P.G. 2012. *Cross-Layer Framework for the ns-2 Simulator.* Online: https://wiki.ittc.ku.edu/resilinets/Cross-Layer_Framework_for_the_ns-2_Simulator/. (accessed 16/04/2012).

VASILIOU, A., & ECONOMIDES, A. 2007. Game-based learning using MANETs. *Pages 154–159 of: in N. Mastorakis and ph (eds.): Proceedings of the 4th WSEA/ASME International Conference on Engineering Education (EE'07).* WSEAS Press.

VERVERIDIS, CHRISTOPHER N., & POLYZOS, GEORGE C. 2005 (Jun.). Impact of Node Mobility and Network Density on Service Availability in MANETs. *In: Proc. 5th IST Mobile Summit.*

VIDHYAPRIYA, R., & VANATHI, P.T. 2007 (Feb.). Energy Aware Routing for Wireless Sensor Networks. *Pages 545–550 of: International Conference on Signal Processing, Communications and Networking, 2007. ICSCN '07.*

VITTORI, K., TALBOT, G., GAUTRAIS, J., FOURCASSIÉ, V., ARAUJO, A. F. R., & THERAULAZ, G. 2006. Path efficiency of ant foraging trails in an artificial network. *Journal of Theoretical Biology*, **239**, 507–515.

VON FRISCH, KARL. 1967. *The Dance Language and Orientation of Bees*. Oxford University Press.

WANG, JIANPING, OSAGIE, ESEOSA, THULASIRAMAN, PARIMALA, & THULASIRAM, RUPPA K. 2009. HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Networks*, **7**(4), 690 – 705.

WANG, YU, & WU, JIE. 2006. *Label Routing Protocol: A New Cross-layer Protocol for Multi-hop Ad hoc Wireless Networks*. TROUBADOR.

WEDDE, HORST, FAROOQ, MUDDASSAR, & ZHANG, YUE. 2004a. BeeHive: An Efficient Fault-Tolerant Routing Algorithm Inspired by Honey Bee Behavior. *Pages 83–94 of:* DORIGO, MARCO, BIRATTARI, MAURO, BLUM, CHRISTIAN, GAMBARDELLA, LUCA MARIA, MONDADA, FRANCESCO, & STÜTZLE, THOMAS (eds), *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Brussels, Belgium, September 5 - 8, 2004, Proceedings*. Lecture Notes in Computer Science, vol. 3172. Springer.

WEDDE, HORST, SENGE, SEBASTIAN, LEHNHOFF, SEBASTIAN, KNOBLOCH, FABIAN, LOHMANN, TIM, NIEHAGE, ROBERT, & STRÄBER, MANUEL. 2010. Bee Inspired Online Vehicle Routing in Large Traffic Systems. *ADAPTIVE 2010: The Second International Conference on Adaptive and Self-Adaptive Systems and Applications*, 78–83.

WEDDE, HORST F., FAROOQ, MUDDASSAR, & ZHANG, YUE. 2004b. BeeHive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. *Pages 83–94 of: Lecture Notes in Computer Science*. Springer.

WEDDE, HORST F., FAROOQ, MUDDASSAR, PANNENBAECKER, THORSTEN, VOGEL, BJOERN, MUELLER, CHRISTIAN, METH, JOHANNES, & JERuschkat, Rene. 2005a. BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. *Pages 153–160 of: Proceedings of the 2005 conference on Genetic and evolutionary computation.* GECCO '05. ACM.

Wedde, Horst F., Farooq, Muddassar, Pannenbaecker, Thorsten, Vogel, Bjoern, Mueller, Christian, Meth, Johannes, & Jeruschkat, Rene. 2005b. Beeadhoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. *Pages 153–160 of: In GECCO.*

Wedde, Horst F., Farooq, Muddassar, Fischer, Julia, Kowalski, Martin, Langhans, Michael, Range, Niko, Schletter, Cornelia, Tarak, Recai, Tchatcheu, Michel, Timm, Constantin, Volmering, Friedrich, Werner, Sebastian, Wang, Kai, & Zhai, Bin. 2005c. *BeeAdHoc Efficient/Secure/Scalable Routing Framework für AdHoc Netze.* Tech. rept. PG460, LSIII. School of Computer Sience, University of Dortmund.

Xiao, Hannan, Zhang, Ying, Malcolm, James A., Christianson, Bruce, & Chua, Kee Chaing. 2010. Modelling and Analysis of TCP Performance in Wireless Multihop Networks. *Wireless Sensor Network*, **2**(7), 493–503.

Xu, Dahai, Chiang, Mung, & Rexford, J. 2011. Link-State Routing With Hop-by-Hop Forwarding Can Achieve Optimal Traffic Engineering. *IEEE/ACM Transactions on Networking*, **19**(6), 1717–1730.

Xu, K., Gerla, M., & Bae, S. 2003. Effectiveness of RTS/CTS Handshake in IEEE 802.11 based Ad Hoc Networks. *Ad Hoc Networks*, **1**(1), 107–123.

Xu, Ning. 2002. A Survey of Sensor Network Applications. *IEEE Communications Magazine*, **40**.

Yoshida, E., Yamamoto, M., Arai, T., Ota, J., & Kurabayashi, D. 1995 (May). A design method of local communication area in multiple mobile robot system. *Pages 2567–2572 of: Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 3.

Zainuddin, Zarita, & Pauline, Ong. 2008. Function approximation using artificial neural networks. *WSEAS Trans. Math.*, **7**(6), 333–338.

Zhang, G.P. 2000. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, **30**(4), 451–462.

Zhang, Y. 2012. *Routing modeling application simulation environment.* Online: http://www2.parc.com/isl/groups/era/nest/Rmase/. (accessed 05/04/2012).

Zhou, Gang, He, Tian, Krishnamurthy, Sudha, & Stankovic, John A. 2004. Impact of Radio Irregularity on Wireless Sensor Networks. *In: MobiSys.* USENIX.

Zhu, Hua, Li, Ming, Chlamtac, I., & Prabhakaran, B. 2004. A survey of quality of service in IEEE 802.11 networks. *Wireless Communications, IEEE*, **11**(4), 6–14.

Zimmermann, H. 1980. OSI Reference Model–The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, **28**(4), 425–432.

Index