MODELING AND SIMULATION OF A ROUTING PROTOCOL FOR AD HOC NETWORKS COMBINING QUEUING NETWORK ANALYSIS AND ANT COLONY ALGORITHMS

Dissertation

zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)

vorgelegt dem Fachbereich 5,
Institut für Informatik und Wirtschaftsinformatik
der Universität Duisburg-Essen (Campus Essen)

Von

TAREK HELMI ABD EL-NABI ALI AHMED

geboren in Port-Said (Ägypten)

April 2005

Datum der mündlichen Prüfung: 25. April 2005

Erstgutachter: Prof. Dr. Bruno Müller-Clostermann

Zweitgutachter: Prof. Dr.-Ing. Erwin P. Rathgeb

ACKNOWLEDGMENTS

First of all and the greatest important, I would like to thank my GOD for all his blessings without which nothing of my work would have been done.

I would like to take the opportunity to thank people who guided and supported me during this period of my study. Without their contributions, this research would not have been possible.

I owe great thanks to my Ph.D. supervisor, Prof. Dr. Bruno Müller-Clostermann for his guidance, advice and support throughout the entire thesis. I feel I have really been lucky to be working with someone like him. This thesis could not have been written without the supervision and support of him.

As my supervisor, he provided me with the encouragement and freedom to pursue my own ideas. I think in the last four years, Dr. Müller-Clostermann taught me how to do the research. In other words, he taught me how to make gold out of stones. Besides, he was always patient and helpful whenever his guidance and assistance were needed. As an international student, I felt a certain difficulty in technical writing. Dr. Müller-Clostermann also contributed much of his valuable time to help me improve my writing skills. Dr. Müller-Clostermann also helps me to attend different conferences inside and outside Germany. Therefore, I do want to show my deep appreciation to him.

Also, I'd like to thank Prof. Dr. Erwin P. Rathgeb and Prof. Dr. Michael Goedicke as members of my examination committee.

I am also grateful to Dr. Michael Sampels who put my legs on the first step in my work. He succeeded in opening my eyes focused on so many optimization techniques, which I used in my work. For that, I'm deeply thankful.

I'd like to thank all my colleagues in the department of computer science in Essen, Hesham K. Mohamed, Corinna Flüs, Andreas Pillekeit and Stefan Hanenberg with whom I have had and still have a wonderful time. They provided me with a very friendly atmosphere and ensured that working at the department was always fun.

Finally, I'd like to express my great appreciation to my parents, my brothers and sister for their continuous support during all these years.

Tarek H. Ahmed

DEDICATION

I dedicate this work to my parents,
my grandfather, grandmother,
my brothers, my sister
and my uncles

All good work is done the way ants do things: Little by little

Lafcadio Hearn

ABSTRACT

The field of *Mobile Ad hoc Networks* (MANETs) has gained an important part of the interest of researchers and become very popular in last few years. MANETs can operate without fixed infrastructure and can survive rapid changes in the network topology. They can be studied formally as graphs in which the set of edges varies in time. The main method for evaluating the performance of MANETs is simulation. Our thesis presents a new adaptive and dynamic routing algorithm for MANETs inspired by the *Ant Colony Optimization* (ACO) algorithms in combination with network delay analysis. Ant colony optimization algorithms have all been inspired by a specific foraging behavior of ant colonies which are able to find, if not the shortest, at least a very good path connecting the colony's nest with a source of food.

Our evaluation of MANETs is based on the evaluation of the mean End-to-End delay to send a packet from source to destination node through a MANET. We evaluated the mean End-to-End delay as one of the most important performance evaluation metrics in computer networks.

Finally, we evaluate our proposed ant algorithm by a comparative study with respect to one of the famous On-Demand (reactive) routing protocols called *Ad hoc On-Demand Distance Vector* (AODV) protocol. The evaluation shows that, the ant algorithm provides a better performance by reducing the mean End-to-End delay than the AODV algorithm.

We investigated various simulation scenarios with different node density and pause times. Our new algorithm gives good results under certain conditions such as, increasing the pause time and decreasing node density. The scenarios that are applied for evaluating our routing algorithm have the following assumptions: 2-D rectangular area, no obstacles, bi-directional links, fixed number of nodes operate for the whole simulation time and nodes movements are performed according to the *Random Waypoint Mobility* (RWM) or the *Boundless Simulation Area Mobility* (BSAM) model.

KEYWORDS: Ant Colony Optimization (ACO), Mobile Ad hoc Network (MANET), Queuing Network Analysis, Routing Algorithms, Mobility Models, Hybrid Simulation.

GLOSSARY

ABAM	On-Demand Associativity-Based Multicast
ABC	Ant Based Control
ABR	Associativity Based Routing protocol
ACO	Ant Colony Optimization
ACS	Ant Colony System
ADMR	Adaptive Demand-Driven Multicast Routing protocol
AMRIS	Ad hoc Multicast Routing protocol utilizing Increasing id-numbers
AMRoute	Ad hoc Multicast Routing Protocol
ANN	Artificial Neural Network
ANSim	Ad-hoc Network Simulation
AODV	Ad hoc On-Demand Distance Vector routing protocol
ARA	Ant-Colony Based Routing
ARH	Ants Routing with routing History
ARHnr	Ants Routing with routing History and no return rule
AS	Ant System
ASR	Adaptive Swarm-based Routing
ATSP	Asymmetric Traveling Salesman Problem
BANT	Backward Ant
BRP	Bordercast Resolution Protocol
BSAM	Boundless Simulation Area Mobility model
BSR	Backup Source Routing protocol
CAMP	Core-Assisted Mesh Protocol
CBM	Content Based Multicast
CBRP	Cluster Based Routing Protocol
CEDAR	Core Extraction Distributed Ad hoc Routing
CGSR	Clusterhead Gateway Switch Routing algorithm
DBF	Distributed Bellman-Ford routing protocol
DDM	Differential Destination Multicast
DDR	Distributed Dynamic Routing algorithm

DREAM	Distance Routing Effect Algorithm for Mobility
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing protocol
DSRFLOW	Flow State in the Dynamic Source Routing protocol
DSR-MB	Simple Protocol for Multicast and Broadcast using DSR
DTDV	Highly Dynamic Destination-Sequenced Distance Vector routing protocol
EA	Evolutionary Algorithm
EAS	Elitist strategy Ant System
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolution Strategies
FANT	Forward Ant
FGMP	Forwarding Group Multicast Protocol
FIFO	First-In-First-Out policy
FORP	Flow Oriented Routing Protocol
FSR	Fisheye State Routing protocol
GA	Genetic Algorithms
GeoGRID	Geographical GRID
GeoTORA	Geographical TORA
GH	Greedy Heuristics
GLS	(Grid) Geographic Location Service
GP	Global Position
GPL	Global Position-Less
GPS	Global Positioning System
GPSAL	Global Positioning System Ant-Like Routing Algorithm
GSR	Global State Routing algorithm
HARP	Hybrid Ad Hoc Routing Protocol
HSLS	Hazy Sighted Link State routing protocol
HSR	Hierarchical State Routing protocol
IARP	Intrazone Routing Protocol
ILS	Iterated Local Search
ISAIAH	Infra-Structure Aodv for Infrastructured Ad Hoc networks
ITEF	Internet Engineering Task Force

JSP	Job-shop Scheduling Problem
LAM	Lightweight Adaptive Multicast protocol
LAN	Local Area Network
LANMAR	Landmark Routing Protocol for Large Scale Networks
LAR	Location-Aided Routing protocol
LBM	Location Based Multicast
LCA	Linked Cluster Architecture
LMR	Lightweight Mobile Routing protocol
LUNAR	Lightweight Underlay Network Ad hoc Routing
MABR	Mobile Ants Based Routing
MAC	Medium Access Control
MANET	Mobile Ad hoc Networks
MAODV	Multicast Ad hoc On-Demand Distance Vector routing
MC	Monte Carlo simulation
MCEDAR	Multicast CEDAR
MMAS	MAX–MIN Ant Systems
MMBDP	Mobile Mesh Border Discovery Protocol
MMLDP	Mobile Mesh Link Discovery Protocol
MMRP	Mobile Mesh Routing Protocol
MN	Mobile Node
MRGR	Mesh-Based Geocast Routing
MZR	Multicast Zone Routing protocol
ODMRP	On-Demand Multicast Routing Protocol
OLSR	Optimized Link State Routing Protocol
PAMAS	Power Aware Multi Access Protocol with Signaling Ad Hoc Networks
PAN	Personal Area Network
PARO	Power-Aware Routing Optimization Protocol
PERA	Probabilistic Emergent Routing Algorithm for Mobile Ad Hoc Networks
PSO	Particle Swarm Optimization
QAP	Quadratic Assignment Problem
RDMAR	Relative-Distance Micro-discovery Ad hoc Routing protocol
RWM	Random Waypoint Mobility model
SA	Simulated Annealing

SI	Swarm Intelligence
SO	Self-Organization
SPF	Straight Packet Forwarding
SRMP	Source Routing-based Multicast Protocol
SSR	Signal Stability Routing protocol
STAR	Source Tree Adaptive routing protocol
TAP	Topology Abstracting Protocol
TBRPF	Topology Broadcast based on Reverse-Path Forwarding routing protocol
TORA	Temporally Ordered Routing Algorithm
TS	Tabu Search
TSP	Traveling Salesman Problem
TTL	Time-To-Live
UTC	Universal Time Coordinate
VoIP	Voice over Internet Protocol
WLAN	Wireless Local Area Networks
WRP	Wireless Routing Protocol
ZHLS	Zone-based Hierarchical Link State routing algorithm
ZRP	Zone Routing Protocol

LIST OF FIGURES

LIST OF TABLES

CONTENTS

Chapter 3: Mobile Ad Hoc Networks

Chapter 4: Ant Colony Optimization

Chapter
1

INTRODUCTION

1.1 Background and Motivation

Nature has been a source of inspiration for developing meta-heuristics to deal with a variety of problems. These meta-heuristics are not bound to solve a single problem but represent a general mechanism or framework for solving a whole class of optimization problems.

In 1943, McCulloch and Pitts began to model one of the most direct mappings of biological learning process to a computer algorithm [McCulloch and Pitts, 1943]. This field is called *Artificial Neural Networks* (ANNs) which are based on understanding neurology and modeled to simulate neurons and synapses [Haykin, 1998]. These models made several assumptions about how neurons work. There are many applications for using neural networks such as, machine learning, cognitive science, prediction and pattern matching [Mostafa-Sami and Tarek, 1997]. ANNs have also some applications in combinatorial optimization, started by the proposition of the Hopfield Network for the *Traveling Salesman Problem* (TSP) in [Hopfield and Tank, 1985].

In 1975, other meta-heuristic algorithms inspired by nature -which have been widely researched since their proposal by [Holland, 1975]- are called *Evolutionary Algorithms* (EAs). These evolutionary alogrithms are a class of direct, probabilistic search and optimization alogrithms discovered from the model of organic evolution. The main representatives of this computational paradigm are *Genetic Algorithms* (GAs), *Evolution Strategies* (ESs) and *Evolutionary Programming* (EP), which were developed independently of each other [Thomas, 1996]. EAs are based on the classic principal of “survival of the fittest“ [Darwin and Wallace, 1858] where, in a gene pool the individuals corresponding to the fittest genes

have better chances for passing their genetic information on to future generations. The desired result of this process is an individual combination of the best characteristics of all its ancestors or taking the mutation of a gene into consideration, something even better. In the algorithm, a number of initial solutions are treated as genes, which are crossed with one another, with a bias toward better solutions being selected for the crossover, to creat offspring that inherits characteristics from all involved parents. The new generation of solutions keeps evolving, hopefully toward the area of good solutions which contains the optimal solution. For keeping a multitude of solutions, EAs are capable of exploring several promising areas of search space at once, if diversity and overall fitness of the population are well balanced. The theoretical results for certain types of EA show that it converges toward the optimum, when given enough time. A detailed introduction to EAs is given in [Goldberg, 1989], [Michalewicz, 1996]. There are many different possible applications of EAs. Any problem that has a large search domain could be suitably tackled by EAs. One such example is optimization of circular networks by GA [Tarek and Sampels, 2001].

In 1983, the world of combinatorial optimization was shattered by the appearance of a paper [Kirkpatrick et al., 1983] in which the authors were describing a new heuristic approach called *Simulated Annealing* (SA). It is commonly said to be the oldest among the metaheuristics and surely one of the first algorithms that had an explicit strategy to avoid local minima. The origins of the algorithm are based on the observation that the physical shape of a metal's crystals formed by slow cooling of a super saturated solution is crystalline, but amorphous when it results from fast cooling. This is due to the fact that in a slowly cooled metal solution, the individual atoms have the time necessary to reach the position with the lowest possible energy level, which is achived in a crystalline structure. The algorithmic exploitation of this phenomenon, first undertaken by [Metropolis et al., 1953], proceeds as follows; given a starting solution and temperature, the solution is altered a little bit and reevaluated. If the new solution is better, it definitely becomes the starting point for the next iteration. If the solution is worse, it replaces the old solution only with a certain probability correlated to the temperature, which slowly decreases over time. This has the effect that, at the beginning of the algorithm, nearly every new solution is adopted, while over time it becomes more and more likely that only better solutions are accepted, restricting the algorithm to hill-climbing. It has been shown in [Mitra et al., 1986] that, given a sufficiently slow cooling schedule, SA is guaranteed to converge to the optimal solution. It could be shown to converge to an optimal solution of a combinatorial problem, even though in infinite computing time. Based on analogy with statistical mechanics, SA can be interpreted as a form

of controlled random walk in the space of feasible solutions. The emergence of SA indicated that one could look for other ways to tackle combinatorial optimization problems and encourage the interest of the research community. In the following years, many other new approaches, mostly based on analogies with natural phenomena, were proposed such as *Swarm Intelligence* and together with some older ones, such as *Genetic Algorithms* [Holland, 1975], they gained an increasing popularity. Now collectively known under the name of *Meta-Heuristics* (a term originally coined by Glover in 1986), these methods have become over the last fifteen years the leading edge of heuristic approaches for solving combinatorial optimization problems.

A relatively new field in terms of its application to combinatorial optimization problems is *Swarm Intelligence* (SI). The term was first used in [Beni, 1988, Beni and Wang, 1989a,b] to describe the coordination of robots. Since then, two main interpretations of SI have been used for deriving methods to solve optimization problems. On one hand, *Particle Swarm Optimization* (PSO), which is evolutionary computation technique inspired by the social behavior of bird flocks or fish schools, has been proposed [Kennedy et al., 2001].

In PSO, population of solutions evolve by moving through the solution space, with good solutions acting as attractors. On the other hand, SI is defined to include any scheme which employs the swarm mechanics derived from the behavior of social insects, particulary ants, for solving optimization problems [Bonabeau et al., 1999]. The concept of *Ant Algorithms* was first introduced in [Dorigo, 1992] and since that time, it has been applied to both theoretical and practical optimization problems with great success. The performance exhibited by ant algorithms and the possibility of adaptation to new problems make the study of this field very worthwhile.

Ant algorithms are an iterative, probabilistic meta-heuristic for finding solutions to combinatorial optimization problems. They are based on the foraging mechanism employed by real ants attempting to find a short path from their nest to a food source. While foraging, the ants communicate indirectly via pheromone, which they use to mark their respective paths and which attracts other ants. In the ant algorithm, artificial ants use virtual pheromone to update their path through the decision graph, i.e. the path that reflects which alternative an ant chooses at certain points. Ants of the later iterations use the pheromone marks of previous good ants as a means of orientation when constructing their own solutions, which ultimately result in focusing the ants on promising parts of the search space.

In sometimes, a problem might be dynamic in nature, changing over time and requiring the algorithm to keep track of the occurring modifications and continually reoptimize in order to

be able to present a valid, good solution at all times. Ant algorithms have a number of attractive features, including adaptation, robustness and decentralized nature, which are well suited for routing in modern communication networks such as *Mobile Ad hoc Networks* (MANETs).

This thesis introduces a new approach for routing in MANETs, which is based on ant algorithm in combination with queuing network analysis.

1.2 Research Goal

This thesis presents a new dynamic and adaptive routing algorithm for MANETs inspired by the ant colony paradigm in combination with network delay analysis. Ant colony algorithms have been thoroughly investigated in the past for problems such as the *Traveling Salesman* problem and even the routing problem in communication networks. Our main goals in this thesis are:

- Evaluate the End-to-End delay to send a packet from source to destination node through a MANET using our proposed ant algorithm.
- Evaluate the End-to-End delay using two different mobility models such as, Random Waypoint Mobility (RWM) and Boundless Simulation Area Mobility (BSAM). The aim of using two different mobility models is to investigate the effect of mobility on End-to-End delay.
- Comparison study between our proposed ant algorithm with another ad hoc routing protocol such as Ad hoc On-Demand Distance Vector (AODV) by evaluating the End-to-End delay in both algorithms. The aim of this comparison study is to investigate the performance of our algorithm in comparison with other routing protocols.

1.3 Organization of Thesis

This thesis covers different search areas as indicated in Figure 1-1. These search areas are mobile ad hoc networks, ant colony algorithms, mobility models and queuing network analysis which are organized to work together yielding our emergent hybrid simulation algorithm for minimizing End-to-End delay through mobile ad hoc networks.

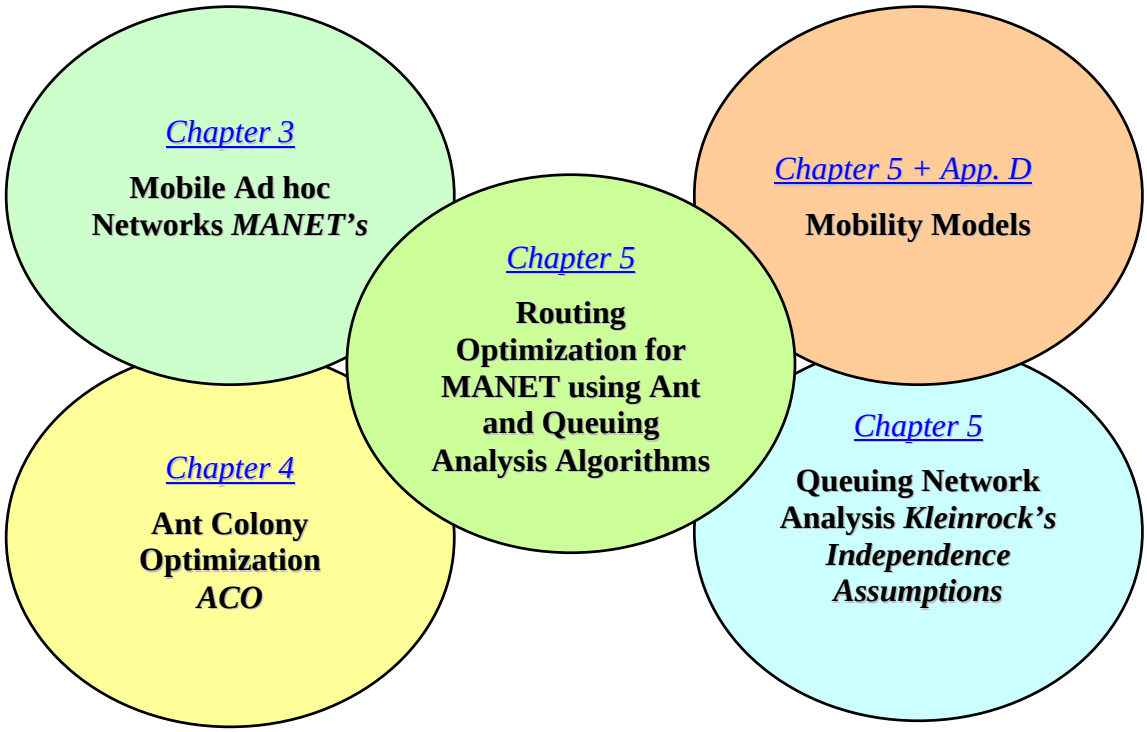


Figure 1-1. Covered study areas of our thesis

The rest of this thesis is organized as follows:

Chapter 2 provides a review of literature relevant to this research. Practically, the literature review contains an overview of routing problem in MANET as well as a brief description of developed techniques that are based on metaphors taken from nature by ant algorithms.

Chapter 3 will introduce an overview about ad hoc networks, different definitions, different classifications, properties and different applications.

Chapter 4 will introduce the basic ideas that will be used for future development of artificial ants behavior which is inspired from real ants behavior as well as an exhaustive review of natural ants behavior and ant system development. An account of the usage of artificial ants behavior for developing the ant system tool that can be used in solving routing in dynamic problems is provided later.

Chapter 5 introduces a dynamic adaptive routing algorithm based on ant algorithm in combination with queuing network analysis. Finally, we will present and discuss the results of different experiments that have been done by simulation.

Chapter 6 demonstrates the comparison study between our hybrid simulation algorithm (combination of ant algorithm with queuing network analysis) with one of the famous On-Demand routing protocols (reactive) called *Ad hoc On-Demand Distance Vector* (AODV).

Chapter 7 summarizes the most important results of this thesis and gives an outlook into future research.

Bibliography

[Beni and Wang, 1989a] G. Beni and J. Wang. Swarm Intelligence. *In Proceedings Seventh Annual Meeting of the Robotics Society of Japan*, RSJ Press, pages 425-428, 1989.

[Beni and Wang, 1989b] G. Beni and J. Wang. Swarm Intelligence in Cellular Robotic Systems. *In Proceedings of the NATO advanced Workshop on Robotics and Biological Systems*, 1989.

[Beni, 1988] G. Beni. The Concept of Cellular Robotic System. *In Proceedings 1988 IEEE Int. Symp. On Intelligent Control*, IEEE Computer Society Press, pages 57-62, 1988.

[Bonabeau et al., 1999] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence*. Oxford University Press, 1999.

[Darwin and Wallace, 1858] C. Darwin and A. Wallace. On the Tendency of Species to Form Varieties; and on The Perpetuation of Varieties and Species by Natural Means of Selection. *Journal of the Proceedings of the Linnean Society, Zoology*, vol. 3, pages 45-62, 1858.

[Dorigo, 1992] M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). Ph.D. thesis, DEI, Politecnico di Milano, pages 140, Italy, 1992.

[Glover, 1986] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research*, vol. 13, pages 533-549, 1986.

[Goldberg, 1989] D. E. Goldberg. *Genetic Algorithms*. Addison-Wesley, 1989.

[Haykin, 1998] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 2nd edition, 1998.

[Holland, 1975] J. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Ann Arbor, Michigan, 1975.

[Hopfield and Tank, 1985] J.J. Hopfield and D. W. Tank. Neural Computation of Decision in Optimization Problems. *Biological Cybernetics 52*, pages 141-152, 1985.

[Kennedy et al., 2001] J. Kennedy, R. C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA, 2001.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi. Optimization by Simulated Annealing. *Science* vol. 220, pages 671-680, 1983.

[McCulloch and Pitts, 1943] W. S. McCulloch and W. H. Pitts. A Logical Calculus of The Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, vol.5, pages 115-133, 1943.

[Metropolis et al., 1953] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics 21*, pages 1087-1092, 1953.

[Michalewicz, 1996] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition, 1996.

[Mitra et al., 1986] D. Mitra, F. Romeo, and A. Sangiovanni-Vincetelli. Convergence and Finite Time Behavior of Simulated Annealing. *Adv. Appl. Prob. 18*, pages 747-771, 1986.

[Mostafa-Sami and Tarek, 1997] M. Mostafa-Sami and A. Tarek. Modeling of Neural Network by High Level Petri Net. *Fifth International Conference on Artificial Intelligence Applications (5th ICAIA'97)*, Cairo, Egypt, 1997.

[Tarek and Sampels, 2001] A. Tarek and M. Sampels. Optimization of Circular Networks by Genetic Algorithms. *Third Middle East Symposium on Simulation and Modelling (3rd MESM'01)*, Amman, Jordan, 2001.

[Thomas, 1996] B. Thomas. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

Chapter 2

RELATED WORK

2.1 Introduction

Routing algorithms are often difficult to be formalized into mathematics, they are instead tested using extensive simulation [Broch et al.,1998]. Early work on volatile network environments experienced in mobile ad hoc networks (MANETs) depends primarily on applying the traditional approaches of routing in wired networks, such as distance vector or link state algorithms. While many optimizations to these algorithms exist, each of them is primarily concerned with finding the minimum hop route from source to destination [Perkins and Royer, 1999] [Johnson and Maltz, 1996] [Jacquet et al., 2001]. A large amount of work has also been done in the area of energy efficient routing. This approach attempts to maximize network lifetime by routing through paths, which use the least amount of energy relative to each node [Gomez et al., 2001].

Recently, more attention has been paid to use specific network parameters when specifying routing metrics. Examples might include delay of the network, link capacity, link stability or identifying low mobility nodes. These schemes are generally based on previous work, which is then enhanced with the new metrics.

In the last years, a great deal of literature has been published in the field of mobile ad hoc networks. There exists relatively little work with regard to biologically inspired algorithms for routing in communications networks. However, there are a number of notable examples which show that these concepts can provide a significant performance gain over traditional approaches. In this chapter, we describe a number of the most famous algorithms in this field and give an overview of using ACO algorithms in routing problems. This chapter is intended

to serve as an exhaustive guide to all possible and available literature that applying ACO algorithms to solve routing problems, see Figure 2-1.

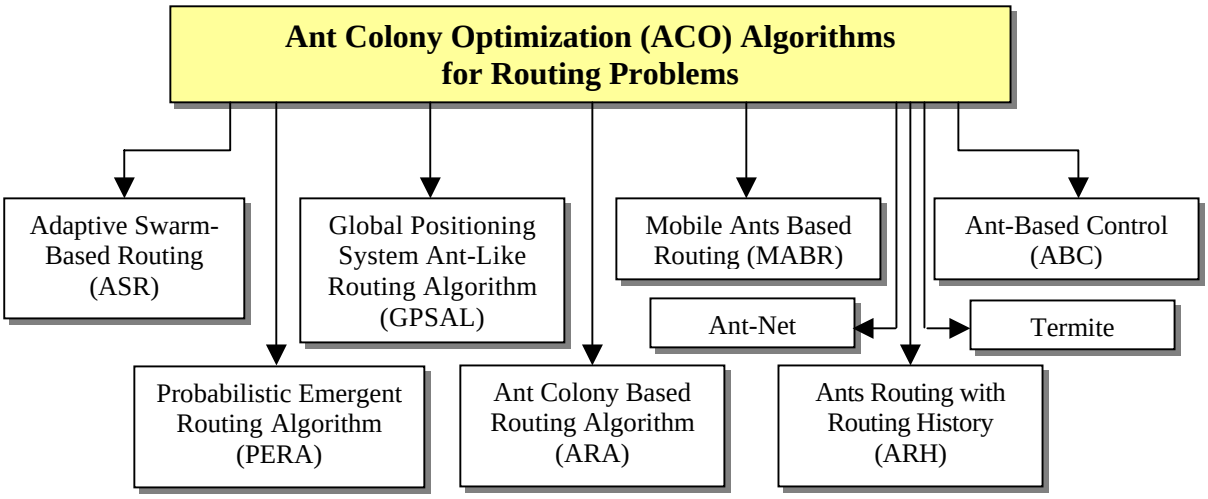


Figure 2-1. Ant colony optimization algorithms for routing problems

2.2 Ant-Based Control Algorithm

One of the earliest work on swarm intelligent routing is done by Schoonderwoerd et al. on the *Ant Based Control* (ABC) algorithm [Schoonderwoerd et al.,1996]. This algorithm is applied for a wired circuit switched network, such as a telephony network. ABC was later modeled analytically in [Subramanian et al., 1997].

The algorithm is adaptive and exhibits robustness under various network conditions. This is accomplished by using many simple agents, called ants, which modify the routing policy at every node in a telephone network by depositing a virtual pheromone trail on routing table entries. The ants' goal is to build routing tables, and adapt them to load changes in the telephone network so that performance is optimized. Performance here is measured by the rate of incoming calls, which are accepted. A phone call is either accepted or rejected at setup time depending on the available network resources.

Each node has a routing table, which has the same form as given in Table 2-1, where each row represents a destination and each column represents a neighbor. The ants are launched from the sources to various random destinations and eliminated once they reach their destinations. Therefore, the probabilities of the routing tables are updated as the ant visits the node, based on the lifetime of the ant T at the time of the visit.

Table 2-1. ABC routing table

Destination	Next Hop	
	B	C
E	0.45	0.55
F	0.75	0.25

Then a step size is defined for that node, according to: $\delta r = \frac{a}{T} + b$ where *a* and *b* are both design parameters. This step size rule is chosen heuristically. It assigns a greater step size to those ants that are successful at reaching the node faster. The routing table is then updated according to:

$$r^{i}_{i-1,s}(t+1) = \frac{r^{i}_{i-1,s}(t) + \delta r}{1 + \delta r}$$

$$r^{i}_{n,s}(t+1) = \frac{r^{i}_{n,s}(t)}{1 + \delta r}, \quad n \neq i-1$$

where *s* is the source node, *i* is the current node and *i*-l the previous node. The age of the ant, *T*, can be manipulated by adding delays D_i at every node, which are given by $D_i = c.e^{-d.s}$ where *c*, *d* are design parameters and *s* is the spare capacity of each node in the telephone network. Note that the ant both uses and updates the routing table at the same time. For example, in Table 2-1, if the source is node *F* and the destination is node *E*, then the ant will update the row for *F* and use the node for *E* to find the next hop. The update rules are applied such that the condition, $\sum_n r^{i}_{n,s} = 1$ where *n* is the number of all neighbors to *i*, is satisfied. Another improvement is the incorporation of an *exploration factor g*, where with probability (1-*g*) the ants are forwarded with a total uniform distribution and with probability *g*, ants are forwarded according to the routing table probabilities. The problem with this algorithm is that it updates the table probabilities when the ants move forward. The underlying assumption is that the network is symmetric, which is not always true. This issue is discussed by the next algorithm, called *AntNet.*

2.3 AntNet Algorithm

DiCaro and Dorigo introduced an adaptive routing algorithm based on the adaptive learning and ant colonies called AntNet, for routing in packet-switching networks [DiCaro and Dorigo, 1997]. This algorithm explores the network with the goal of building (and rebuilding) routing tables and keeping them adapted to traffic conditions. It is presented as a distributed, scalable and responsive algorithm for routing in wired networks. The algorithm is improved upon in [Barán and Sosa, 2000].

In the *AntNet* algorithm, routing is determined by means of very complex interactions of *forward* and *backward* network exploration agents "ants". The idea behind this sub-division of agents is to allow the backward ants to utilize the useful information gathered by the forward ants on their trip from source to destination. Based on this principle, no node routing updates are performed by the forward ants. Their only purpose in life is to report network delay conditions to the backward ants, in the form of trip times between each network node. The backward ants inherit this raw data and use it to update the routing table of each node.

The entries of the routing table are probabilities, where the sum of each row in this table must equal 1. These probabilities serve a dual purpose:

1. The exploration agents of the network use them to decide the next hop to a destination, randomly selecting among all candidates based on the routing table probabilities for a specific destination.
2. The data packets deterministically select the path with the highest probability for the next hop.

The sequence of actions in AntNet is simple and intuitive:

1. Each network node launches forward ants to all destinations in regular time intervals.
2. The ant finds a path to the destination randomly based on the current routing tables.
3. The forward ant creates a stack, pushing in trip times for every node as that node is reached
4. When the destination is reached, the backward ant inherits the stack.
5. The backward ant pops the stack entries and follows the path in reverse.
6. The routing tables of each visited node are updated based on the trip times.

2.3.1 Updating Routing Tables

The update of the routing table is done by using the quantity, *r'*, which derived according to:

$$r' = \begin{cases} \frac{T}{C\mu}, C \geq 1, \textit{if } \frac{T}{C\mu} < 1 \\ 1, \qquad \textit{Otherwise} \end{cases}$$

where T is the trip time from the current node to the destination, μ is average of T, and C is a scaling factor, usually set to 2.

Except for the routing table, each node also possesses a table with records of the mean and variance of the trip time to every destination. The ratio of the variance to the mean (σ/μ) is used as a measure of the consistency of the trip times, and accordingly, it alters the effect of the trip time on the routing table. Based on the value of *r'*, we determine the relative

goodness of the trip time of an ant. Corresponding strategies of either decreasing or increasing the value of r' by a certain amount are then followed, based on setting the threshold for the good/bad trip time to 0.5, and selecting a threshold for the (σ/μ) ratio (see Table 2-2).

Table 2-2. AntNet processing cases

	$r'<0.5$	$r'>0.5$
$\sigma/\mu>\delta$	$+(1-e^{-a'\frac{\sigma}{\mu}})$	$-(1-e^{-a'\frac{\sigma}{\mu}})$
$\sigma/\mu<\delta$	$-e^{-a\frac{\sigma}{\mu}}$	$+e^{-a\frac{\sigma}{\mu}}$

The principle of these updates is that small values of r' correspond to small values of T and vice versa. By testing, the case in which the consistency is high and the time is good, the algorithm requires the processed value r' to be even smaller. Therefore, an exponential quantity is subtracted. This quantity is the exponentially decaying function of the consistency ratio, and it achieves its highest value when the variance is very small. The decay rate can be controlled through parameters a' and a.

Further, positive or negative reinforcement of good or bad routes takes place next, via *positive* and *negative feedback*. Any positive reinforcement of probability should be negatively proportional to current probabilities, and any negative one should be proportional to current probabilities. The effect of this is to prevent saturation of the routing table probabilities to be 0 or 1. The node that receives the positive reinforcement is the one from which the backward ant comes. This is the same node chosen by the forward ant as next-hop on the way to its destination. All the other neighbors of the current node need to be negatively reinforced to preserve the unit sum of all the next-hop probabilities. The reinforcement equations are:

$$r_+ = (1 - r') \cdot (1 - P_{df})$$

$$r_- = -(1 - r') P_{dn}, n \neq f, n \in N_k$$

where P_{df}, P_{dn} are the previous routing table probabilities, f is the node from which the backward ant comes, N_k is the neighbor of node k (current node), and d is the destination node. The last step is to update the routing table probabilities using the following rules.

$$P_{df} = P_{df} + r_+ \qquad (1)$$

$$P_{dn} = P_{dn} + r_- \qquad (2)$$

The packets of the network then use these probabilities in a deterministic way, choosing as next hop the one with the highest probability.

The behavior of the AntNet algorithm can be more easily understood with an example, therefore, we will discuss in detail an example for this algorithm.

2.4 Mobile Ants Based Routing

Mobile Ants Based Routing (MABR) algorithm is inspired by social insects and introduced as the first routing algorithm for large-scale mobile ad hoc networks MANETs [Heissenbüttel and Braun 2003].

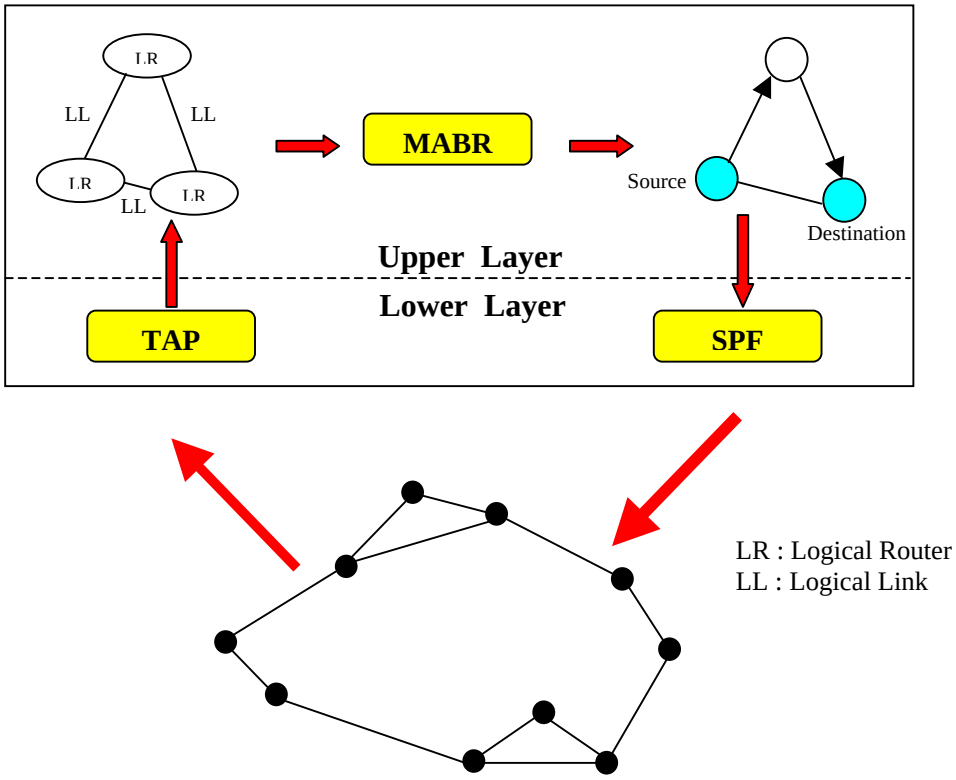


Figure 2-2. The interactions between TAP, MABR and SPF protocols

The approach presented in AntNet is extended to ad hoc networks by abstracting the dynamic, irregular topology of a MANET into a topology with *logical routers* and *logical links*, this is done by using TAP (Topology Abstracting Protocol). The term logical router represents a collection of mobile hosts, which all together build and share among each other the same routing tables. A logical link represents a path along a roughly straight line to a distant logical router over possible multiple hops. To build these logical routers, nodes geographically close to each other are grouped together. Logical links are established between selected logical routers. An optimized greedy routing algorithm is used to forward messages between logical nodes. On top of this abstract topology, the actual routing protocol MABR (Mobile Ants Based Routing) will be run. This ants-based protocol is responsible for updating the routing tables of logical routers and determining logical paths for routing packets over this abstract topology. Finally, the SPF (Straight Packet Forwarding) protocol is applied in order to transmit packets over a logical link. Therefore, it forwards packets along this logical link in a

greedy manner. An overview of the architecture with the interactions between the protocols is depicted in Figure 2-2.

2.5 Ant-Colony Based Routing Algorithm

This Ant-Colony Based Routing (ARA) algorithm presents a detailed routing scheme for MANETs [Günes et al., 2002]. It is similarly constructed as many other routing approaches and consists of three phases.

2.5.1 Route Discovery

Route discovery is responsible for creating new routes. The creation of new routes requires the use of a *forward ant* (FANT) which is initiated at the source and a *backward ant* (BANT) which is initiated at the destination. The FANT is a small packet with a unique sequence number. A forward ant is broadcasted by the sender and will be relayed by the neighbors of the sender as shown in Figure 2-3.

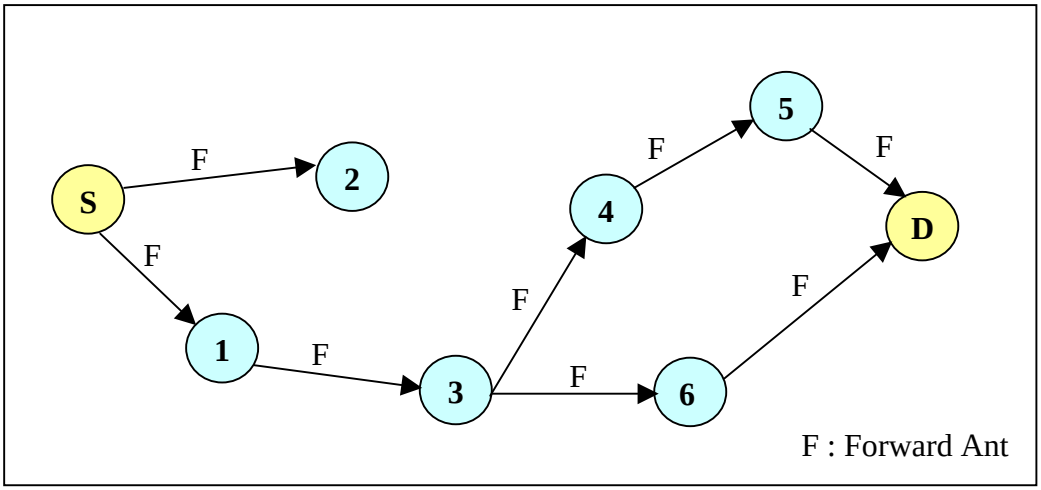


Figure 2-3. Route discovery phase by forward ant

A node receiving a FANT for the first time, creates a record in its routing table. A record in the routing table consists of (*destination address*, *next hop*, *pheromone value*). The node interprets the source address of the FANT as destination address, the address of the previous node as the next hop and computes the pheromone value depending on the number of hops the FANT needed to reach the node. The node forwards the FANT to its neighbors. If the FANT reaches the destination, a BANT will be returned to the source.

The BANT will take the same path as the FANT but in the opposite direction as indicated in Figure 2-4. When the sender receives the BANT from the destination node, the path is

established and data packets can be sent. Figures 2-3 and 2-4 schematically depict the route discovery phase.

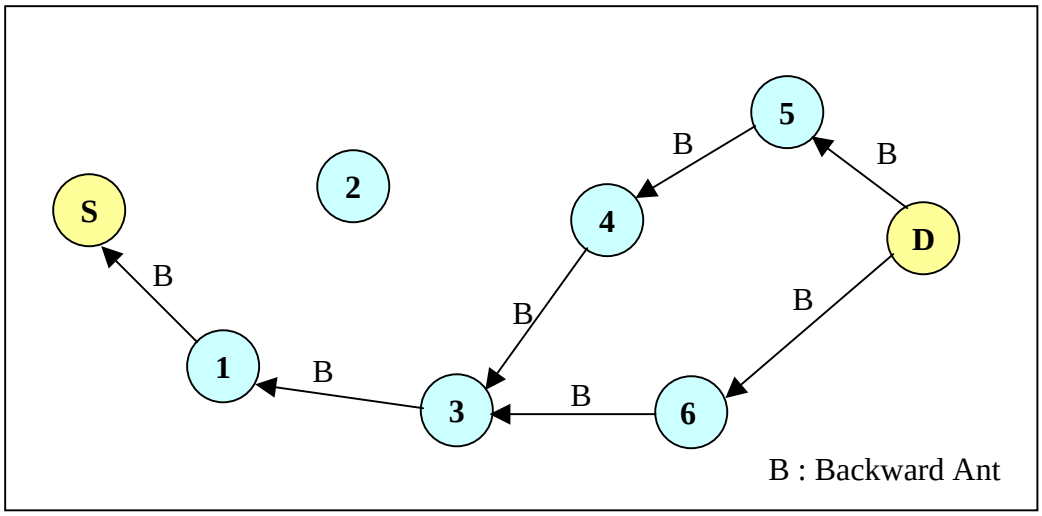


Figure 2-4. Route discovery phase by backward ant

2.5.2 Route Maintenance

Route maintenance is responsible for the improvement of the routes during the communication. ARA does not need any special packets for route maintenance. Once the FANT and BANT have established the pheromone tracks for the source and destination nodes, subsequent data packets are used to maintain the path. Similar to the nature, established paths do not keep their initial pheromone values forever. A high decay rate will quickly reduce the pheromone value. If a regular time interval of one second is taken, the decreasing formula will be $\varphi_{i,j} = (1- q).\ \varphi_{i,j}$, where $\varphi_{i,j}$ is the pheromone concentration and $q \in (0,1]$. Every time data passed by the pheromone value in the routing table will be increased by $\varphi_{i,j} = \varphi_{i,j} + \Delta\varphi$ where $\Delta\varphi$ is a constant amount of pheromone.

ARA prevents loops by a very simple method, which is also used during the route discovery phase. Duplicated FANTs can be identified through the unique sequence number and are removed. If a node receives a duplicated packet, it will set the DUPLICATE_ERROR flag and send the packet back to the previous node. The previous node deactivates the link to this node so that packets cannot be sent anymore to this direction.

2.5.3 Route Failure Handling

Route failure handling is responsible for handling routing failures that caused especially through node mobility which is very common in mobile ad hoc networks. ARA recognizes a route failure through a missing acknowledgment. The ROUTE_ERROR is set when the node

misses an acknowledgment. When this occurs, first, the link will be disabled and then it will check if there exists any other route to the destination node. If no route exists to the destination, it will inform its neighbors, hoping that they can forward the packet toward the destination. If none of the neighbors can forward the packet toward the destination, the packet is sent back to previous nodes until the source has been reached. Control bandwidth overhead is minimized and remains constant across several degrees of the network volatility.

2.6 Termite

Termite is a distributed routing algorithm for mobile wireless ad hoc networks [Roth and Wicker, 2003]. It is designed using the swarm intelligence framework in order to achieve better adaptivity, lower control overhead and lower per-node computation. The algorithm is inspired by the hill building behavior of termites. The Termite algorithm may be described simply as follows, each node in the network has a specific pheromone scent. As packets move through the network links between nodes, they are biased toward moving in the direction of destination pheromone gradients. Packets follow this gradient while laying pheromone on the links that they traverse toward their source. The amount of pheromone deposited by a packet on a link is equal to the utility of its traversed path. Using this method of pheromone updating, consistent pheromone trails are built through the network. Changes in the network environment, such as topological or path quality changes, are accounted for by allowing pheromone to decay over time. This requires paths to be continuously reinforced by new traffic, new information about the network is added to links. Each node records the amount of pheromone that exists for each destination on each of its links. This creates a routing table similar to those found in traditional link-state routing algorithms.

2.6.1 Pheromone Update

The pheromone update equation is a function which updates the pheromone of the packet source at a node upon its arrival. The update function shown here is the traditional approach and is known as the *Pheromone Filter*. Pheromone on all links decays simultaneously upon packet arrival, and proportionally to packet interarrival times. This is known as continuous pheromone decay and was originally presented in [Günes et al., 2003]. This idea is extended in this work also by decaying pheromone when it is checked to send a packet. Each packet maintains the total utility of the path it has traversed (updated at each node visited) and

deposits this amount of pheromone on each link. The *Pheromone Filter* update equation is,

$$\forall i \quad P_{i,s}^n = P_{i,s}^n \cdot e^{-(t - t_{s,\,obs}^n)\tau} \quad \text{and} \quad P_{r,s}^n = P_{r,s}^n + \gamma$$

where $P_{i,s}^n$ is the amount of pheromone from source node *s*, on the link from neighbor node *i*, at node *n*. The previous hop of the packet is node *r*. The variable γ is the amount of pheromone carried by the packet, which will vary from packet to packet depending on its path. The time $t_{s,\,obs}^n$ is the last instant that the pheromone from source node *s* at node *n* was observed, either due to packet sending or packet receiving. τ is the pheromone decay rate.

2.6.2 Forwarding Equation

The forwarding equation is used to determine the probability of using a link, based on the amount of pheromone on it.

$$P_{i,d}^n = \frac{(P_{i,d}^n + K)^F}{\sum_{j=1}^{N_n} (P_{j,d}^n + K)^F}$$

where $P_{i,d}$ is the probability of using neighbor node *i* in order to get to destination node *d*, at node *n*. N_n is the number of neighbors of node *n*. $K \geq 0$ is the pheromone threshold and $F \geq 0$ is the pheromone sensitivity.

2.6.3 Termite Summary

Termite has been shown to perform well, especially in regions of high node mobility [Roth and Wicker, 2003]. However, packets often take substantially longer paths than necessary ones. In part, this is required in order to maintain current pheromone through the network, but this behavior can also generate significant resource inefficiencies.

AntNet is a related algorithm, although its forward/backward ant feature is not used. Termite and ARA share nearly the same data routing mechanism, although they differ significantly with regards to route discovery and failure recovery.

2.7 Ants Routing with Routing History in Dynamic Networks

Ants Routing with routing History (ARH) algorithm is a stochastic routing algorithm that chooses a suitable route and efficiently acquires the information of a route, when the nodes communicate man-to-man in ad hoc communication environments [Fujita et al., 2002].

ARH algorithm considers each node as agent, and copes with the dynamic environments by learning the route. Additionally, ARH can perform a robust routing by selecting stochastically the

good route, and efficiently acquire the information of a route by using routing history. It is based on two algorithms Ants-Routing and AntNet that improve learning efficiency by recording routing history in a message packet. It is based on two main processes, selecting next route and updating processes.

2.7.1 Selecting Next Route

In this process, each node determines next hop according to the stochastic values on the routing table. The stochastic values satisfy this normalizing condition $\sum_{z \in Neighbor\ of\ k} P_k(d,z) = 1$ for each destination node *d*, where $P_k(d,z)$ denotes the sending probability from *k* to *z* as next hop, and *z* is the neighbor node of *k*.

2.7.2 Updating Process

In this process, each node stores its own ID, sum of transmission time, message processing time and other information (as routing history) in a message packet, and sends them to next hop. The size of queue to store routing history is fixed. If the size of stored routing history is larger than the size of queue, the oldest history is removed, and new history is stored. That is because the reliability of the oldest history is low. A node that accepts a message packet is trained by *backward exploration*. The training phase is performed to the routing table of which its destinations correspond to the relay nodes that have been recorded in queue. The probabilistic table is updated using the following equations,

$$\Delta p = \gamma^{h} \cdot \frac{k}{f(t_{d'})} \qquad \gamma \in (0,1),\ k > 0 \tag{1}$$

$$P_y(d',x) = \frac{P_y(d',x) + \Delta p}{1 + \Delta p} \tag{2}$$

$$P_y(d',z) = \frac{P_y(d',z)}{1 + \Delta p} \qquad z \in Neighbor\ of\ y,\ z \neq x \tag{3}$$

where $P_y(d',x)$ denotes the sending probability from *y* to *x* as next hop, d' is the destination node, *z* is neighbor node of y, *k* is learning rate, $f(t_{d'})$ is a non-decreasing function of $t_{d'}$, $t_{d'}$ time elapsed since message has been generated, γ is a discount rate and *h* is the number of hops from *y* to d'. Δp denotes the amount of change in the stochastic value. In equation (2), the sending probability from *y* to *x* as next hop is updated. In equation (3), the stochastic values of the neighbor node of y without *x* are normalized. According to Δp, its great value is

assigned to the nearer and higher reliability route and the small value is assigned to the farther and lower reliability route. This updating process makes the possibility to learn more efficient by providing the appropriate Δp.

2.8 Ants Routing with Routing History and No Return Rule

Ants-Routing with routing History and no return rule (ARHnr) is a modified version of the previous algorithm ARH that adds enhanced "*no return rule"* to ARH to solve the *routing-lock* problem. Routing-lock is a phenomenon in which a route is fixed because a routing probability converges in the neighbor of one route during the learning progress. When change of topology takes place and it becomes impossible to use the routes according to this phenomenon, a lot of time is required to discover the new one. In addition, selecting the conventional route will be continued without the ability of discovering a new route, although a new efficient one may be available. "*No return rule*" function is to eliminate return rule (back track path) while selecting next node. It is firstly introduced into *Accelerated Ants Routing* and enhanced in this algorithm.

2.9 Global Positioning System Ant-Like Routing Algorithm

In 2000, Daniel Camara and Antonio A.F. Loureiro present a novel routing algorithm for MANET called Global Positioning System Ant-Like Routing Algorithm (GPSAL) which is based on the physical location of a destination node and mobile software agents modeled on ants [Camara and Loureiro, 2000]. Ants are used to collect and disseminate information about nodes location in the MANET. GPSAL works with datagram packets and assumes that all mobile hosts participating in a MANET have a GPS unit, which provides the host its approximate three-dimensional position (latitude, longitude, and altitude), velocity and accurate time in Universal Time Coordinate (UTC) format. The proposed algorithm assumes that mobile hosts are moving in a plane, and therefore, the altitude information is not used. The algorithm makes use of location information of a mobile host to reduce the number of routing messages. All hosts in the MANET have a routing table, where each entry represents a known host *d* and has the following information; current location of *d*, previous location of *d*, time-to-live of current location, time-to-live of previous location and whether *d* is a mobile or fixed host.

Whenever a mobile node wants to join the MANET, it listens to the medium to find out a neighbor node *n*. Once a neighbor node *n* is identified, the mobile host sends a request packet to *n* asking for its routing table which is sent back to the host. From this moment on, the new

mobile host can start routing and sending packets in the MANET. The routing protocol is based on the physical location of a destination host *d* stored in the routing table. If there is an entry in the routing table for host *d*, the best possible route is chosen using the shortest path algorithm. The route, comprised of a list of nodes and the corresponding TTL's, is attached to the packet which is sent to first host in the list. If host *d* is not found in the routing table, the mobile node sends a message to the nearest fixed node that tries to find the destination node. Note that the information in the routing table, which was used to route the packet, probably reflects a snapshot in the past, and the current network configuration may be different. Therefore, each host upon receiving a packet compares the routing information present in the header with the information in its routing table. The entries that have older information than that in the packet received are updated. This is performed by comparing the TTL field in the packet received and in the routing table. Furthermore, each intermediate node can change the route to a destination node when there is a better route.

An important aspect of any routing algorithm for MANETs is how the routing table is updated. It is clear that better routes can be determined whenever a host has a more recent information about the network configuration. Routing information can be obtained both locally and globally. Local information is obtained from a neighbor node that periodically broadcasts only the changes occurred since the last time (this interval is a configuration parameter). Global information can be disseminated more rapidly using mobile software agents modeled on ants. Ants are agents sent to random nodes on the network. In a MANET, when a mobile computer is powered on, it may not know the physical location (current or past) of other hosts, however, this information can be gathered when a computer receives a message to be forwarded to another node or as a result of its own route discovery. All messages carry the routing information that can be used by intermediate hosts in the routing process. The route discovery can be accelerated using mobile software agents modeled on ants which are responsible for collecting and disseminating more up-to-date location information of mobile hosts. Any node can send an ant to any other node on the MANET. When an intermediate or destiny host receives an ant, it compares the routing information present in the packet received with its routing table, and updates the entries that have older information as explained above. The main goal of the ant agent is to exchange routing information among the network nodes. This process is fundamental for the good performance of our algorithm, since newer information can be disseminated without waiting table exchange. Of course there is an overhead associated with this process that can be controlled by the number of ants in the MANET, but it makes the neighbors' tables exchange more

interesting. The information about node movement can now flow more quickly through the entire network. Another important point is how to determine a destination to where an ant should be sent in order to collect more up-to-date information. One possibility is to choose the node with the oldest information in the routing table or the farthest known node in the MANET or just any node.

2.10 Probabilistic Emergent Routing Algorithm

A Probabilistic Emergent Routing Algorithm for mobile ad hoc networks (PERA) which presented in [Baras and Mehta, 2003] is a routing algorithm for mobile ad hoc networks based on the swarm intelligence paradigm and similar to the swarm intelligence algorithms described in [DiCaro and Dorigo, 1997] and [Subramanian et al., 1997]. The algorithm uses three kinds of agents - *regular forward ants, uniform forward ants* and *backward ants*.

Uniform and regular forward ants are agents (routing packets) that are of *unicast* type. These agents proactively explore and reinforce available paths in the network. They create a probability distribution at each node for its neighbors. The probability or goodness value at a node for its neighbor reflects the likelihood of a data packet reaching its destination by taking the neighbor as a next hop.

Backward ants are utilized to propagate the information collected by forward ants through the network and to adjust the routing table entries according to the perceived network status. Nodes proactively and periodically send out forward regular and uniform ants to randomly chosen destinations. Thus, regardless of whether a packet needs to be sent from a node to another node in the network or not, each node creates and periodically updates the routing tables to all the other nodes in the network.

The algorithm assumes bidirectional links in the network and that all the nodes in the network fully cooperate in the operation of the algorithm.

2.11 Adaptive Swarm-based Routing in Communication Networks

Adaptive Swarm-based Routing (ASR) algorithm increases the speed of convergence and has quite good stability by using a novel variation of reinforcement learning and a technique called *momentum* [Lü et al., 2004]. The momentum technique is used to increase the speed of convergence and avoid high oscillation. It is a very effective mechanism that seeks to incorporate a memory in the learning process, increases the stability of the scheme and helps to increase the learning efficiency of the network.

ASR algorithm uses two types of ants called *Forward Ant*, denoted F_{ant}, which will travel from the source node *s* to a destination *d* and *Backward Ant*, denoted B_{ant}, will be generated by a forward ant F_{ant} in the destination *d*. It will come back to *s* following the same path traversed by F_{ant}, with the purpose of using the information already picked up by F_{ant} in order to update routing tables of the visited nodes.

ASR algorithm can react and deal with changes of network. Reported results showed that ASR is better than AntNet in performance and in behavior. ASR is converging rapidly toward a good stable delay value after an initial transitory phase.

2.12 Ants-Routing Algorithms

Ants-Routing Algorithms are distributed routing algorithms for data networks based on biological ants that explore the network and rapidly learn good routes, using a novel variation of reinforcement learning [Subramanian et al., 1997]. The framework of reinforcement leaning is shown in Figure 2-5. The basic reinforcement learning procedure is stated as follows,

Step1: An agent decides an action according to observed condition in time *t*, and performs selected action a_t.

Step2: Environment is changed from s_t to s_{t+1}, and an agent acquires reward r_t corresponded with transition environment.

Step3: $t \leftarrow t+1$, and go to step 1.

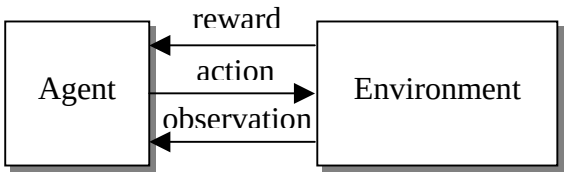


Figure 2-5. The framework of reinforcement learning

An agent selects an action for the purpose of the maximum reward acquisition.

There are two algorithms called *Regular Ant Algorithm* and *Uniform Ant Algorithm* both are fully adaptive to topology changes and link costs changes in the network. The Regular Ant Algorithm is based on earlier work by [Schoonderwoerd et al., 1997] for call routing in telephone networks. It is the single shortest path algorithm and is only applicable to networks with symmetric path costs. Uniform ant algorithm is a natural multi-path routing algorithm that is applicable to data networks with or without path cost symmetry.

2.13 Accelerated Ants-Routing

Accelerated Ants-Routing algorithm uses ant-like agents that go through the network randomly, without a specific destination, updating pheromone entries pointing to their source. Converging speed of the routing table is the most important factor to evaluate this routing algorithm, due to frequent and unpredictable changes in ad hoc network topology. Accelerated Ants-Routing algorithm is a modified Ants-Routing algorithm that makes convergence speed accelerated compared with other reinforcement base algorithms as Q-Routing, DRQ-Routing and Ants-Routing algorithms [Matsuo and Mori, 2001].

Bibliography

[Barán and Sosa, 2000] B. Barán, R. Sosa. A New Approach for AntNet Routing. *Ninth International Conference on Computer Communications and Networks IEEE ICCCN-2000*, Las Vegas - Estados Unidos, 2000.

[Baras and Mehta, 2003] J.S. Baras, H. Mehta. A Probabilistic Emergent Routing Algorithm for Mobile Ad Hoc Networks. *WiOpt03 Proc. of Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

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

[Camara and Loureiro, 2000] D. Camara, and A. Loureiro. A GPS/Ant-Like Routing Algorithm for Ad Hoc Networks. *Wireless Communications and Networking Conference, WCNC 2000*, vol. 3, pages 1232-1236, 2000.

[Camara and Loureiro, 2000] D. Camara, and A. Loureiro. A Novel Routing Algorithm for Ad Hoc Networks. *Proceedings of the 33rd Hawaii International Conference on System Sciences*, vol. 8, 2000.

[DiCaro and Dorigo, 1997] G. DiCaro, M. Dorigo. AntNet: A mobile Agents Approach to Adaptive Routing. *Technical Report IRIDIA/97-12*, Universite Libre de Bruxelles, Belgium, 1997.

[Fujita et al., 2002] K. Fujita, A. Saito, T. Matsui, and H. Matsuo. An Adaptive Ant-Based Routing Algorithm used Routing History in Dynamic Networks. *4th Asia-Pacific Conference*

on Simulated Evolution And Learning (SEAL'02). IPSJ SIGNotes MoBiLe computing and wireless communications, Abstract No.020 – 040. 2002.

[Gomez et al., 2001] J. Gomez, A. T. Campbell, M. Naghshineh, C. Bisdikian. Conserving Transmission Power in Wireless Ad-Hoc Networks. *IEEE 9th International Conference on Network Protocols*, 2001.

[Günes et al., 2002] M.Günes, U. Sorges, and I. Bouazizi. ARA- The Ant Colony Based Routing Algorithm for MANETs. *International Conference on Parallel Processing Workshops (ICPPW'02)*, IEEE Computer Society Press, pages 79-85, 2002.

[Günes et al.,2003] M. Günes, M. Kähmer, I. Bouazizi. Ant Routing Algorithm (ARA) for Mobile Multi-hop Ad-hoc Networks - New Features and Results. *In Proceedings of the 2nd Mediterranean Workshop on Ad-Hoc Networks (Med-Hoc-Net'2003)*, Tunesia, 2003.

[Heissenbüttel and Braun, 2003] M. Heissenbüttel and T. Braun. Ants-Based Routing in Large Scale Mobile Ad-Hoc Networks. *Kommunikation in verteilten Systemen (KiVS03)*, Leipzig, Germany, 2003.

[Jacquet et al., 2001] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, L.Viennot. Optimized Link State Routing Protocol for Ad-Hoc Networks. 2001.

[Johnson and Maltz, 1996] D. Johnson, D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth (Kluwer Academic Publishers), chapter 5, pages 153-181, 1996.

[Lü et al., 2004] Y. Lü, G. Zhao, F. Su, and X. Li. Adaptive Swarm-Based Routing in Communication Networks. *Journal of Zhejiang University Science JZUS* 2004, vol. 5(7), pages 867-872, 2004.

[Matsuo and Mori, 2001] H. Matsuo, K. Mori. Accelerated Ants Routing in Dynamic Networks. *In 2nd International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2001.

[Perkins and Royer, 1999] C. Perkins, E. M. Royer. Ad-hoc On-Demand Distance Vector Routing. *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[Roth and Wicker, 2003] M. Roth, S. Wicker. Termite: Emergent Ad-Hoc Networking. *The Second Mediterranean Workshop on Ad-Hoc Networks*, 2003.

[Schoonderwoerd et al. 1997] R. Schoonderwoerd, O. Holland, and J. Bruten. Ant-Like Agents for Load Balancing in Telecommunications Networks. *In Proceedings of the First International Conference on Autonomous Agents*. ACM Press. pages 209–216, 1997.

[Schoonderwoerd et al., 1996] R. Schoonderwoerd, O. Holland, J. Bruten, L. Rothkrantz. Ant-Based Load Balancing in Telecommunications Networks. *Adaptive Behavior*, vol. 5, pages 169-207, 1996.

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

[Subramanian et al., 1997] D.Subramanian, P. Druschel, and J. Chen. Ants and Reinforcement Learning: A Case Study in Routing in Dynamic Networks. In *Proceedings of IJCAI-97, International Joint Conference on Artifical Intelligence*. Palo Alto, CA: Morgan Kauffman, pages 832–838, 1997.

Chapter
3

MOBILE AD HOC NETWORKS

3.1 Introduction

The history of wireless networks started in the 1970s and the interest has been growing ever since. During the last decade, and especially at its end, the interest has almost exploded probably because of the fast growing Internet. The tremendous growth of personal computers and the handy usage of mobile computers necessitate the need to share information between computers. At present, this sharing of information is difficult, as the users need to perform administrative tasks and set up static, bi-directional links between the computers. This motivates the construction of temporary networks with no wires, no communication infrastructure and no administrative intervention required. Such interconnection between mobile computers is called an *Ad hoc Network*. In such environment, it may be necessary for the mobile computers to take help of other computers in forwarding a packet to the destination due to the limited range of each mobile host's wireless transmission.

In this chapter, we will navigate deeply in Mobile Ad hoc Networks (MANETs) and give a detailed overview about many different points in MANET, such as, classification, some different definitions, applications, special features and some routing protocols of MANET.

3.2 Wireless Networks

Today, we see two kinds of wireless networks but the difference between them is not as obvious as it seems [David, 2002]. The first kind and most used today is a wireless network

built on top of a "wired" network and thus creates a *reliable infrastructured wireless network*. The wireless nodes are able to act as bridges in a wired network. This kind of wireless nodes are called base-stations. An example of this wireless network is the cellular-phone networks where a phone connects to the base-station with the best signal quality. When the phone moves out of range of a base-station, it does a "hand-off" and switches to a new base-station within reach. The "hand-off" should be fast enough to be seamless for the user of the network. Other more recent networks are wireless networks for offices that are usually called Wireless Local Area Networks (WLAN). In this kind, there is no infrastructure at all except the participating mobile nodes. This is called an *infrastructureless network* or more commonly an ad hoc network. The word "ad hoc" can be translated as "improvised" or "not organized" which often has a negative meaning, but the sense in this context is not negative but only describing the dynamic network situation.

All or some nodes within an ad hoc network are expected to be able to route data-packets for other nodes in the network beyond their own transmission-range. This is called *peer-level multi-hopping* and is the base for ad hoc networks that constructs the interconnecting structure for the mobile nodes.

Another classification introduced in [Tao, 2004] classified wireless networks into three different types according to relative mobility of hosts and routers:

1. **Fixed wireless network**. Fixed hosts and routers use wireless channels to communicate with each other and form a fixed wireless network. An example is a wireless network formed by fixed network devices using directed antennas, as shown in Figure 3-1.

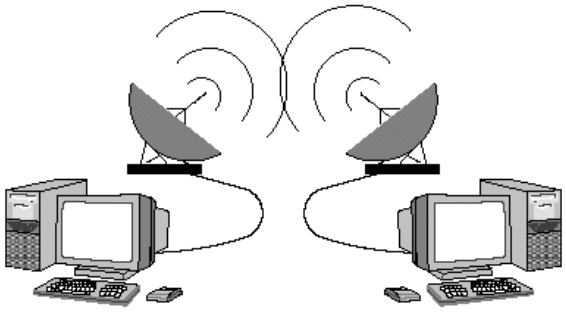


Figure 3-1. An example of a fixed wireless network

2. **Wireless network with fixed access points**. Mobile hosts use wireless channels to communicate with fixed access points, which may act as routers for those mobile hosts, to form a mobile network with fixed access points. An example is a number of mobile laptop users in a building that access fixed access points, as illustrated in Figure 3-2.

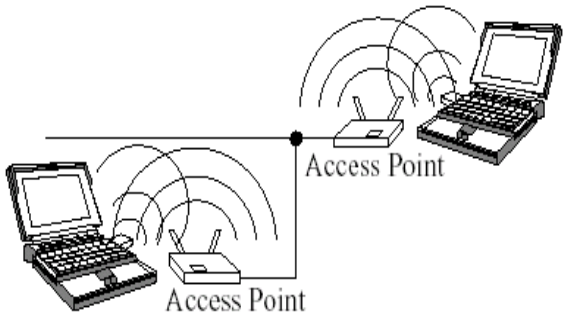


Figure 3-2. An example of a wireless network with access points

3. Mobile ad hoc network. A mobile ad hoc network is formed by mobile hosts. Some of these mobile hosts are willing to forward packets for neighbors. These networks have no fixed routers, every node could be router. All nodes are capable of moving and can be connected dynamically in an arbitrary manner. The responsibilities for organizing and controlling the network are distributed among the terminals themselves. The entire network is mobile, and the individual terminals are allowed to move freely. In this type of networks, some pairs of terminals may not be able to communicate directly with each other and have to rely on some other terminals so that the messages are delivered to their destinations. Such networks are often referred to as *multi-hop* or *store-and-forward* networks. The nodes of these networks function as routers, which discover and maintain routes to other nodes in the networks. The nodes may be located in or on airplanes, ships, trucks, cars, perhaps even on people or very small devices. Figure 3-3 shows an example for vehicle-to-vehicle network communicating with each other by relying on peer-to-peer routings.

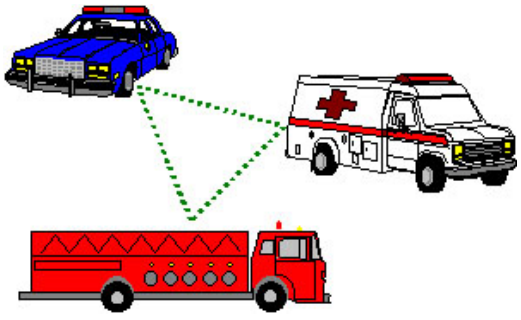


Figure 3-3. An example of a vehicle-to-vehicle network

3.3 Day-to-Day Life Scenario for Wireless Network

Let us imagine here some of the possible use-cases involving mobile terminals and what the future could look like when some technology related problems are solved. Today *Asmaa* is going in a class trip to the Telecom Museum. Like every day, she takes her mobile phone, which has wireless capabilities. On the way to her class, she goes through heterogeneous

wireless networks. When she takes the bus, her terminal enters an ad hoc mode. She meets a friend and wants to exchange files in order to use electronic tools for comparing the results of their math homework. As they are rapidly bored with that, they would like to play a game. In order to play with other players, they check if there are other people that already play a network game within the bus or in the cars around. *Asmaa* runs a peer-to-peer application and makes a search for a good game that could amuse her during the short bus trip. Fortunately, someone in the bus has made his terminal available for others to download programs, and he has a great collection of games. *Asmaa* chooses to try out one of the games made available. The game is automatically downloaded, installed and configured on her terminal. She can start playing right away. Her friend may join also the game easily on her terminal. Later, at the train station *Asmaa*'s terminal automatically connects to the wireless station gateway and enters an access-point mode. As the station gateway offers connectivity to the Internet, *Asmaa* can check her e-mails. When she goes outside the station in order to wait for her friends, she can keep her connectivity to the Internet through the environment management modules of the people walking or waiting in the train station. These modules were downloaded from the station gateway or from some privileged user terminals and they implement an algorithm optimized for the specific wireless station environment. She checks the train departure time in order to make sure that she is not late. Unfortunately, she gets lost, so she establishes a visio call with her classmate that has the possibility to send her a map of the station. When all her classmates and the teachers arrive, they go into the train and *Asmaa*'s terminal then connects to the wireless access-point located in the coach she is in. This gateway has a lot of different applications available for download (such as chat or file-exchange peer-to-peer application). As *Asmaa* is feeling hungry, she orders a snack by initiating a VoIP (Voice over Internet Protocol) with the "coach service", the order is then processed by the coach gateway and transmitted to the bar coach. One of the accompanying teachers has brought her laptop and she uploads the program of the day onto the gateway collaborative-work application. The program is a very big document since it contains pictures, sounds and videos. The students cannot download it on their mobile phones due to the limited memory of their terminals, but they can read it from the gateway. Then a discussion between the students and the teachers starts and they decide to change the program of the day, the teacher makes her document editable and the students start editing it thanks to their PDA. Of course, the collaborative-work application keeps track of all the suggestions issued by the class thanks to a sophisticated yet simple version management system. Once arrived at the museum, the terminals of the students get from the museum wireless gateway all the software components

needed to operate. These components include: authentication, resource discovery, naming and routing. The PDA also downloads the terminal part of the museum applications. This way, the young visitors will be able to access additional documentation in relation with their current location in the museum just by downloading a video, or play with a quiz especially designed to have the students focus on their visit. When the students leave the museum, the results of the quiz and the data they have collected are synchronized with their home data.

3.3.1 The Wireless Network Environments

In the previous scenario, the little girl's terminal enters four different wireless networks, each one having specific characteristics.

The bus environment as shown in Figure 3-4 is a pure mobile ad hoc sub-network, i.e. there is no wired infrastructure, and the terminals themselves have to configure the network without centralized administration.

Figure 3-4. Pure ad hoc network (bus environment)

This network itself may be split into several sub-networks, a private network between *Asmaa* and her friend, and a "public" network initiated by someone who makes publicly available games. The previous scenario shows that the two networks may be merged dynamically.

The train station as shown in Figure 3-5 is an ad hoc sub-network operating in access-point mode, i.e. the terminals use the access-points in the station as bridges to communicate with each other. At the MAC layer, in access-point mode, terminals send and receive messages only from the access point, whereas in ad hoc mode every terminal sends messages to every other. The wireless network at the train station also offers Internet connectivity and multi-hop routing (to allow people walking a small distance from the train station to keep their sessions open).

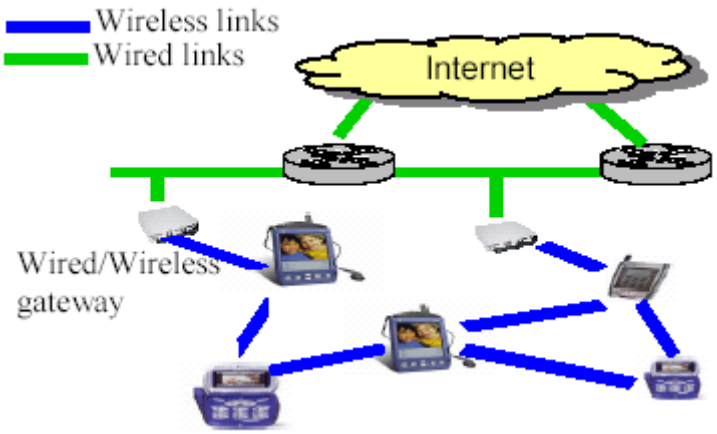


Figure 3-5. The train station environment

Inside the train, wireless terminals can operate in access-point mode (which is more efficient than ad hoc mode) because there is one access point per coach. All the terminals switch automatically from the network in the station to the network in the coach: more precisely the network split into two parts; each part has to reconfigure itself automatically. Each coach also carries a gateway that offers applications to the train company customers as shown in Figure 3-6.

Figure 3-6. The coach environment

The museum is the fourth wireless environment described here. It is very similar to the station environment except it offers different services and applications. Also, one is more likely to trust the people he meets in the museum than in the train station (regarding the personal information disclosed for instance), so may be the services offered could take advantage of this fact.

3.4 Mobile Ad hoc Network

Ad hoc networks are emerging as the next generation of networks and defined as a collection of mobile nodes forming a temporary (spontaneous) network without the aid of any

centralized administration or standard support services. In Latin, *ad hoc* literally means “for this,” further meaning “for this purpose only,” and thus usually temporary [Perkins, 2001].

An ad hoc network is usually thought of as a network with nodes that are relatively mobile compared to a wired network. Hence the topology of the network is much more dynamic and the changes are often unpredictable oppose to the Internet which is a wired network. This fact creates many challenging research issues, since the objectives of how routing should take place is often unclear because of the different resources like bandwidth, battery power and demands like latency. The routing protocols used in ordinary wired networks are not well suited for this kind of dynamic environment.

3.4.1 Mobile Ad hoc Networks Definitions

A clear picture of exactly what is meant by an ad hoc network is difficult to pinpoint. In today’s scientific literature, the term is used in many different ways. There are many different definitions, which describe ad hoc networks, but we just present two of them. The first one is given by *Internet Engineering Task Force* (IETF) group and the other one is given by Murphy [Murphy et al., 1998]. After that, we describe an ad hoc network from a graph theoretical point of view.

3.4.1.1 IETF Definition of Ad hoc Network

A Mobile Ad hoc Network (MANET) is a self-configuring (autonomous) system of mobile routers (and associated hosts) connected by wireless links -the union of which form an arbitrary topology. The routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably. Such a network may operate in a standalone fashion, or may be connected to the larger Internet operating as a hybrid fixed/ad hoc network.

This is in contrast to the well-known single hop cellular network model that supports the needs of wireless communication by installing base stations as access points. In these cellular networks, communication between two mobile nodes completely relies on the wired backbone and the fixed base stations. In a MANET, no such infrastructure exists and the network topology may dynamically change in an unpredictable manner since nodes are free to move as shown in Figure 3-7.

As for the mode of operation, ad hoc networks are basically peer-to-peer multi-hop mobile wireless networks where information packets are transmitted in a store-and-forward manner from a source to an arbitrary destination, via intermediate nodes.

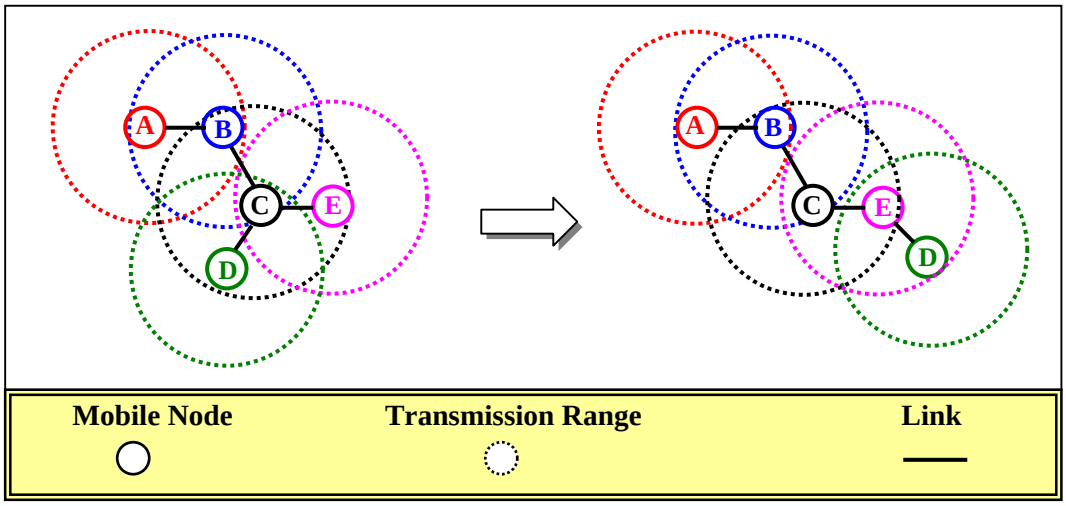


Figure 3-7. Dynamic topology in ad hoc networks

3.4.1.2 Murphy Definition of Ad hoc Network

According to [Murphy et al., 1998], an ad hoc network is "*a transitory association of mobile nodes which do not depend upon any fixed support infrastructure. Participants at a conference and disaster relief workers may find it necessary to interact with each other in this manner when the static support infrastructure is not available. An ad hoc network can be visualized as a continuously changing graph. Connection and disconnection is controlled by the distance among nodes and by willingness to collaborate in the formation of cohesive, albeit transitory community*".

Distance among nodes, With ad hoc networks, no additional infrastructure is required beside the network nodes themselves. It is the distance among nodes, or rather their *adjacency*, that defines the boundaries of the network. That means, the only arranging of two or more mobile nodes within a certain boundary defines a new network in an *ad hoc* manner. Now, if the nodes were not mobile, an ad hoc network would not be different from LAN (Local Area Network). So, it is also the mobility of nodes, causing variations in their distance that gives such networks their ad hoc nature.

The actual boundary defining an ad hoc network really depends on the technology being used. Some ad hoc network solutions are limited to Personal Area Network (PAN). The Bluetooth technology, for instance, is allowed only for PAN (up to about 10 meters) [Kardash, 2000].

Willingness to collaborate, The collocation of several nodes within a certain distance is a necessary but not sufficient condition to form an ad hoc network. In addition, collocated nodes need to be willing to collaborate. By definition of ad hoc networks, this willingness is

expressed at the *network level*. The decision to collaborate or not is expressed by going online or offline.

Transitory peer-to-peer communities, Intuitively, the above features of ad hoc networks characterize their somehow "*here and now*" nature. That means, at any point in time (*now*), the network is defined by all nodes that are both within a certain distance and online (*here*). As consequence, nodes tend to appear and disappear in an ad hoc network much more often than in other types of networks, leading to so-called *transitory community*. People can join and withdraw from the community at any time.

3.4.2 Mobile Ad hoc Network Graph

A MANET topology can also be defined as a dynamic (arbitrary) multi-hop graph $G = (N, L)$, where N is a finite set of mobile nodes MNs and L is a set of edges which represent wireless links. A link $(i, j) \in L$ exists if and only if the distance between two mobile nodes is less or equal than a fixed radius r as shown in Figure 3-8. This r represents the radio transmission range that depends on wireless channel characteristics including transmission power. Accordingly, the neighborhood of a node x is defined by the set of nodes that are inside a circle (assume that MNs are moving in a two-dimensional plane) with center at x and radius r, and it is denoted by $N_r(x) = N_x = \{n_j | d(x,n_j) \leq r, x \neq n_j, \forall j \in N, j \leq |N|\}$, where x is an arbitrary node in graph G and d is a distance function [Nikaein et al., 2000].

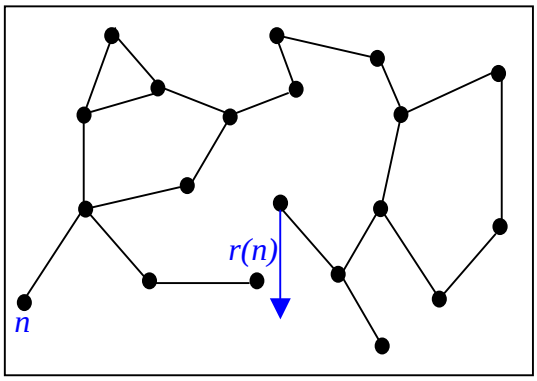


Figure 3-8. The graph of a wireless ad hoc network

A *path* (*route*) from node i to node j, denoted by R_{ij} is a sequence of nodes $R_{ij}=(i,n_1, n_2,\ldots,n_k, j)$ where (i,n_1), (n_k,j) and (n_y,n_{y+1}) for $1 \leq y \leq k$-1 are links. A *simple path* from i to j is a sequence of nodes with no node being repeated more than once. Due to the mobility of the nodes, the set of paths (wireless links) between any pair of nodes and distances is changing over time. New links can be established and existing links can vanish.

3.4.3 Example of An Ad hoc Network

Imagine a scenario as in *Tsunami disaster* relief operations wherein timely communication is a very important factor, the relief workers come in the area and without the need of any existing infrastructure, just switch on their handsets and start communicating with each other while moving and carrying out rescue work. In this case of rescue problems, for example in scenes of natural disasters, an ad hoc network could be formed by communication devices in fire brigades, helicopters, ambulances, policies and also people with laptop computers or mobile phones in hospitals, pharmacies and so on, all together work in a collaborative way to provide effective solutions to the problem. Figure 3-9 shows an example of an ad hoc network which has different communication devices and some connections amongst them (when devices are in the same transmission range).

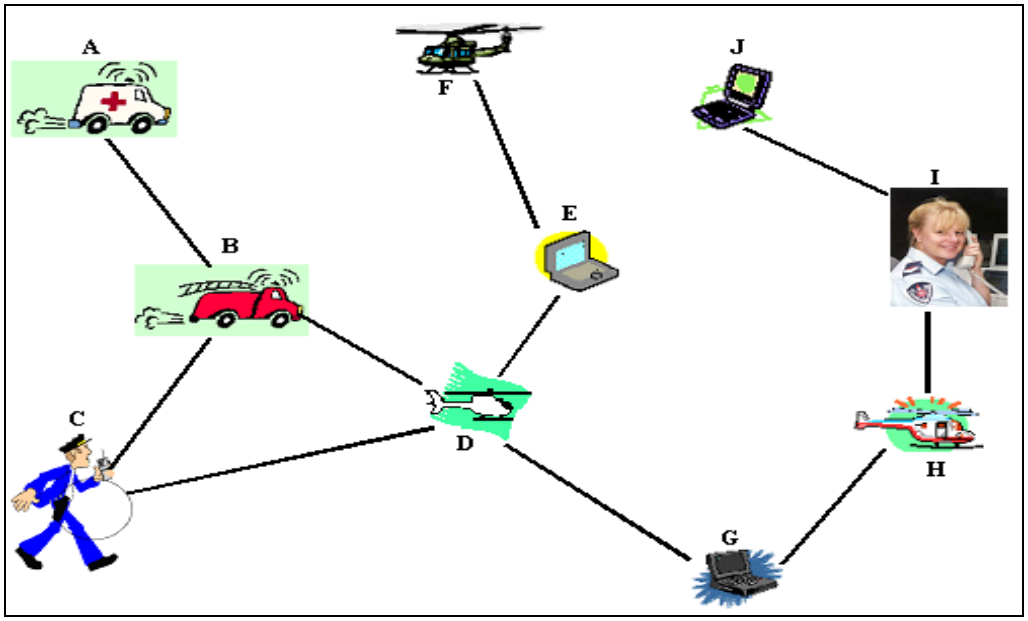


Figure 3-9. Representation of a mobile ad hoc network to solve rescue problems

It is obvious that they need to communicate with each other; however the high degree of mobility in this kind of network makes them change quickly. Some devices could be out of range with respect to others and therefore each device (node) must be able to act as a router to relay packets generated by other nodes. In Figure 3-10 when node *G* needs to communicate with node *I*, node *H* has to act as a router and transmit its information. But, what happen when node *H* is out of range with respect to node *I*?. The possibility that node *E* is now in range of node *I* and the topology of the network has changed to different one may exist as it is shown in Figure 3-10. In this new topology, node *G* can communicate with node *I* through nodes *D*, *E* acting as routers.

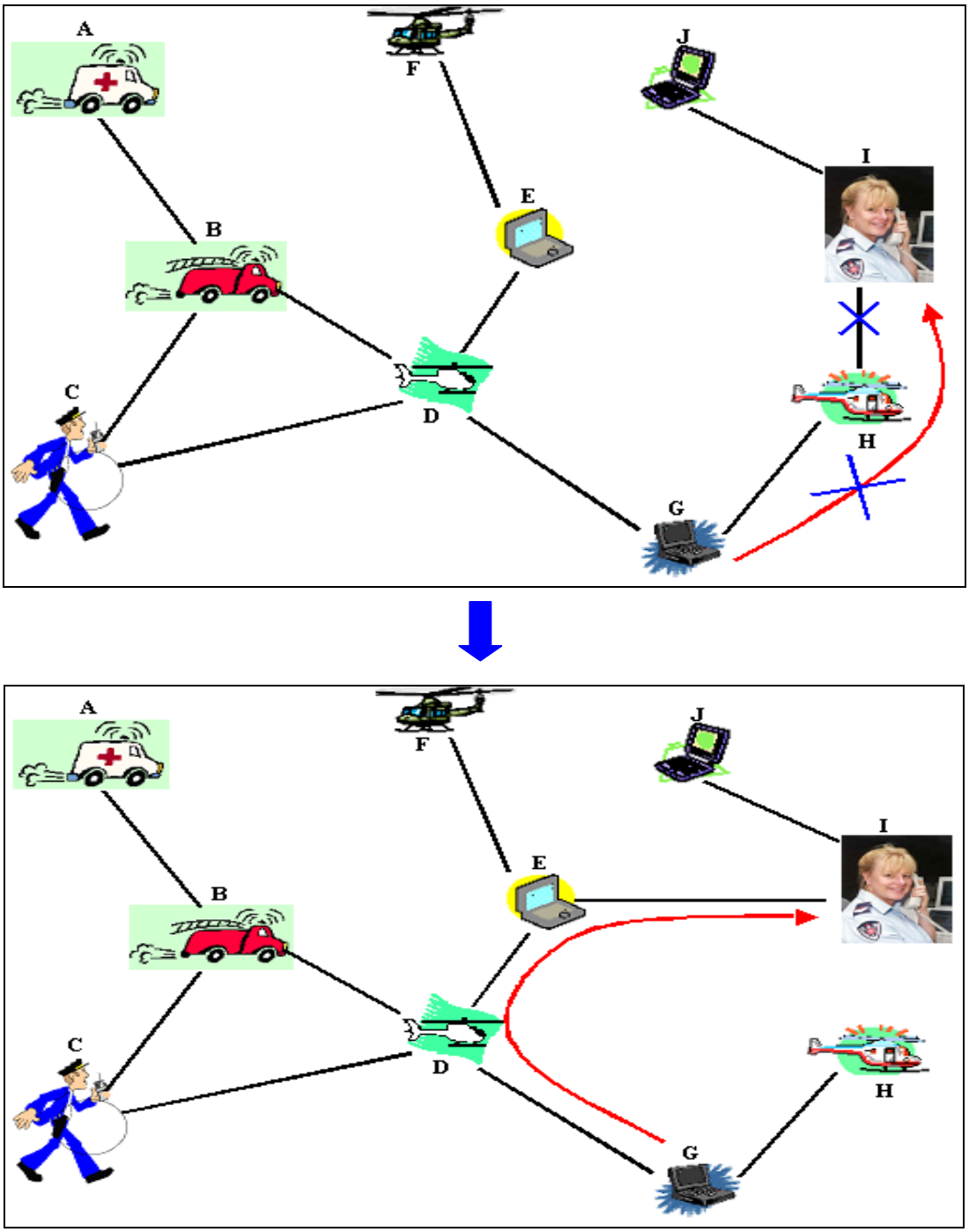


Figure 3-10. Representation of a new network topology in a case of rescue problems

Then, we can see how difficult the tasks of routing and maintaining paths in MANETs are and how much they are affected because of the node mobility, transmission signal, limitations on the battery power of mobile nodes and moreover limitations on the resources in terms of bandwidth of the wireless medium and the fact of sharing this medium (most commonly controlled by the IEEE 802.11 Medium Access Control (MAC) protocol).

3.5 Classification of Ad hoc Networks

There is no general recognized classification of ad hoc networks in the literature. However a classification on the basis of the network types treated in the literature can be executed. In the following, ad hoc networks are classified according to three different aspects [Günes, 2004].

3.5.1 Classification According to Communication

This simple classification is based on the formation and type of communication.

3.5.1.1 Single-hop Ad hoc Network

Nodes are in their reachable area and can communicate directly as shown in Figure 3-11. Single-hop ad hoc networks are the simplest type of ad hoc networks where all nodes are in their mutual range, that means the individual nodes can communicate directly together, without any help of other intermediate nodes. One could call this type of networks also plug and play networks, since it concerns mainly the simple and fast structure from temporary connections.

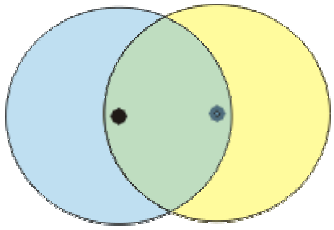


Figure 3-11. Single-hop ad hoc network

With this class, the mobility does not play a role. The individual nodes do not have to be static, they must remain however within the range of all nodes, that means the entire network could move as group, which would not modify in the communication relations anything.

3.5.1.2 Multi-hop Ad hoc Network

This class in the literature is the most examined type of ad hoc networks. It differs from the first class in that, some nodes are far and cannot communicate directly. Therefore, the traffic of these communication end-points has to be forwarded by other intermediate nodes. Figure 3-12 shows the communication path of far nodes as black lines. With this class also, one assumes that the nodes are mobile. The basic difficulty of the networks of this class is the node mobility, whereby the network topology is subjected to continous modifications.

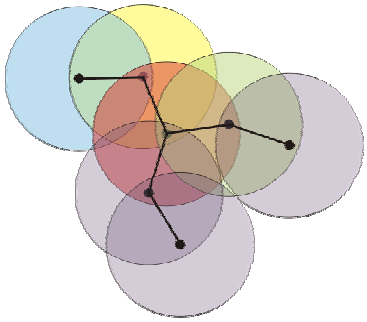


Figure 3-12. Multi-hop ad hoc network

The general problem in networks of this class is the assignment of a routing protocol. High performance routing protocols must be adaptive to the fast topology modification.

3.5.1.3 Mobile Multi-hop Multimedia Ad hoc Network (3M-Network)

This class is an extension of the second class with the difference, that communication traffic consists here mainly of real time data such as audio and video. The networks of this class are also defined as 3M-Network [Bouazizi and Günes, 2002] [Bouazizi and Günes, 2003]. The quality requirements play the main role in this class of networks.

3.5.2 Classification According to Topology

Ad hoc networks can be classified according to the network topology. The individual nodes in an ad hoc network are divided into different types with special functions. There are three different classes: *flat*, *hierarchical* and a*ggregate ad hoc networks*.

3.5.2.1 Flat Ad hoc Networks

In *flat ad hoc networks*, all nodes carry the same responsibility and there is no distinction between the individual nodes as indicated in Figure 3-13. All nodes are equivalent and can transfer all functions in the ad hoc network.

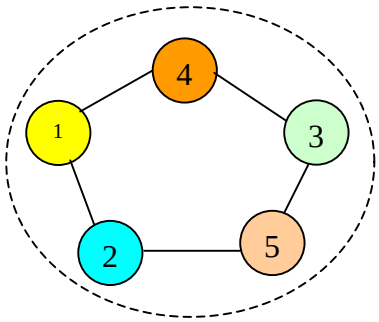


Figure 3-13. Flat ad hoc network

Control messages have to be transmitted globally throughout the network, but they are appropriate for highly dynamic network topology. The scalability decreases when the number of nodes increases significantly.

3.5.2.2 Hierarchical Ad hoc Networks

Hierarchical ad hoc networks consist in this case of several clusters, each one represents a network and all are linked together as indicated in Figure 3-14. The nodes in hierarchical ad hoc networks can be differentiated into two types:

- Master nodes: administer the cluster and are responsible for passing the data on to other cluster.
- Normal nodes: Communicate within the cluster directly together and with nodes in other clusters with the help of the master node. Normal nodes are called also *slave* nodes.

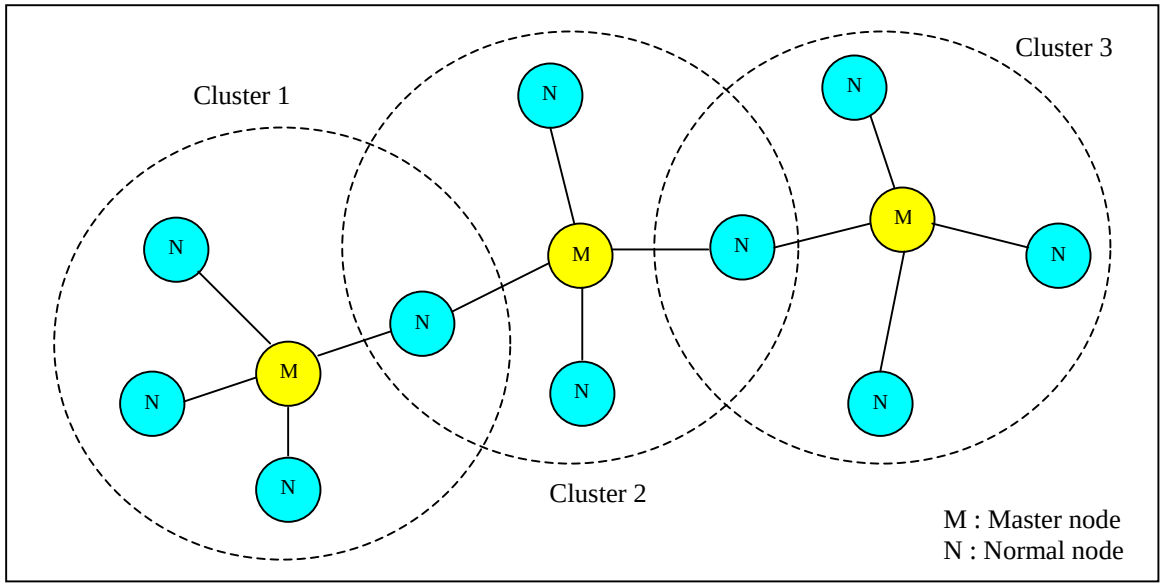


Figure 3-14. Hierarchical ad hoc network

One assumes, that the majority of communication (control messages) takes place within the cluster and only a fraction between different clusters. During communication within a cluster no forwarding of communication traffic is necessary. The master node is responsible for the switching of a connection between nodes in different clusters.

The flat architecture has the following advantages over the hierarchical:

- Increased reliability and survivability.
- No single point of failure.
- Alternative routes in the network.
- More optimal routing.
- Better coverage, i.e. reduced use of the wireless resources.

- Route diversity, i.e. better load balancing property.
- All nodes have one type of equipment.

The *no single point of failure* is of great importance for a message to reach its destination. This means that if one node goes down, the rest of the network will still function properly. In the hierarchical approach this is altogether another matter. If one of the clusterheads goes down, that section of the network won't be able to send or receive messages to other sections for the duration of the downtime of the clusterhead.

Hierarchical architectures are more suitable for low mobility case. Although flat architectures are more flexible and simpler than hierarchical ones, hierarchical architectures provide a more scalable approach.

3.5.2.3 Aggregate Ad hoc Networks

Aggregate ad hoc networks bring together a set of nodes into zones. Therefore, the network is partitioned into a set of zones as shown in Figure 3-15. Each node belongs to two levels topology: low level (node level) topology and high level (zone level) topology. Also, each node may be characterized by two ID numbers: node ID number and zone ID number. Normally, aggregate architectures are related to the notion of *zone*. In aggregate architectures, we find both intra-zone and inter-zone architectures which in turn can either support flat or hierarchical architectures.

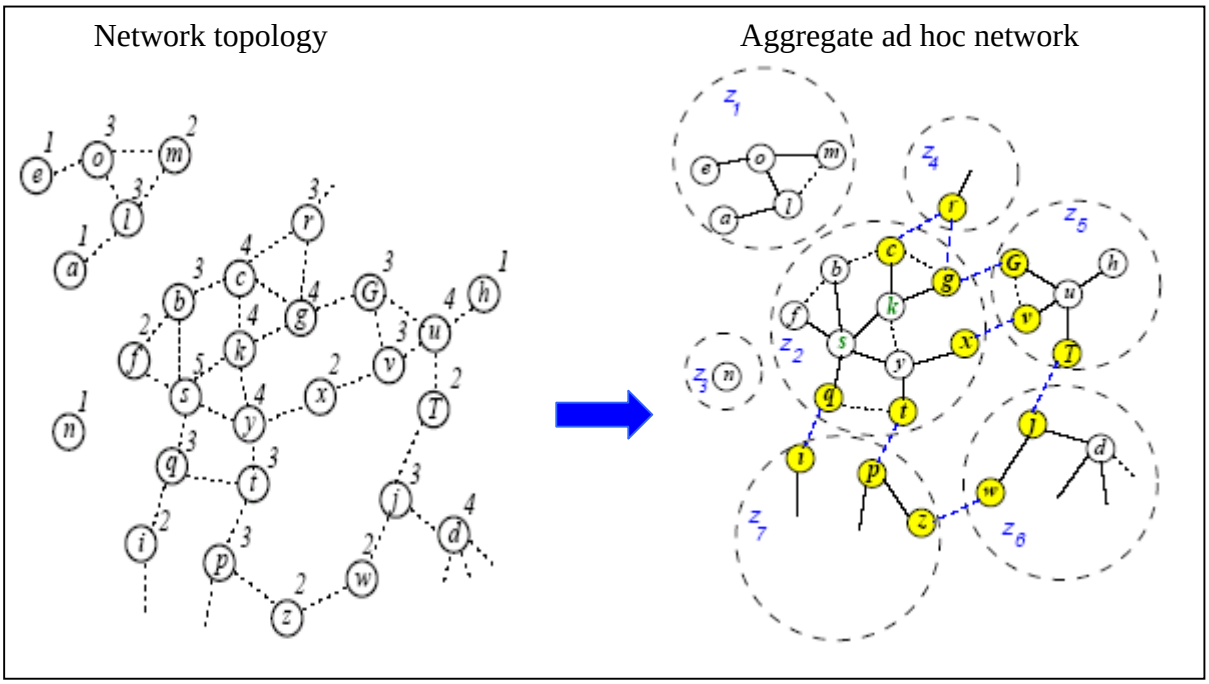


Figure 3-15. Aggregate network architecture

3.5.3 Classification According to Node Configuration

Further classification of ad hoc networks can be executed on the basis of the hardware configuration of the nodes [Toh, 2002]. The configuration of the nodes in a mobile ad hoc network is important and can depend very strongly on the actual application.

3.5.3.1 Homogeneous Ad hoc Networks

In homogeneous ad hoc networks, all nodes possess the same characteristics regarding the hardware configuration as processor, memory, display and peripheral devices. Most well known representatives of homogeneous ad hoc networks are wireless sensor networks. In homogeneous ad hoc networks, applications can proceed from certain prerequisites; for example, the localization is considerably facilitated by the presence of control components in each node.

3.5.3.2 Heterogeneous Ad hoc Networks

In heterogeneous ad hoc networks, the nodes differ according to the hardware configuration. Each node has different characteristics, resources and policies. In ad hoc networks of this class, all nodes cannot provide the same services.

3.5.4 Classification According to Coverage Area

As shown in Figure 3-16, we can classify ad hoc networks, depending on their coverage area, into several classes: Body (BAN), Personal (PAN), Local (LAN), Metropolitan (MAN) and Wide (WAN) area networks [Chlamtac et al., 2003].

Wide and Metropolitan area ad hoc networks are mobile multi-hop wireless networks presenting many challenges that are still being solved (e.g., addressing, routing, location management, security, etc.) and their availability is not on immediate horizon. On the other hand, mobile ad hoc networks with smaller coverage can be expected to appear soon. Specifically, ad hoc single-hop BAN, PAN and LAN wireless technologies are already common on the market, these technologies constitute the building blocks for constructing small multi-hop ad hoc networks that extend their range over multiple radio hops. For these reasons, BAN, PAN and LAN technologies constitute the enabling technologies for ad hoc networking.

A body area network is strongly correlated with wearable computers. A wearable computer distributes on the body its components (e.g., head-mounted displays, microphones, earphones,

etc.), and the BAN provides the connectivity among these devices. The communication range of a BAN corresponds to the human body range, i.e., 1–2 m. As wiring a body is generally cumbersome, wireless technologies constitute the best solution for interconnecting wearable devices.

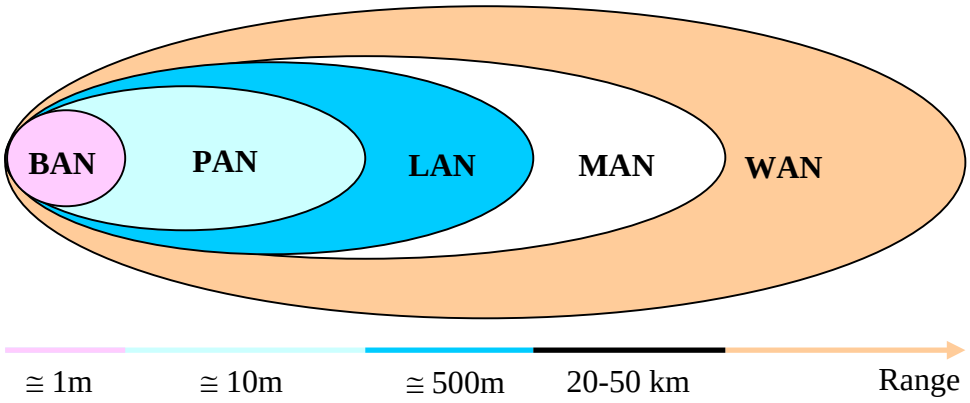


Figure 3-16. Ad hoc networks taxonomy according to coverage area

Personal area networks PAN connect mobile devices carried by users to other mobile and stationary devices. While a BAN is devoted to the interconnection of one-person wearable devices, a PAN is a network in the environment around the persons. A PAN communicating range is typically up to 10 m, thus enabling the interconnection of the BANs of persons close to each other, and the interconnection of a BAN with the environment around it. The most promising radios for widespread PAN deployment are in the 2.4 GHz ISM band. Spread spectrum is typically employed to reduce interference and bandwidth re-use.

Wireless LANs (WLANs) have a communication range typical of a single building, or a cluster of buildings, i.e., 100–500 m. A WLAN should satisfy the same requirements typical of any LAN, including high capacity, full connectivity among attached stations and broadcast capability. However, to meet these objectives, WLANs need to be designed to face some issues specific to the wireless environment, like security on the air, power consumption, mobility and bandwidth limitation of the air interface.

Two different approaches can be followed in the implementation of a WLAN: an infrastructure-based approach, or an ad hoc networking one. An infrastructure-based architecture imposes the existence of a centralized controller for each cell, often referred to as Access Point. The Access Point (AP) is normally connected to the wired network, thus providing the internet access to mobile devices. In contrast, an ad hoc network is a peer-to-peer network formed by a set of stations within the range of each other, which dynamically configure themselves to set up a temporary network. In the ad hoc configuration, no fixed

controller is required, but a controller may be dynamically elected among the stations participating in the communication. The success of a network technology is connected to the development of networking products at a competitive price.

The ad hoc network size in terms of the number of active nodes is the other metric used to classify MANETs. We can classify the scale of an ad hoc network as small scale (i.e., 2–20 nodes), moderate scale (i.e., 20–100 nodes), large scale (i.e., 100+ nodes) and very large-scale (i.e., 1000+ nodes).

3.6 Different States of Ad hoc Networks

We need such networks because the backbone is not present. For some reasons, may be there is no existing backbone or too expensive to build or not deployable in the required *time frame*. An ad hoc network has a temporary purpose which is only executed in a certain time frame.

Some of the situations which motivate the construction of ad hoc networks:

1. A class of students may need to interact during a lecture,
2. Any number of people entering a conferencing room want to communicate with each other,
3. A group of emergency rescue workers may need to be quickly deployed after a natural disaster such as earthquake or flood,
4. A group of friends or business associates run into each other at an airport terminal and wish to share information,
5. Ships need to communicate with each other.

The common points for above are people moving, people exchanging information with each other and an ad hoc network is existing only for a temporary purpose (It does not need to be there for a long time). Thus,

1. It does not need to be power-efficient.
2. It does not bother to spend much overhead for routing if there is not too much stable traffic to transport.
3. Topology must change, at least in some area in a certain time but not limited to any area of the network.

A *taxonomy states* of mobile ad hoc networks can be classified based on the rate of connections/disconnections as follows:

(i) *stable* with a low rate of disconnections and connections;

(ii) *high connection* with a high rate of connections and low rate of disconnections;

(iii) *high disconnection* with a high rate of disconnections and low rate of connections; or

(iv) *highly dynamic* with high rate of disconnections as well as connections.

The topology can be highly dynamic in some regions and stable in other regions, at least during certain periods in the most general case. A single network may operate in different states at different times, or different regions of the network may operate in different modes at the same time. For example, consider a mobile robot team exploring a reinforced building suspected of possible nuclear contamination. Initially, when the robots are deployed at an entry point, the network is in *stable state.* As the robots disperse to explore various rooms, the network enters a *high disconnection state,* and as they move in and out of rooms located along a corridor the network enters a *highly dynamic state*. Toward the end of the exploration, when the robot reassemble at the entry point, the network will be in *high connection state.*

3.7 Properties of Mobile Ad hoc Networks

MANETs have the following special features that should be considered in designing solutions for this kind of networks.

- **Dynamic Topology**

Due to the node mobility, the topology of mobile multi-hop ad hoc networks changes continuously and unpredictably. The link connectivity among the terminals of the network dynamically varies in an arbitrary manner and is based on the proximity of one node to another node. It is also subjected to frequent disconnection during node's mobility. MANET should adapt to the traffic and propagation conditions as well as to the mobility patterns of the mobile network nodes. The mobile nodes in the network dynamically establish routing among themselves as they move about, forming their own network on the fly. Moreover, a user in the MANET may not only operate within the ad hoc network, but may require access to a public fixed network.

- **Bandwidth**

MANETs have significantly lower bandwidth capacity in comparison with fixed networks. The used air interface has higher bit error rates, which aggravates the expected link quality. Current technologies suitable for the realization of MANETs are IEEE 802.11(b,a) with bandwidth up to 54Mbps and Bluetooth providing bandwidth of 1Mbps. The nature of high bit-error rates of wireless connection might be more profound in a MANET. One end-to-end path can be shared by several sessions. The channel over which the terminals communicate is subjected to noise, fading and interference, and has less bandwidth than a wired network. In some scenarios, the path between any pair of users can traverse multiple wireless links and the links themselves can be heterogeneous.

- **Energy**

All mobile devices will get their energy from batteries, which is a scarce resource. Therefore the energy conservation plays an important role in MANETs. This important resource has to be used very efficiently. One of the most important system design criteria for optimization may be energy conservation.

- **Security**

The nodes and the information in MANETs are exposed to the same threats like in other networks. Additionally to these classical threats, in MANETs there are special threats, e.g. denial of service attacks. Also mobility implies higher security risks than static operation because portable devices may be stolen or their traffic may insecurely cross wireless links. Eavesdropping, spoofing and denial of service attacks should be considered.

- **Autonomous**

No centralized administration entity is required to manage the operation of the different mobile nodes. In MANET, each mobile terminal is an autonomous node, which may function as both a host and a router. So usually endpoints and switches are indistinguishable in MANET.

- **Distributed Operation**

Since there is no background network for the central control of the network operations, the control and management of the network is distributed among the terminals. The nodes involved in a MANET should collaborate among themselves and each node acts as a relay as needed, to implement functions e.g. security and routing.

- **Multi-hop Routing**

Basic types of ad hoc routing algorithms can be single-hop and multi-hop, based on different link layer attributes and routing protocols. Single-hop MANET is simple in comparison with multi-hop MANET in terms of structures and implementation. When delivering data packets from a source to its destination out of the direct wireless transmission range, the packets should be forwarded via one or more intermediate nodes.

- **Light-Weight Terminals**

In most cases, the MANET nodes are mobile devices with less CPU processing capability, small memory size and low power storage.

- **Infrastructureless and Self Operated**

A mobile ad hoc network includes several advantages over traditional wireless networks, including: ease of deployment, speed of deployment and decreased dependence on a fixed

infrastructure. MANET is attractive because it provides an instant network formation without the presence of fixed base stations and system administrators.

3.8 Ad hoc Network Applications

Generally, ad hoc networks are suitable for all situations in which a temporary communication is desired or in case the building of network infrastructure not possible. There are many applications to ad hoc networks, such as home networks, group discussions, vehicle communications as indicated in Figure 3-17, Personnel Area Network (PAN) for communication of several portable devices and emergency situations. A list of different applications are indicated in appendix A.

As a matter of fact, any day-to-day application such as electronic email and file transfer can be considered to be easily deployable within an ad hoc network environment. Web services are also possible in case any node in the network can serve as a gateway to the outside world. In this discussion, we need not emphasize the wide range of possible military applications using ad hoc networks.

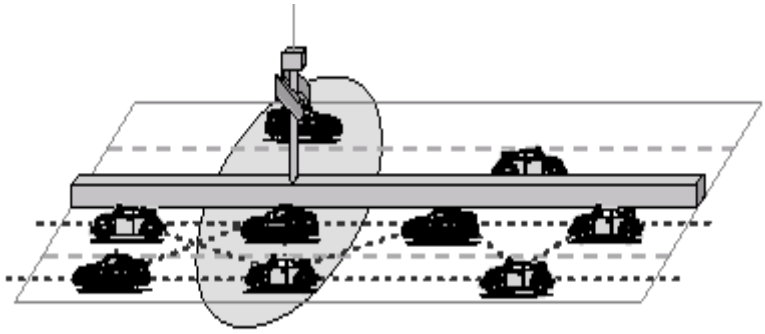


Figure 3-17. Mobile ad hoc network on the road

The technology was initially developed keeping in mind the military applications, such as battlefield in an unknown territory where an infrastructured network is almost impossible to have or maintain. In such situations, the ad hoc networks having self-organizing capability can be effectively used where other technologies either fail or cannot be effectively deployed. Advanced features of wireless mobile systems, including data rates compatible with multimedia applications, global roaming capability and coordination with other network structures, are enabling new applications.

Another ad hoc network scenario may occur at an airport where people carry devices that can be connected based on an ad hoc network. Users' devices can both interconnect with one another and connect to local information point, for instance, in order to obtain updated information on flight departures, gate changes and so on. The ad hoc devices can also forward

traffic on behalf of two devices that are out of each other's range. Thus, this airport scenario includes a mixture of single and multiple radio hops [Frodigh et al., 2000].

The following are some well-known ad hoc network applications:

- **Collaborative Work – Conferencing**

Perhaps the prototypical application requiring the establishment of an ad hoc network is mobile conferencing. For some business environments, the need for collaborative computing might be more important outside office environments than inside. Today's project environments are heavily computerized. For projects in a very broad range of industries, the need for creating an ad hoc network seems clear.

- **Crisis and Catastrophe Scenario Management Applications**

A very popular scenario in the literature is the use of ad hoc networks in emergency situations. These arise, for example, as a result of a large natural disasters like an earthquake where the entire communications infrastructure is in disarray and the communication medium (cable, glass fiber) is partially or completely damaged. Restoring communications quickly is essential. By using ad hoc networks, the communications could be set up in hours instead of days/weeks required for wire-line communications.

- **Personal Area Networking and Bluetooth**

A Personal Area Network (PAN) is a short-range, localized network where nodes are usually associated with a given person. These nodes could be attached to someone's pulse watch, belt and so on. In these scenarios, mobility is only a major consideration when interaction among several PANs is necessary. Bluetooth [Haarsten, 1998] is a technology aimed at, among other things, supporting PANs by eliminating the need of wires between devices such as printers, PDAs, notebook computers, digital cameras and so on, and is discussed later.

3.9 Routing Definition and Basic Functions

A routing protocol is the mechanism by which user traffic is directed and transported through the network from the source node to the destination node. Objectives include maximizing network performance from the application point of view (application requirements) while minimizing the cost of network itself in acordance with its capacity. The application requirements are hop count, delay, throughput, loss rate, stability, jitter, cost, etc. The network capacity is a function of available resources at each node, density (i.e. number of nodes), frequency of end-to-end connection (i.e. number of communication) and frequency of topology changes (mobility rate).

Here is the basic routing functionality for mobile ad hoc networks:

- *Path generation* which generates paths according to the assembled and distributed state information of the network and of the application;
- *Path selection* which selects appropriate paths based on network and application state information;
- *Data Forwarding* which forwards user traffic along the selected route;
- *Path Maintenance* which is responsible for maintaining the selected path.

Consequently routing is bounded by traffic requirements and network capacity. This is illustrated in Figure 3-18.

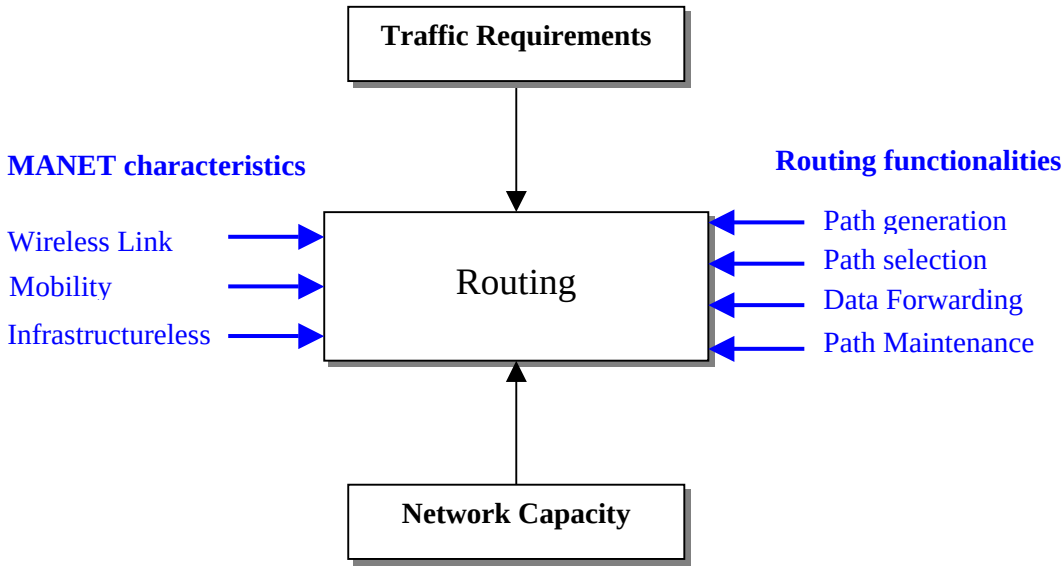


Figure 3-18. Routing and basic functions

Routing in MANETs is a difficult problem due to the bandwidth, energy constraints and rapidly changing topologies. In mobile ad hoc routing, each node acts as a router and a number of nodes cooperate. Routing is multi-hop, so that data packets are forwarded from the source by a number of nodes before reaching the destination.

3.9.1 Classification of Routing Algorithms

Routing algorithms are classified as follows [Di Caro and Dorigo, 1998]:

- *Centralized* versus *Distributed*
- *Static* versus *Dynamic*.
- *Proactive* versus *Reactive*
- *Unicast* versus *Multicast*

In *centralized* routing algorithms, a main controller is responsible for updating all the nodes' routing tables and/or to make every routing decision. Centralized algorithms can be used only in particular cases and for small networks.

In *distributed* routing algorithms, the computation of routes is shared among the network nodes, which exchange the necessary information. The distributed paradigm is currently used in the majority of network systems.

In *static* routing systems, the path taken by a packet is determined only on the basis of its source and destination, without regard to the current network state. This path is usually chosen as the shortest one according to cost criterion.

In *dynamic* routing systems, the routing policy can adapt to time and varying traffic conditions. As a drawback, they can cause oscillations in selected paths. This fact can cause circular paths, as well as large fluctuations in measured performance.

In a *proactive* routing protocol, all the routes to each destination are kept in an up-to-date table. Changes in the network topology are continually updated as they occur. The differences between the protocols are in how the changes are spread through the network and how many tables each node maintains.

In the *reactive* approach, a connection between two nodes is only created when it is asked for by a source. When a route is found, it is kept by a route maintenance procedure until the destination no longer exists or is not needed anymore.

In case of *multicast* routing, a single packet is sent simultaneously to multiple recipients, but in *unicast* routing, a single packet is only sent to one recipient every transmission. Thus, the multicast method is a very efficient and useful way to support group communication when bandwidth is limited and energy is constrained.

3.10 Introduction in MANETs Routing

A special form of routing protocols is necessary in case of ad hoc networks, since two hosts wishing to exchange packets may not be within wireless transmission range of each other (i.e. not be able to communicate directly), but could communicate if other mobile nodes between them are also participating in the ad hoc network and are willing to forward packets for them.

For example, Figure 3-19 illustrates a mobile ad hoc network with three wireless mobile hosts. Node 3 is not within the range of node 1's wireless transmitter (indicated by the circle around node 1) and vice versa. If node 1 and node 3 wish to exchange packets, they must enlist the services of node 2 to forward packets for them, since node 2 is within the range overlap between node 1 and node 3.

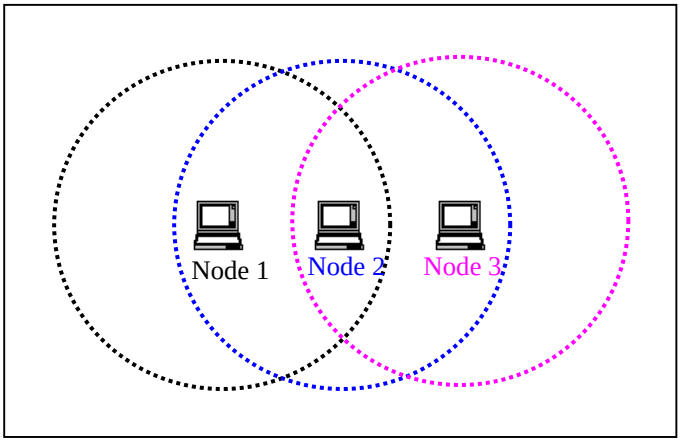


Figure 3-19. Exchange packets in a three mobile nodes ad hoc network

Each time a packet is transmitted to a neighboring node, it is said to have made a *hop*. In the above example, when node 1 sends a packet to node 3, the packet makes two hops: first from node 1 to node 2, and second from node 2 to node 3.

The situation in Figure 3-19 becomes more complicated with the addition of more nodes. The addition of just one node, as illustrated in Figure 3-20, results in multiple paths existing between nodes 1 and 3; packets may travel the path 1 - 2 - 3, 1 - 4 - 3, or even 1 - 4 - 2 - 3. An ad hoc routing protocol must be able to decide on a single "best" path between any two nodes.

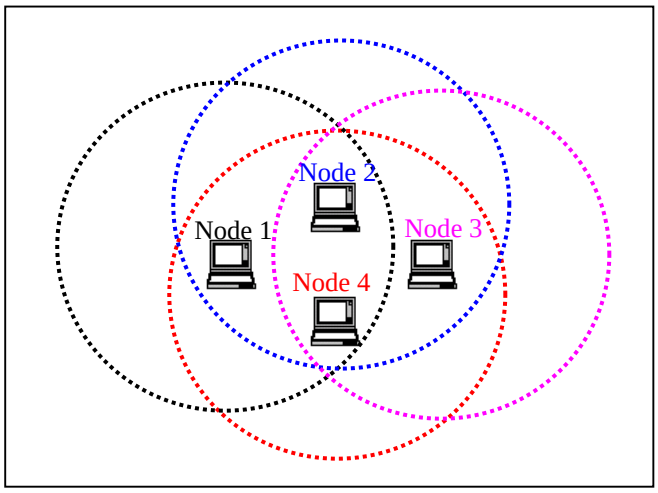


Figure 3-20. Exchange packets in a four mobile nodes ad hoc network

Sometimes in a network of mobile nodes, parts of the network may become isolated and separate. This effect is called network *partitioning* and is illustrated in Figure 3-21. When network partitioning occurs, nodes can continue to exchange information within the same partition, but it is impossible for information to flow between separated partitions.

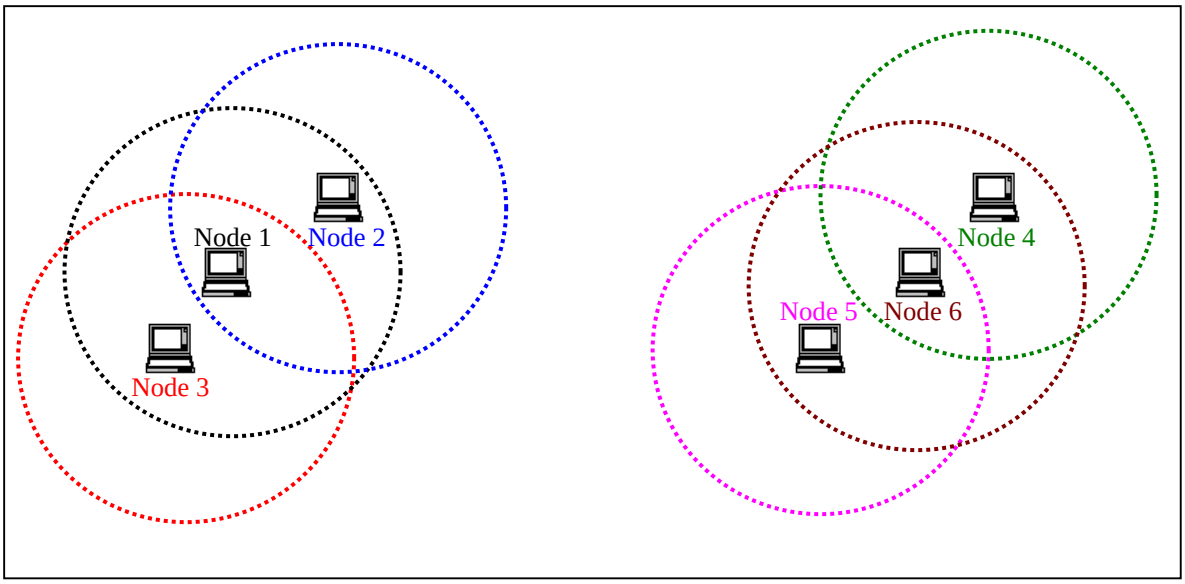


Figure 3-21. Ad hoc network partitioned into two separate networks

Another unique situation of wireless networks is illustrated in Figure 3-22. In this example, node 1 has a large enough range to transmit packets directly to node 3. However, node 3 has a much smaller range and must enlist the help of node 2 in order to return packets to node 1. This makes the link between node 1 and node 3 appears as a one-way or unidirectional link. Most conventional routing protocols require bi-directional links.

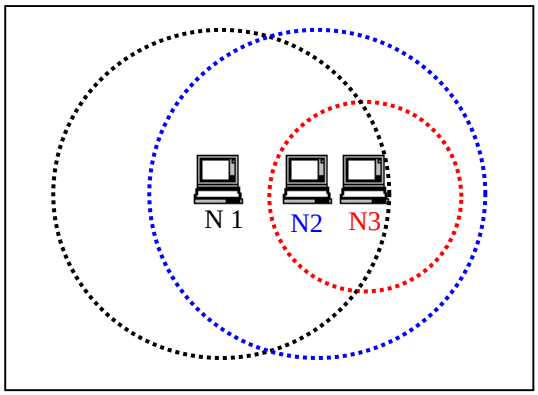


Figure 3-22. Ad hoc network with a one-way link

In summary, routing protocols for ad hoc networks face several challenges. Ad hoc networks require multi-hop forwarding to facilitate information exchange. Wireless links are nonuniform and can cause unidirectional links. By their nature, mobile nodes tend to "wander around", changing their network location and link status on a regular basis, sometimes partitioning the network. Furthermore, new nodes may unexpectedly join the network or existing nodes may leave or be turned off. It is the goal of ad hoc routing protocols to provide network connectivity despite these challenges.

Ad hoc routing protocols must minimize the time required to converge after these topology changes. A low convergence time is more critical in ad hoc networks because temporary routing loops can result in transmitting packets in circles, further consuming valuable bandwidth.

3.11 Challenges in Mobile Ad hoc Network Routing

Some of the important factors that make routing in mobile ad hoc networks with wireless links challenging are:

- **Frequent Topology Changes**

All nodes in a MANET are mobile; this means that the topology is dynamic and routes that existed may not exist some time later due to the movement of the intermediate nodes. To alleviate this problem, the protocol should be resilient to topology changes. Reactive protocols that try to mend disconnected paths incur large control overhead and yet packets are lost during the reconstruction phase. Proactive protocols constantly update their routes but the constant overhead generated proves to be expensive in cases where the movement among the nodes is limited.

- **Frequent and Unpredictable Connectivity Changes**

MANETs are expected to be used in hostile environments such as battlefields and emergency operations. As such, any routing protocol should not only be able to deal with frequent topology changes but also with unpredictable connectivity changes for instance due to changing weather patterns or varied geographies. Wireless links behave unpredictably in these situations.

- **Bandwidth Constrained and Variable Capacity Links**

MANETs use wireless links that offer limited bandwidth for transmission. Further, these links may be of variable capacity. This presents one of the most important challenges to any routing protocol - keeping the bandwidth usage to a minimum even while responding reactively to changes in topology or proactively maintaining routes. Further, this also presents challenges in terms of the optimization of network routes in the MANET environment.

3.12 Why We Need New Routing Protocols?

In ad hoc networks, we need new routing protocols because of the following reasons:

1. Nodes in ad hoc networks are mobile and the topology of interconnections between them may be quite dynamic.
2. Existing protocols exhibit least desirable behavior when presented with a highly dynamic interconnection topology.
3. Existing routing protocols place a too heavy computational burden on each mobile computer in terms of the memory-size, processing power and power consumption.
4 Existing routing protocols are not designed for dynamic and self-starting behavior as required by users wishing to utilize ad hoc networks.
5 Existing routing protocols like the Distance Vector Protocol take a lot of time for convergence upon the failure of a link, which is very frequent in ad hoc networks.
6. Existing routing protocols suffer from looping problems either short lived or long lived.
7. Methods adopted to solve looping problems in traditional routing protocols may not be applicable to ad hoc networks.

3.13 Features Desired for a Routing Protocol in Ad hoc Networks

The protocols to be used in the ad hoc networks should have the following features:

1. The protocol should adapt quickly to topology changes.
2. The protocol should provide loop free routing.
3. The protocol should provide multiple routes from the source to destination and this will solve the problems of congestion to some extent.
4. The protocol should have minimum control message overhead due to exchange of routing information when topology changes occur.
5. The protocol should allow quick establishment of routes so that they can be used before they become invalid.

3.14 Routing Protocols Issues

One of the issues with routing in ad hoc networks concerns about whether nodes should keep track of routes to all possible destinations, or instead keep track of only those destinations that are of immediate interest. A node in an ad hoc network does not need a route to a destination until that destination is to be the recipient of packets sent by the node, either as the actual source of the packet or as an intermediate node along a path from the source to the destination.

Several routing protocols have been proposed for mobile ad hoc network regarding application requirements and network properties. A comparison of different existing

approaches and examination of the main strengths of each tendency can be found in [Rover and Toh, 1999][Lee et al., 1999][Broch et al., 1998].

In [Nikaein et al., 2001] the routing design issues for mobile ad hoc network are classified according to four criteria as indicated in Figure 3-23:

1. Routing Philosophy: *Table-driven* versus *on-demand* versus *hybrid approach,*

2. Routing Architecture: *Flat* versus *hierarchical* versus *aggregate architecture,*

3. Routing Information: *Global Position* versus *Global Position-Less based protocols,*

4. Routing Generation: *First generation* versus *second generation* versus *third generation*.

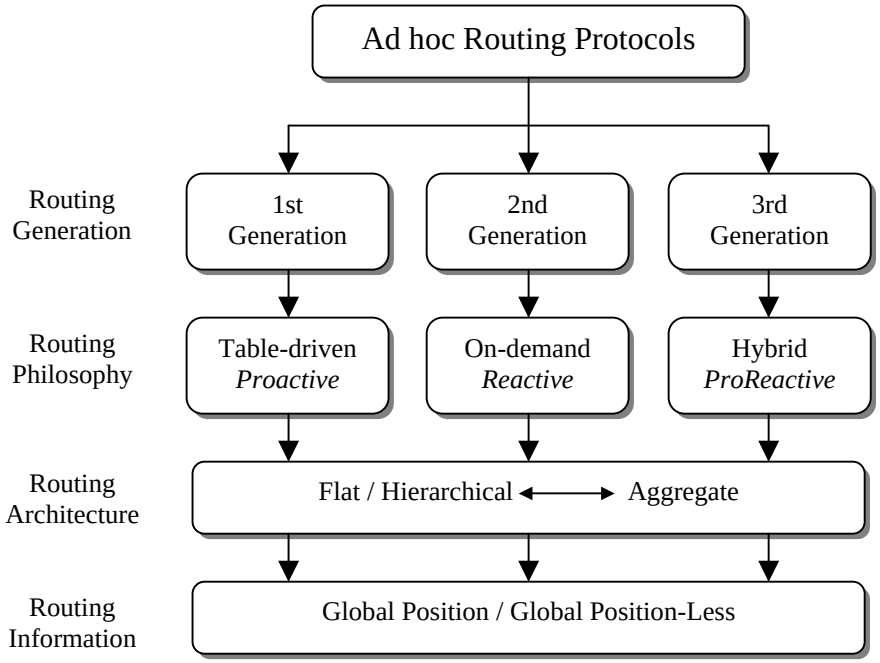


Figure 3-23. Classification of ad hoc network routing protocols

3.14.1 Routing Philosophy

- **Table Driven Routing Protocols (Proactive)**

In proactive or table-driven routing protocols, each node continuously maintains up-to-date routes to every other node in the network. Routing information is periodically transmitted throughout the network in order to maintain routing table consistency. Thus, if a route has already existed before traffic arrives, transmission occurs without delay. Otherwise, traffic packets should wait in queue until the node receives routing information corresponding to its destination. However, for highly dynamic network topology, the proactive schemes require a significant amount of resources to keep routing information up-to-date and reliable.

Proactive protocols suffer the disadvantage of additional control traffic that is needed to continually update stale route entries. Since the network topology is dynamic, when a link goes down, all paths that use that link are broken and have to be repaired. If no application is

using these paths, then the effort gone in to repair may be considered wasted. This wasted effort can cause scarce bandwidth resources to be wasted and can cause further congestion at intermediate network points. Proactive protocols are scalable in the number of flows and the number of nodes but are not scalable in the frequency of topology change. Thus, this strategy is appropriate for a network with low mobility.

Certain proactive routing protocols are Destination-Sequenced Distance Vector (DSDV), Wireless Routing Protocol (WRP), Global State Routing (GSR) and Clusterhead Gateway Switch Routing (CGSR).

- **On-Demand Routing Protocols (Reactive)**

In contrast to proactive approach, in reactive or on demand protocols, a node initiates a route discovery throughout the network, only when it wants to send packets to its destination. For this purpose, a node initiates a *route discovery* process through the network. This process is completed once a route is determined or all possible permutations have been examined. Once a route has been established, it is maintained by a *route maintenance* process until either the destination becomes inaccessible along every path from the source or until the route is no longer desired. In reactive schemes, nodes maintain the routes to active destinations. A route search is needed for every unknown destination. Therefore, theoretically the communication overhead is reduced at expense of delay due to route research. Furthermore, the rapidly changing topology may break an active route and cause subsequent route searches [Joa-Ng and Lu, 1999]. Reactive protocols may not be optimal in terms of bandwidth utilization because of flooding of the route discovery request, but they remain scalable in the frequency of topology change. Such protocols are not scalable in the number of nodes, however, they can be made scalable if a hierarchical architecture is used. Further reactive protocols are not scalable in the number of flows. Thus, reactive strategies are suitable for networks with high mobility and relatively small number of flows.

Some reactive protocols are Cluster Based Routing Protocol (CBRP), Ad hoc On-Demand Distance Vector (AODV), Dynamic Source Routing (DSR), Temporally Ordered Routing Algorithm (TORA), Associativity-Based Routing (ABR), Signal Stability Routing (SSR) and Location Aided Routing (LAR).

- **Hybrid Protocols**

Finally in hybrid protocols, each node maintains both the topology information within its zone and the information regarding neighboring zones, that means, proactive behavior within a zone and reactive behavior among zones. Thus, a route to each destination within a zone is

established without delay, while a route discovery and a route maintenance procedure is required for destinations that are in other zones.

The zone routing protocol (ZRP), zone-based hierarchical link state (ZHLS) routing protocol and distributed dynamic routing algorithm (DDR) are three hybrid routing approaches. The hybrid protocols can provide a better trade-off between communication overhead and delay, but this trade-off is subjected to the size of a zone and the dynamics of a zone. Furthermore, hybrid approaches provide a compromise on scalability issue in relation to the frequency of end-to-end connection, the total number of nodes and the frequency of topology change. Thus, the hybrid approach is an appropriate candidate for routing in a large network.

3.14.2 Routing Architecture

- **Flat Architecture**

In flat architecture, all nodes carry the same responsibility. Flat architectures do not optimize bandwidth resource utilization in large networks because control messages have to be transmitted globally throughout the network, but they are appropriate for highly dynamic network topology. The scalability decreases when the number of nodes increases significantly.

- **Hierarchical Architecture**

On the contrary, in hierarchical architecture, aggregated nodes into clusters and clusters into super-clusters conceal the details of the network topology. Some nodes, such as clusterheads and gateway nodes have a higher computation communication load than other nodes. Hence, the mobility management becomes complex. The network reliability may also be affected due to single points of failure associated with the defined critical nodes. However, control messages may only have to be propagated within a cluster. Thus, the multilevel hierarchy reduces the storage requirement and the communication overhead of large wireless networks by providing a mechanism for localizing each node. In addition, hierarchical architectures are more suitable for low mobility case. Although flat architectures are more flexible and simpler than hierarchical one, hierarchical architectures provide more scalable approach.

- **Aggregate Architecture**

Finally, aggregate architecture aggregates a set of nodes into zones. Therefore, the network is partitioned into a set of zones. Each node belongs to two levels topology: low level (node level) topology and high level (zone level) topology. Also, each node is characterized by two ID numbers: node ID number and zone ID number. Normally, aggregate architecture is

related to the notion of zone. In aggregate architecture, we find both intra-zone and inter-zone architectures which in turn can either support flat or hierarchical architecture.

3.14.3 Routing Information

- **Global Position (GP) Based Protocols**

In global position (GP) based protocols, the network relies on another system which can provide the physical information of the current position of MNs. Such physical locations can be obtained by using the Global Position System (GPS). This involves increasing in energy consumption, cost of the network maintenance and hardware requirements. Generally, satellites are used to deliver this physical information. Any problem in one of the used satellites will surely affect the efficiency of the network and in some cases can easily make this latter blocked.

- **Global Position-Less (GPL) Based Protocols**

In global position-less (GPL) based protocols, the network is stand-alone in the sense that it operates independently of any infrastructure. However, there are some situations where the physical location remains useful such as emergency disaster relief after a hurricane or earthquake.

3.14.4 Routing Generation

- **First generation** of routing protocols in mobile ad hoc networks is ”*table-driven*” approaches which are mainly influenced by *Internet* routing protocols. They attempt to maintain consistent and up-to-date routing information from each node to every other node in the network.
- **Second generation** of routing protocols is ”*on-demand*” approaches that are designed in order to decrease high communication overhead in table-driven approach due to maintaining up-to-date routing information.

While on-demand routing protocols decrease communication overhead, there are doubts about its scalability and delay. On-demand routing protocols are different in the way they construct and maintain a route to the destination and the metrics they use to differentiate the discovered routes, as well as the mechanism to avoid loop formation. Path finding differs from reactive to proactive approach in the sense that reactive approach applies an explicit route request that is followed by an explicit route reply while proactive approach uses implicit route reply.

• **Third generation** of routing protocols called *hybrid approach* which is introduced to provide a better compromise between communication overhead and delay as well as better scalability.

Finally, we have compiled a list of most routing protocols we have found. The detailed list can be found in Appendix B.

Bibliography

[Bouazizi and Günes, 2002] I. Bouazizi, and M. Günes. A Framework for Transmitting Video over Mobile Multi-hop Ad-Hoc Networks. In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Band XV, Orlando, USA, 2002.

[Bouazizi and Günes, 2003] I. Bouazizi and M. Günes. A Video Streaming Framework for Transporting MPEG-4 Video over Mobile Ad-Hoc Networks. In *Proceedings of the 2nd Mediterranean Workshop on Ad-Hoc Networks (Med-Hoc-Net'2003)*, pages 211-218, Mahdia, Tunisia, 2003.

[Broch et al., 1998] J. Broch, D. A. Maltz, D. B. Johnsin, Y. C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wirless Ad Hoc Network Routing Protocols. In *Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MOBICOM'98, 1998.

[Chlamtac et al., 2003] I. Chlamtac, M. Conti, and J.-N. Jennifer. Mobile Ad Hoc Networking: Imperatives and Challenges. *In Elsevier proceeding for ad hoc networks*, vol.1, pages13-64, 2003.

[David, 2002] L. David. Ad Hoc Protocol Evaluation And Experiences Of Real World Ad Hoc Networking. *Technical report to Department of Information Technology*, Uppsala University, Sweden, 2002.

[Di Caro and Dorigo, 1998] G. Di Caro, M. Dorigo. AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research (JAIR)*, vol. 9, pages 317-365, 1998.

[Frodigh et al., 2000] M. Frodigh, P. Jonasson, and P. Larsson. Wireless Ad Hoc Networking – The Art of Networking without a Network. *Ericsson Review*, vol. 4, 2000.

[Günes, 2004] M. Günes. *Routing und Adressierung in Mobilen Multi-Hop Ad-Hoc-Netzen*, Ph.D. thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 2004.

[Haartsen, 1998] J. Haartsen. Bluetooth-The Universal Radio Interface for Ad Hoc Wireless Connectivity. *Ericsson Review (3)*, 1998.

[Joa-Ng and Lu, 1999] M. Joa-Ng and I. T. Lu. A Peer-To-Peer Zone-Based Two-Level Link State Routing for Mobile Ad Hoc Networks. *IEEE on Selected Areas in Communications*, vol. 17, no. 8, pages 1415–1425, 1999.

[Kardash, 2000] J. Kardash. Bluetooth Architecture Overview. *Intel Technology Journal Q2*, Mobile Computing Group, Intel Corporation, 2000.

[Lee et al., 1999] S-J. Lee, M. Gerla, and C. K. Toh. A Simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks. *IEEE Network*, vol. 13, no. 4, pages 48–54, 1999.

[Murphy et al., 1998] A. L. Murphy, G. C. Roman, G. Varghese. An Exercise in Formal Reasoning about Mobile Communications. *Proceedings of the Ninth International Workshop on Software Specifications and Design*, IEEE Computer Society Technical Council on Software Engineering, Japan, pages 25-33, 1998.

[Nikaein et al., 2000] N. Nikaein, H. Labiod, and C. Bonnet. DDR-Distributed Dynamic Routing Algorithm for Mobile Ad hoc Networks. *First Annual Workshop on Mobile and Ad Hoc Networking and Computing (MobiHOC)*, 2000.

[Nikaein et al., 2001] N. Nikaein, S. Wu, C. Bonnet, and H. Labiod. Designing Routing Protocol For Mobile Ad Hoc Networks, 2001.

[Perkins, 2001] C. E. Perkins. *Ad Hoc Networking*. Addison-Wesley, ISBN 0-201-30976-9, 2001.

[Rover and Toh, 1999] E. M. Rover and C. K. Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. *IEEE Personal Communications*, vol. 6, no. 2, pages 46–55, 1999.

[Tao, 2004] L. Tao. Mobile Ad-hoc Network Routing Protocols: Methodologies and Applications. *Ph.D. thesis in Computer Engineering, Virginia Polytechnic Institute and State University*, 2004.

[Toh, 2002] C. K. Toh. *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall PTR, ISBN: 0130078174, 2002.

Chapter
4

ANT COLONY OPTIMIZATION

4.1 Introduction

During recent years, new algorithms have been introduced to tackle the combinatorial optimization problems. Some examples of these algorithms are G*enetic Algorithms* [Hertz, 1991], *Ant Systems* [Dorigo et al., 1996] and *Neural Networks* [GoldBerg, 1991]. Those algorithms are inspired by behavior or processes present in nature. Such mechanisms have been proven to be extremely effective and evolutionary. These algorithms demonstrate adaptive, robust and effective behavior as nature does; where adaptive means that it improves its goal-achieving competence over time, robust means it is flexible and never completely breaks down while effective means it is eventually finding a satisfactory solution.

The observation and study of ants societies have long since attracted the attention of the professional entomologist, but in recent years, the ant model of organization and interaction has also captured the interest of the computer scientist and engineers. Ant societies, among other societies, have many features like: autonomy of individuals, fully distributed control, collective and cooperative strategies, emergence of complex behaviors with respect to the single ant and self-organization. The simultaneous presence of these unique chracteristics has made ant societies an attractive and inspiring model for building new algorithms and new multi-agent systems.

Since the year 1990, ant societies have provided the force for a growing field of scientific work, mostly in the field of robotics, operations research and telecommunications. Researchers from all over the world, processing different scientific backgrounds, have made significant progress concerning both implementation and theoretical aspects, within this novel research framework. Their contributions have given the field a solid basis and have shown

how the "ant way", when carefully engineered, can result in successful applications to many real-world problems.

A particular successful research in ant algorithms, known as *Ant Colony Optimization* (*ACO*), is dedicated to their application to discrete optimization problems. Ant colony optimization has been applied successfully to a large number of difficult combinatorial problems such as the traveling salesman problem, the quadratic assignment problem and scheduling problems, as well as routing in telecommunication networks.

Ant colony optimization algorithms have all been inspired by a specific foraging behavior of ant colonies which are able to find if not the shortest, at least a very good path connecting the colony's nest with a source of food.

Ant algorithm was described in the doctoral thesis of Marco Dorigo, at the beginning of the 1990s. This first application stimulated the interest of other researchers around the world to design algorithms for several important optimization and network problems getting inspiration from the same ant foraging behavior, and from ant system in particular. Recently, in 1998, the ant colony optimization metaheuristic was defined, providing a common framework for describing this important class of ant algorithms.

In the next section, we will introduce in more details a survey of the nowadays most important metaheuristics because ant colony optimization is considered as one of these new metaheuristics. We outline the different components and concepts that are used in the different metaheuristics in order to analyze their similarities and differences.

4.2 Metaheuristics

In the last 20 years, a new kind of approximate algorithm has emerged, which basically tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. This kind of algorithms is nowadays commonly called *metaheuristics*. The term *metaheuristic*, first introduced in [Glover, 1986], is derived from the composition of two Greek words. *Heuristic* is derived from the verb *heuriskein* which means "to find", while the suffix *meta* means "beyond, in an upper level". Before this term was widely adopted, metaheuristics were often called *modern heuristics* [Reeves, 1993].

Under the umbrella of meta-heuristics, there are variety of heuristic procedures such as *Ant Colony Optimization* (ACO), *Evolutionary Computation* (EC) including *Genetic Algorithms* (GA), *Iterated Local Search* (ILS), *Simulated Annealing* (SA) and *Tabu Search* (TS).

Metaheuristics are a general class of heuristics for solving hard problems. They are sometimes considered as intelligent heuristic search, which can avoid the local optimality and

incorporate various strategies inspired from natural behaviors of species, mathematical reasoning, physical science, nervous systems and statistical mechanics.

Up to now, there is no commonly accepted definition for the term metaheuristic. It is just in the last few years that some researchers in the field tried to propose a definition. In the following, we quote some of them:

"*A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions*." [Osman and Laporte, 1996].

"*A metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method*." [Voß et al. 1999].

"*Metaheuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance. The main goal is to avoid the disadvantages of iterative improvement and, in particular, multiple descent by allowing the local search to escape from local optima. This is achieved by either allowing worsening moves or generating new starting solutions for the local search in a more "intelligent" way than just providing random initial solutions. Many of the methods can be interpreted as introducing a bias such that high quality solutions are produced quickly. This bias can be of various forms and can be cast as descent bias (based on the objective function), memory bias (based on previously made decisions) or experience bias (based on prior performance). Many of the metaheuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in metaheuristic algorithms randomness is not used blindly but in an intelligent, biased form*." [Stützle, 1999].

"*A metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem*." [Metaheuristics Network Website, 2000].

C. Blum and A. Roli summarized the main fundamental properties which characterize metaheuristics [Blum and Roli, 2003]:

- Metaheuristics are strategies that "guide" the search process.

- The goal is to efficiently explore the search space in order to find optimal or nearly optimal solution for a given problem.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
- The basic concepts of metaheuristics permit an abstract level description.
- Metaheuristics are not problem-specific.
- Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.
- Todays, more advanced metaheuristics use search experience to guide the search.

4.2.1 Classification of Metaheuristics

There are different ways to classify and describe metaheuristic algorithms. Depending on the characteristics selected to differentiate among them, several classifications are possible, each of them is the result of a specific viewpoint. The following paragraphs summarize the most important ways of classifying metaheuristics.

4.2.1.1 Nature inspired vs. non-nature inspired

One of the most intuitive way of classifying metaheuristics is based on the origins of the algorithm. There are nature inspired algorithms, like *Genetic Algorithms* and *Ant Algorithms* and non nature-inspired ones such as *Tabu Search* and *Iterated Local Search*. This classification is not very meaningful for the following two reasons. First, many recent hybrid algorithms fit both classes at the same time. Second, it is sometimes difficult to clearly attribute an algorithm to one of the two classes.

4.2.1.2 Trajectory methods vs. discontinuous methods

An important distinction between different metaheuristics is whether they follow one single search *trajectory* corresponding to a closed walk on the neighborhood graph or whether larger jumps in the neighborhood graph are allowed.

Simulated Annealing and *Tabu Search* are typical examples of trajectory methods. These methods usually allow moving to worse solutions to be able to escape from local minima. Also, *Local Search Algorithms*, which perform more complex transitions composed of

simpler moves, may be interpreted as trajectory methods. In ant colony optimization, iterated local search and genetic algorithms starting points for a subsequent local search are generated. This is done by constructing solutions with ants, modifying previously visited locally optimal solutions and applying genetic operators, respectively. The generation of starting solutions corresponds to jumps in the search space; these algorithms, in general, follow a discontinuous walk with respect to the neighborhood graph used in the local search.

4.2.1.3 Population-based vs. single point search

Another characteristic that can be used for the classification of metaheuristics is the number of solutions used at the same time. In other words, does the algorithm work on a population at the same time or on a single solution (point)?

Local search-based metaheuristics, like *Tabu Search*, *Iterated Local Search* and *Variable Neighborhood Search*, all share the property of describing a trajectory in the search space during the search process. Population-based metaheuristics, on the contrary, perform search processes, which describe the evolution of a set of points in the search space. Examples of population-based metaheuristics are *Genetic Algorithms* and *Ant Algorithms*. Note that population-based methods are typically discontinuous. In ant algorithm, there is a population of ants in which each ant is trying to find a solution and then, communicates with the other ants hoping that it will help them finding even better solutions guided by the pheromone trails and a heuristic function. In genetic algorithms, the population is modified using the genetic operators. Using a population-based algorithm provides a convenient way for the exploration of the search space. Yet, the final performance depends strongly on the way the population is manipulated.

4.2.1.4 Dynamic vs. static objective function

Metaheuristics can also be classified according to the way they make use of the objective function. While some algorithms keep the objective function given in the problem representation “as it is”, some others, like *Guided Local Search* (GLS) and *Tabu Search* (TS), modify it during the search. The idea behind this approach is to escape from local minima by modifying the search landscape. Accordingly during the search, the objective function is altered by trying to incorporate information collected during the search process. Yet, all the other algorithms use a static objective function.

4.2.1.5 One vs. various neighborhood structures

Most metaheuristic algorithms work on one single neighborhood structure which defines the type of allowed moves. This is the case for *Simulated Annealing* and *Tabu Search*. On the other side, the solution construction process in ant colony optimization is not based on a specific neighborhood structure. Nevertheless, one could interpret the construction process used in ACO as a kind of local search, but this interpretation does not reflect the basic algorithmic idea of these approaches.

4.2.1.6 Memory usage vs. memory-less methods

A very important feature used to classify metaheuristics is the presence of the search history (search experience), that means whether they use memory or not. Memory is explicitly used in *Tabu Search*. Short term memory is used to forbid revisiting recently found solutions and to avoid cycling, while long term memory is used for diversification and intensification features. Ant colony optimization is an indirect kind of *adaptive memory* of previously visited solutions which were kept via the pheromone trail matrix, that is used to influence the construction of new solutions. Also, the population of the genetic algorithm could be interpreted as a kind of memory of the recent search experience. On the contrary, *Simulated Annealing* do not use memory functions to influence the future search direction and therefore are memoryless algorithms. The use of memory is nowadays recognized as one of the fundamental elements of a powerful metaheuristic.

4.2.1.7 Iterative vs. constructive

Iterative metaheuristics start with a complete and feasible solution and make systematic changes to this solution in order to improve its quality. ILS is a typical example of an iterative metaheuristic, as it steps through the components of the given solution and tries modifications within some local neighborhood of each component.

On the other hand side, constructive metaheuristics construct a solution by adding components to a partial solution until a complete and feasible solution is found. *Greedy Heuristics* (GH) are constructive in the sense that they linearly build a solution by repeatedly adding a component, by some local notion of viability (e.g. shortest edge connected to a neighbor-vertex of graph).

4.2.1.8 Single-run vs. repetitive

In single-run metaheuristics, an algorithm only continues to run until a solution is generated.

That means in single-run metaheuristics, a generated solution is not used to generate new better solutions. Examples of single-run metaheuristics could be LS (in which a solution is presented when a local optimum is reached) or GH (that stops when the construction process is finished).

Repetitive metaheuristic algorithms let the user have some control on how much computation is done, and the quality of the solution improves as the computation time increases. Experience gained while generating solutions is used as a basis for generating new solutions. GA is an example of a repetitive heuristic. Typically, this type of algorithms uses *stochastic state transitions* as a way of avoiding local optimums.

Table 4-1 summarizes different kinds of metaheuristics discussed in this section giving an indication for their particular characteristics in standard use. √ means that the feature is present, ∃ that this feature is partially present and ¬ that the feature does not appear.

Table 4-1. Summary of some metaheuristics features

Feature	SA	TS	GA	ACO	ILS
Trajectory	√	√	¬	¬	¬
Population	¬	¬	√	√	¬
Memory	¬	√	∃	√	∃
Multiple neighborhoods	¬	¬	∃	¬	√
Dynamic *f(x)*	¬	∃	¬	¬	¬
Nature-inspired	√	¬	√	√	¬

From the previous properties of metaheuristics we can say that, ACO metaheuristic can be classified as nature-inspired, population-based, static objective function, memory usage, constructive and repetitive.

4.3 Swarm Intelligence and Ant Algorithms

It is worth mentioning that ant algorithms belong to a larger field of algorithms called *Swarm Intelligence* (SI) algorithms. Swarm intelligence is an area of research that over the last decade has been experienced. It is inspired by the seemingly intelligent behavior of swarms of primitive animals and has proven to be a promising field of research in many different areas. SI is defined as '*the emergent collective intelligence of groups of simple agents'* [Bonabeau et al., 1999]. This field is inspired by the surprising capabilities of collective social insects.

SI is used to refer to systems whose design is inspired by models of social insect behavior. Key characteristics of these models are:

- Large numbers of simple agents.
- Agents may communicate with each other directly.
- Agents may communicate indirectly by affecting their environment, a process known as *stigmergy*.
- Intelligence contained in the networks and communications between agents.
- Local behavior of agents causes some *emergent global behavior*.

A swarm of ants in the search for food shows the remarkable capability of finding shortest paths between a food source and the nest. The amazing thing is that ants cooperate on finding food, by trail-laying and trail-following behavior when foraging. The individual ants deposit a chemical substance called *pheromone* as they move from their nest to a food source, and foragers follow such pheromone trails. The pheromone trails used as a simple indirect form of communication. The process of emerging global information from local actions, through small independent agents not directly communicating with each other, is called *Stigmergy* [Dorigo et al., 1999] that is developed out of the study of social insects. Stigmergy was introduced by French entomologist Pierre-Paul Grassé [Grassé, 1959]. Grassé recognised that the insects have biological characteristics that drive inter-individual stimuli. These stimuli generate highly stereotyped behavior that results in making colonies of individuals capable of performing very complex tasks rather than only simple actions. Ant colony algorithm uses the stigmergy characteristic of pheromone trail laying used by ants when foraging. This property is used in algorithms and heuristics to solve various NP-hard problems.

Some research have been focused on the use of swarm intelligence type systems for routing within communications networks, such as AntNet and Ant-based Control routing protocols. This chapter focuses deeply on ant algorithms, we will give a historical presentation and some of the properties that are central in the context of ant algorithms and in their inspiration.

4.4 Historical Development of ACO Algorithms

The first *Ant Colony Optimization* (ACO) algorithm proposed was *Ant System* (AS) algorithm [Dorigo et al., 1991]. AS was applied to some rather small instances of the *Traveling Salesman Problem* (TSP) with up to 75 cities. It was able to reach the performance of other general-purpose heuristics like evolutionary computation [Dorigo, 1992] and [Dorigo et al., 1996]. In spite of these initial encouraging results, AS could not prove to be competitive with

state-of-the-art algorithms specifically designed for the TSP when attacking large instances. Therefore, a substantial amount of recent research has been focused on ACO algorithms which show better performance than AS when applied, for example, to the TSP.

Initially, three different versions of AS were proposed [Dorigo et al., 1991a; Colorni et al., 1992a; Dorigo, 1992]. These were called *ant-cycle*, *ant-density* and *ant-quantity*. In *ant-density* and *ant-quantity* versions, the ants updated the pheromone directly after a move from one city to an adjacent city, but in *ant-cycle* version the pheromone update was only done after all the ants had constructed the tours and the amount of pheromone deposited by each ant was set to be a function of the tour quality. Because *ant-cycle* performed better than the other two variants, it was later simply called Ant System (AS), while the other two algorithms were no longer studied. The two main phases of the AS algorithm are the ants solution construction and the pheromone update.

The major merit of AS was that its computational results were promising but not competitive with other more established approaches. On the other hand, this first version exhibited two major disadvantages, the excessive computational time and a clearly suboptimal convergence. In order to remedy these two weaknesses, two approaches had been proposed. The first improvement on the initial AS, called the *elitist strategy* for Ant System (EAS) denoted by AS_{elite}, was introduced in [Dorigo et al., 1991a, 1996]. Another improvement over AS is the *rank-based* version of AS denoted by AS_{rank}, proposed in [Bullnheimer et al., 1997].

AS_{elite} improved the convergence by emphasizing the best solution found. The modification of the basic version aims to select appropriate number of elitist ants that allows AS to find better paths and to find them earlier in the run. If too many elitist ants are used, the search will concentrates early around suboptimal solutions leading to a premature stagnation of the search. By choosing the best solution that is either found during one cycle of the algorithm or from its start, two different subversions have been developed, the best-of-cycle and the global best.

AS_{rank} introduced a new phase after each cycle of the algorithm. The solutions of all the ants have been evaluated, the ants ranked according to the quality of their solutions. Only a fixed percentage p of the best ants is allowed to update the pheromone levels. This reduces the amount of necessary computational time in the phase when the pheromone is updated and accelerates the convergence of the algorithm.

A more radical approach to the modification of the algorithm was *MAX–MIN* Ant Systems (*MM*AS), introduced in [Stützle and Hoos, 1997, 2000; Stützle, 1999]. In *MM*AS, only the best ant is allowed to update the trails. In order to compensate for the emerging premature

convergence, a minimum amount of pheromone τ_{min} is fixed. Once the pheromone deposited on a basic solution drops underneath this lower bound, it is reset to the minimum value. In order to make sure that a greater part of the space is searched, the algorithm initializes the pheromone levels to a preset maximum value τ_{max} which constitutes an upper bound for the trail intensity.

Another ant algorithm was derived from a well-known reinforcement learning algorithm called Q-Learning which was introduced in [Watkins,1989]. The resulting algorithm, Ant–Q, was proposed in [Gambardella and Dorigo, 1995]. The new ideas consist of partially evaporating the pheromone of an elementary assignment immediately after its construction in order to speed up the search of the entire search space. It was the observation that two ants tend to build similar solutions once they start close to each other that led to this idea.

Evaporating the pheromone, and thus creating a slight demotivation to follow the same path for the next ant, leads to a broader search. A second feature is the modified rule for choosing the next elementary assignment. With a certain probability, the local best possibility is chosen without taking the others into consideration. If this is not the case, the basic rule is used using the local and global quality of the solution. This results in a strategy which closely resembles the elitist approach. The third major change was the idea to use the prospective quality of the following elementary assignment to calculate the amount of pheromone deposited on a chosen partial solution. This concept was quickly abandoned for reasons of inefficiency. Concerning the choice of the ants which are allowed to deposit pheromone, Ant–Q follows the same strategy as *MAX–MIN* Ant Systems: Only the best ant has this privilege. The so far final and probably best performing algorithm was derived from Ant–Q by replacing the above mentioned concept with an implicitly guaranteed minimum levels of pheromone. The resulting algorithm was presented in [Dorigo et al., 1999] and named *Ant Colony System* (ACS). There exists an interesting relationship between *MM*AS and ACS, they both use pheromone trail limits. Figure 4-1 represents an overview of the above different algorithms.

Finally, the ACO metaheuristic was defined with the goal of providing a common characterization of a new class of algorithms and a reference framework for the design of new instances of ACO algorithms.

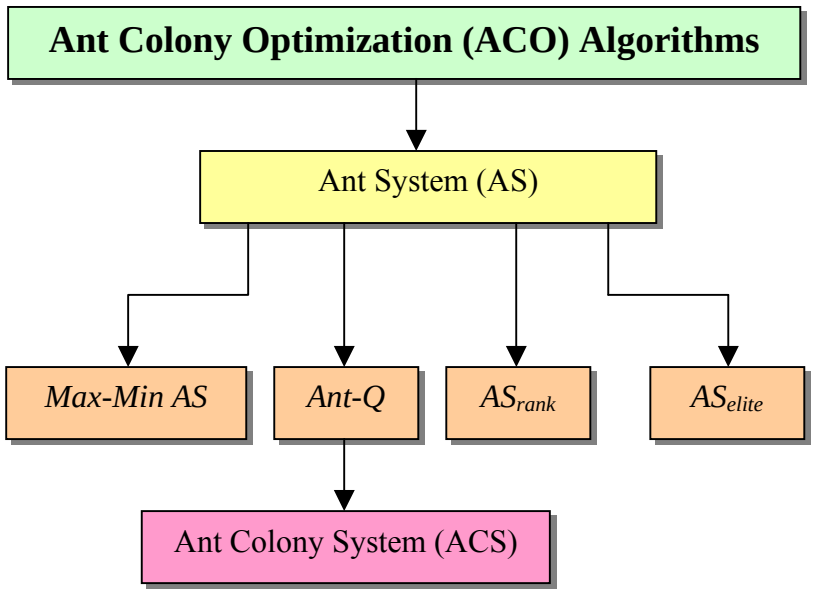


Figure 4-1. Overview of the different ACO algorithms

4.5 Similarities and Differences between Real and Artificial Ants

Most of the ideas of ACO stem from real ants. In particular, the use of a colony of cooperating individuals, an artificial pheromone trail for local stigmergetic communication, a sequence of local moves to find shortest paths and a stochastic decision policy using local information.

Researchers have used the most ideas from real ants behavior in order to build *Ant System* AS. There exist some differences and similarities between real and artificial ants which could be stated as follows [Dorigo et al., 1999]:

A. Similarities

- **Colony of cooperating individuals**. Both real ant colonies and ant algorithms are composed of a population, or colony of independent individual agents. They globally cooperate in order to find a good solution to the task under consideration. Although the complexity of each artificial ant is such that it can build a feasible solution (as a real ant can find somehow a path between the nest and the food), high quality solutions are the result of the cooperation among the individuals of the whole colony.
- **Pheromone trail and stigmergy**. Like real ants, artificial ants change some aspects of their environment while walking. Real ants deposit a chemical substance called pheromone on the visited state. Artificial ants will change some numerical information of the problem state, locally stored, when that state is visited. Based on analogy, these information and

changes could be called an artificial pheromone trail. Ant system algorithms assume that a local pheromone trail is the single way of communication among artificial ants. AS algorithms include artificial pheromone evaporation in form of reduction in the artificial pheromone trail over time as in nature. Pheromone evaporation in nature and in AS algorithms is important because it will allow ant colony to slowly forget the history and direct the searching process in new directions. Artificial pheromone evaporation could be helpful to move the searching process toward new regions and to avoid stacking in local extremes.

- **Local moves and the shortest path searching**. Despite real ants are walking through adjacent states and artificial ants are jumping from one to another adjacent state of the considered problem, both walking and jumping have the same purpose, which is finding the shortest path between the origin and the destination.
- **Transition policy**. Both real ants and artificial ones will build solutions by applying decision making procedures to move through adjacent states. Decision making procedures could be based on some probabilistic rules introduced in [Colorni et al., 1991, 1992] and [Dorigo et al., 1996] or probabilities could be calculated based on approximate reasoning rules. In both cases, the transition policy will use local information that should be local in the space and time sense. The transition policy is a function of local state information represented by problem specifications (this could be equivalent to the terrain's structure that surrounds the real ants) and the local modification of the environment (existing pheromone trails) introduced by ants that have visited the same location.
- **Deposited amount of pheromone**. Amount of pheromone that an artificial ant will deposit is mostly a function of the quality of the discovered solution. In nature, some ants behave in a similar way, the deposited amount of pheromone is highly dependent on the quality of the discovered food source.

B. Differences

Main differences could be stated as follows:

- Artificial ants live in a discrete world. All their moves are jumps from one discrete state to another adjacent one.
- Artificial ants have memory, they could remember states that have been visited already (tabu lists in the model).

- Pheromone deposit methodology is significantly different between real and artificial ants. Timing in pheromone laying is problem dependent and often does not have similarities with the real ants pheromone deposit methodology.
- To improve overall performance, AS algorithms could be enriched with some additional capabilities that cannot be found in real ant colonies. Most AS contains some local optimization techniques to improve solutions developed by ants.

4.6 Ant Algorithm Characteristics

An ant algorithm presents the following characteristics:

- **Natural algorithm** since it is based on the behavior of real ants in establishing paths from their colony to source of food and back.
- **Parallel and distributed** since it concerns a population of agents moving simultaneously, independently and without a supervisor.
- **Cooperative** since each agent chooses a path on the basis of the information, pheromone trails laid by the other agents, which have previously selected the same path. This cooperative behavior is also autocatalytic, i.e., it provides a positive feedback, since the probability of choosing a path increases with the number of agents that previously chose that path.
- **Versatile** that it can be applied to similar versions of the same problem; for example, there is a straightforward extension from the traveling salesman problem (TSP) to the asymmetric traveling salesman problem (ATSP).
- **Robust** that it can be applied with minimal changes to other combinatorial optimization problems such as quadratic assignment problem (QAP) and the job-shop scheduling problem (JSP).

4.7 Ant Algorithm Similarities with Some Other Optimization Approaches

ACO algorithms show similarities with some optimization, learning and simulation approaches like *heuristic graph search*, *Monte Carlo simulation*, *neural networks* and *evolutionary computation* [Gambardella, 2004]. These similarities are briefly discussed in the following:

4.7.1 Heuristic Graph Search

In ACO algorithms, each ant performs a heuristic graph search in the space of the components of a solution. Ants take biased probabilistic decisions to choose the next component to move to, where the bias is given by a heuristic evaluation function which favors components that are perceived as more promising. It is interesting to note that this is different from what happens, for example, in *Simulated Annealing*, where:

- An acceptance criteria is defined and only those randomly generated moves which satisfy the criteria are executed,
- The search is usually performed in the space of the solutions.

4.7.2 Monte Carlo Simulation

Monte Carlo (MC) simulation [Rubistein, 1981] is a stochastic simulation technique (meaning they are based on the use of random numbers and probability statistics to investigate problems) used to solve problems such as economics, nuclear physics and regulating the flow of traffic. This technique performs repeated sampling experiments on the model of the system under consideration, by making use of a stochastic component in the state sampling and/or transition rules. Experiment results are used to update some statistical knowledge about the problem, as well as to estimate the variables the researcher is interested in. In turn, this knowledge can be also iteratively used to reduce the variance in the estimation of the desired variables, directing the simulation process toward the most interesting state space regions. Analogously, in ACO algorithms the ants sample the problem's solution space by repeatedly applying a stochastic decision policy until a feasible solution of the considered problem is built. The sampling is realized concurrently by a collection of differently instantiated replicas of the same ant type. Each ant experiment allows ants to adaptively modify the local statistical knowledge on the problem structure (i.e., the pheromone trails). The recursive transmission of such knowledge by means of *stigmergy* determines a reduction in the variance of the whole search process. ACO algorithms can be interpreted as parallel replicated MC systems.

4.7.3 Neural Networks

Ant colonies, being composed of numerous concurrently and locally interacting units, can be seen as *connectionist systems* [Feldman and Ballard, 1982]. The most famous example of connectionist systems is neural networks. Figure 4-2 shows the structure of a single neuron, which represents the basic part of the neural networks. From a structural point of view, the

parallel between the ACO and neural network is obtained by putting each state *i* visited by ants in correspondence with a neuron *i*, and the problem specific neighborhood structure of state *i* in correspondence with the set of synaptic-like links exiting neuron *i*. The ants themselves can be seen as input signals concurrently propagating through the neural network and modifying the strength of the synaptic-like inter-neuron connections. Signals (ants) are locally propagated by means of a stochastic transfer function and the more a synapse is used, the more the connection between its two end-neurons is reinforced. The ACO-synaptic learning rule can be interpreted as a posteriori rule: signals related to good examples, that means ants which discovered a good quality solution, reinforce the synaptic connections they traverse more than signals related to poor examples. It is interesting to note that the ACO-neural network algorithm does not correspond to any existing neural network model. The ACO-neural network is also reminiscent of networks solving reinforcement learning problems. In reinforcement learning, the only feedback available to the learner is a numeric signal (the reinforcement) that scores the result of actions. This is also the case in the ACO metaheuristic: The signals (ants) fed into the network can be seen as input examples with an associated approximate score measure. The strength of pheromone updates and the level of stochasticity in signal propagation play the role of a learning rate, controlling the balance between exploration and exploitation.

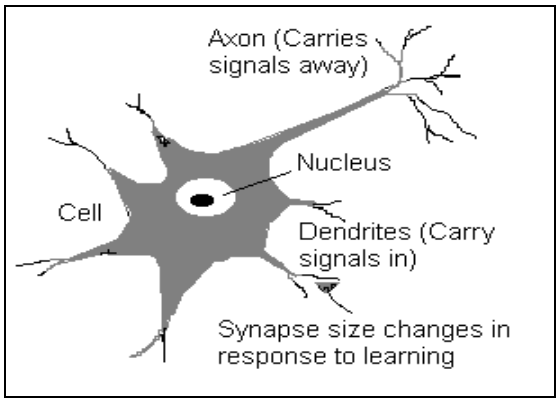


Figure 4-2. The structure of neuron
source of the diagram: NIBS Pte Ltd.

4.7.4 Evolutionary Computation

There are some general similarities between the ACO meta-heuristic and evolutionary computation (EC). Both are bioinspired techniques, that means, both of them mimic in some way a natural process to achieve a good solution of the problem being tackled. Both approaches use a population of individuals to represent problem solutions. Nevertheless, in EC the solutions are in fact the individuals (chromosomes), while in the ACO metaheuristic

the individuals are ants, which construct and remember the solutions found so far. Both approaches use the knowledge collected by the population at the algorithm run-time to stochastically generate a new population of individuals.

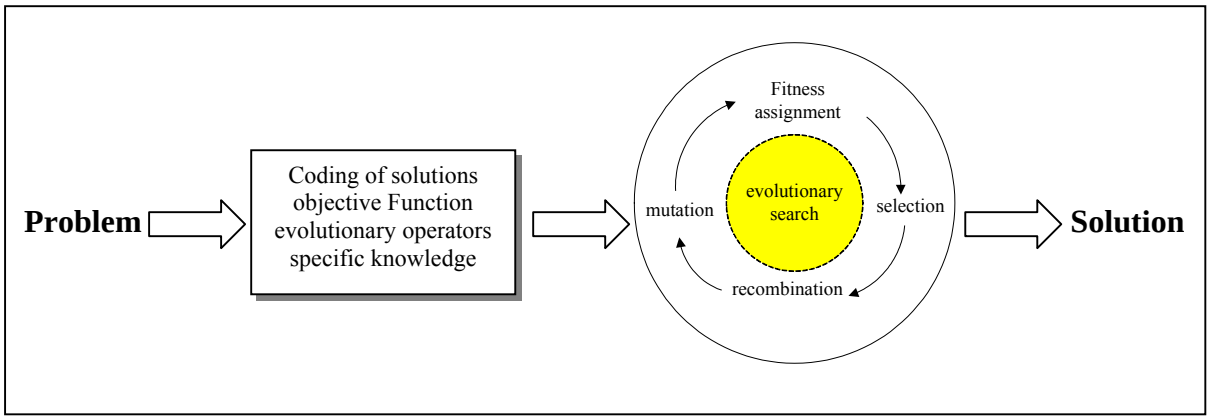


Figure 4-3. Problem solution using evolutionary algorithms

A main difference is that in EC algorithms, all the knowledge about the problem is contained in the current population, while in ACO a memory of past performance is maintained under the form of pheromone trails. Another difference is that EC can use different mutation and crossover operators while a particular ACO algorithm uses a single construction rule to generate new solutions to the problem. Figure 4-3 shows how a problem could be solved by using evolutionary algorithms.

4.7.5 Stochastic Learning Automata

This is one of the oldest approaches to machine learning [Narendra and Thathachar, 1989]. An *automaton* is a machine or control mechanism designed to automatically follow a set of possible predetermined sequence of operations (actions) or respond to encoded instructions. The term *stochastic* emphasizes the adaptive nature of the automaton. This adaptation is the result of the *learning* process.

"*The concept of learning automaton grew out of a fusion of the work of psychologists in modeling observed behavior, the efforts of statisticians to model the choice of experiments based on past observations, the attempts of operation researchers to implement optimal strategies in the context of the two-armed bandit problem, and the endeavors of system theorists to make rational decisions in random environments*".

Automata are connected in a feedback configuration with the environment. A set of penalty signals from the environment to the actions is defined as indicted in Figure 4-4.

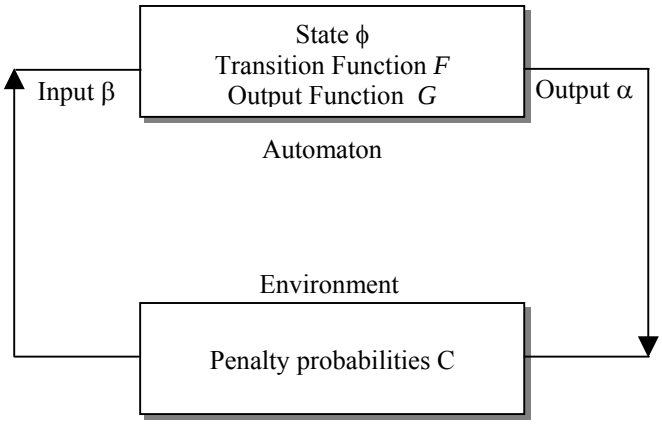


Figure 4-4. The automaton and the environment

The similarity of stochastic learning automata and ACO approaches can be made clear as follows: The set of pheromone trails available on each link is seen as a set of concurrent stochastic learning automata. Ants play the role of the environment signals, while the pheromone update rule is the automaton learning rule. The main difference lies in the fact that in ACO the "environment signals" (i.e., the ants) are stochastically biased by means of their probabilistic transition rule, to direct the learning process toward the most interesting regions of the search space. That means, the whole environment plays a key active role to learn good state-action pairs.

4.8 Collective Behavior of Social Insects

In order to better understand the mechanisms in artificial social insects, one must understand the collective behavior in real social insects. In the following, we will give a short description of the primary mechanisms that are determining the collective behavior of social insects during food foraging, since this is the primary inspiration for the ACO metaheuristic. This will facilitate a later discussion regarding the creation of artificial insects, and also regarding algorithmic problems in a virtual domain. Ant algorithms rely mainly on two concepts adopted from biology, *self-organization* and *positive feedback*.

4.8.1 Self-Organization

The term *self-organization* (SO) is first defined by Farley and Clark of Lincoln Laboratory in [Yovits et al., 1962] as follows: "*A self-organizing system is a system that changes its basic structure as a function of its experience and environment*".

This definition clearly relates to today's hot topics of adaptive control, neural networks, genetic algorithms and ant colony algorithms. One encounters self-organization in many fields. Biology (insect societies, ecosystems), chemistry (thermodynamics), computer science (decision algorithms, neural networks and fuzzy logic), geology (tectonic movements), sociology (communication and migration) and economy (socio-spatial systems) are some areas where self-organizing systems are encountered often.

The term can be used to describe a social colony of insects where every single insect functions autonomously. There is no central control or "supervisor" insect, which coordinates the labor of the insects, but the collective organization of the colony remains well ordered. The conjecture is that complex collective behavior can emerge from the interaction of simple entities.

A social insect colony is described as a "*…decentralized problem-solving system, comprised of many relatively simple interacting entities*" [Bonabeau et al., 1999].

The way the system is made up by many small and simple entities enables it to both adapt to changes in the environment and keep on functioning even though one (or more) of the entities stops functioning (dies). Self-organization in the context of social insects depends on the following mechanisms:

4.8.2 Positive Feedback (Amplification)

Positive feedback refers to a mechanism that allows colonies of ants to converge toward finding short paths from the anthill to a food source. It is created by stigmergic communication between the ants. Ants start moving randomly, at certain point an ant could find a food source and start to move back to the anthill (some ant species seems to always know the path back to the anthill). Other ants – still moving around randomly – smell the trail and find the food source and also return to the anthill, following almost the same path. This increase in pheromone value recruits even more ants that start to follow the path, thus the positive feedback mechanism ensures that almost all ants follow the path, and thus the ants have found a path to the anthill.

Examples of positive feedback include recruitment and reinforcement. For instance, recruitment to a food source is a positive feedback that relies on trail laying and trail following in some ant species. In the context of foraging, the problem is to find the shortest path from the anthill to the food source. In certain species, ants deposit pheromone trails while moving between a discovered food source and the anthill. Positive feedback is achieved because ants can smell deposited pheromone and have a natural tendency to follow the trail

laid out by other ants. These ants again deposit pheromone on the same path, reinforcing the already existing pheromone trail, which again enhances the probability of other ants to discover the trail, thus yielding an ever-higher positive feedback as shown in Figure 4-5.

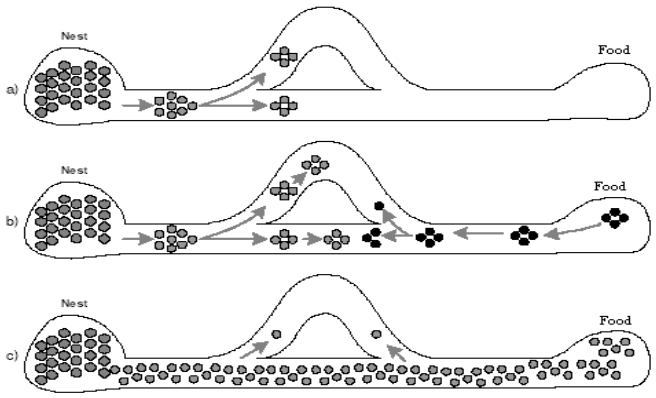


Figure 4-5. Shortest path by ant system

The subtleties of this process will be explained by example in *Double bridge experiment* which we will discuss it in later section, but it will be set in the context of artificial ants.

4.8.3 Negative Feedback

Negative feedback is the mechanism that counteracts the effect of positive feedback. It counterbalances positive feedback and helps to stabilize the collective pattern. It may take the form of saturation, exhaustion or competition. An often-used example of negative feedback is pheromone evaporation. Ants must maintain a trail of pheromone by continuously depositing more pheromone, otherwise pheromone will evaporate and consequently the created trail will be 'forgotten'.

This is a good property, since it allows the ants to leave bad solutions. If a path leads to a food source that is exhausted, the ants will not return to the anthill. Instead, they continue their exploration, thus letting the created trail evaporate in search of other food sources.

4.8.4 Amplification of Fluctuations (Randomness)

SO strongly depends on randomness (random walks and errors in trail-following and so on). Randomness is referred to as a crucial factor for the discovery of new solutions "*Not only do structures emerge despite randomness, but randomness is often crucial, since it enables the discovery of new solutions*" [Bonabeau et al., 1999].

For example, if an ant goes astray, it may discover a new large food source closer to the anthill than any other food source previously found.

4.8.5 The Existence of Multiple Interactions

All cases of SO rely on multiple interactions and generally require a minimal density of mutually tolerant individuals. Moreover, individuals should be able to exploit the results of their own activities as well as those of their teammates, whether they perceive a difference between them or not.

4.9 Characteristics of Self-Organizing Systems

Self-organization has three important characteristics:

- Self-organizing system can accomplish complex tasks with little and simple individual behavior.
- Change in the environment may influence the same system to generate a different task, without any change in the behavioral characteristics.
- Any small difference in individual behavior can influence the collective behavior of the system.

Therefore, social complexity of the system is compatible with simple and identical individuals, as long as communication among the members can provide the necessary amplifying mechanism. In a self-organizing system, individual behavior need not to be changed in order to have different collective behavior.

4.10 Stigmergy

Self-organization primarily describes the macroscopic patterns that can be observed in an ant colony. The concept of stigmergic interactions focuses on the microscopic interactions between the ants of the system.

Stigmergy is an indirect and asynchronous form of communication in which the insects manipulate the environment to transport information to the other insects, which then respond to the change. The insects therefore do not have to be at the same place at the same time as the others to communicate with them. In many ant species colonies, stigmergy refers to the deposition of pheromone by ants while they are moving. Other ants can then smell the deposited pheromone and have a natural tendency to follow the laid trail. This constitutes an asynchronous and indirect communication scheme, where one ant communicates with other ants wherever they are, and it is how positive feedback is created. A little pheromone on a path might lead other ants to follow the same path, depositing even more pheromone, which can lead to a positive feedback effect, if the selected path is good (leading to food) thus

recruiting even more ants to follow the path.

The main elements of a biological stigmergic system are shown in Figure 4-6:

- The insect as the acting individual.
- The pheromone as an information carrier, used to create a dissipation field.
- The environment as a display and distribution mechanism for information.

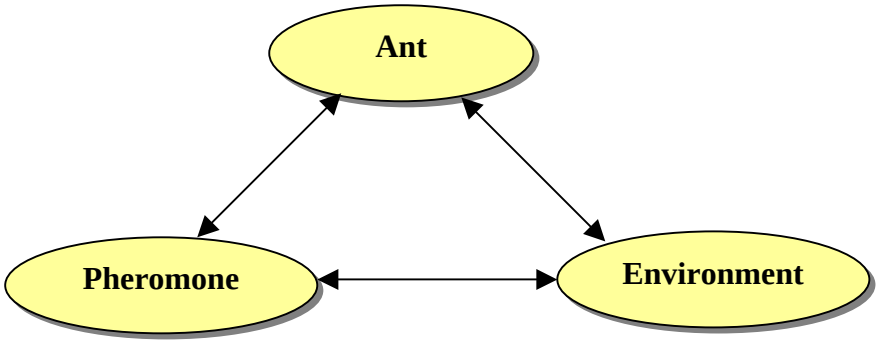


Figure 4-6. Elements of a stigmergic system

We will give a simple example that illustrates the ability of natural ants to find the shortest path in context of artificial ants (see the section 4.11.2).

4.11 Artificial Ants

Artificial ants are not intended to model real ants. They have abilities that cannot be attributed to real ants. The intention in ant algorithms is to keep the ants simple but this is not a requirement since they can be assigned to any needed level of complexity. For instance, artificial ants have a memory about where they have been.

The mechanisms in biological ants described earlier have artificial counterparts. In the following, we will define the presented terms in the context of ant algorithms. Also, some additional modeling is required since we need to express exactly how artificial ants react to their environment. In biological ants, we can observe some randomness in the behavior of the ants, but in order to model this behavior we need to express the randomness of artificial ants. This is a task of the *stochastic state transition* rule that we will describe below. It states how ants react with pheromone and to other stimuli in their environment.

4.11.1 Self-Organization for Artificial Ants

In the sections below, we describe how the different parts of self-organization in colonies of social insects can be modeled by artificial ants. The aim of this modeling is to achieve the same useful properties of self-organization of the biological systems.

4.11.1.1 Stochastic State Transitions

As described in the introduction, we can observe randomness in biological ants, but we need to model this randomness in artificial ants. This is the task of the *stochastic state transition* rule. As the name suggests, the stochastic state transition is responsible for the state transition that ants do. Ants construct solutions by moving on a graph. They perform stochastic walks in the graph, consisting of a series of stochastic steps. If we denote a path as a state, meaning that both a single vertex is a state and a path is a state, the name stochastic state transition is meaningful. It refers to the decision process an ant performs when being in one state and moves to the next (adding a vertex to the path).

Every ant algorithm has a stochastic state transition rule. This rule is used to calculate the stochastic distribution that is used to select the next state. This rule uses local variables at a location such as the pheromone and possibly local heuristic values to calculate the probability distribution that determines what state the ant will move to.

The possibility of an ant making a 'wrong' turn (choosing a state with low probability) is often decisive because it enables the discovery of new solutions. It is up to the stochastic state transition rule to define the relevance of different local variables meaning how much emphasis should be put on pheromone values or on other local heuristics.

Other methods of forcing randomness upon the algorithm can be to normalize the probabilities for a given choice, so that a lower bound on state transition probabilities is set. What has not been pointed out explicitly earlier is that ants must have a termination criterion, which defines when a solution is reached. Stochastic walks in the graph therefore continue until the termination criterion is reached. In the context of the routing problem this stochastic walks have to find the destination where the package is to be routed. Before we continue this discussion, we need to introduce the mechanisms analogous for artificial ants as we did for biological ants.

4.11.1.2 Stigmergy for Artificial Ants

In the domain of ant algorithms, *stigmergy* refers to a similar communication scheme to what natural ants use. They deposit virtual pheromone on the path they have created as a solution. This can either be on the vertices, on the edges or on both.

Two types of pheromone depositing exist. Either the ants can construct a whole solution and then backtrack over the constructed solution depositing pheromone. This is called *delayed trail update* and thus happens when the ants have achieved their termination criterion.

Another option for the ants is to deposit pheromone every time they change state, which is referred to as *step by step trail update*. Thus ants deposit trail on the graph every time they add a vertex to the path. The possibility of combining the two methods also exists.
These trails of pheromone are shared among all ants, so that when an ant deposits pheromone at a location, all others can perceive some effects of this change when they move to the same location. When an ant deposits pheromone, it is added to the pheromone that exists on the location, thus reinforcing the trail value.

4.11.1.3 Positive Feedback

When good solutions to problems are reinforced more (more pheromone is deposited on the path that constitutes the solution) than solutions of less quality, this is referred to as *positive feedback* in the context of ant algorithms.
In ant algorithms, artificial pheromone trails are deposited by ants in amounts proportionate to the quality of the solution. If the solution found is a particular good one, positive feedback sometimes results in an autocatalytic effect (a "snow ball" effect), by which more and more ants are recruited to follow the path that constitutes that solution. This can lead the system to converge to a single solution.
The autocatalytic effect is not always desirable, since it may happen when the ants have found a local optimal solution. The mass recruitment that occurs can leave the ants unable to discover new solutions. This situation is described as a stagnation of the system. It is the job of other mechanisms to ensure that the ants keep looking for better solutions.
Positive feedback implicitly assumes that the stochastic state transition takes pheromone values into account, and uses the relative amount of pheromone on the several possible new states to choose the next state. The general rule should then be "the state with the most pheromone should be chosen with the highest probability".

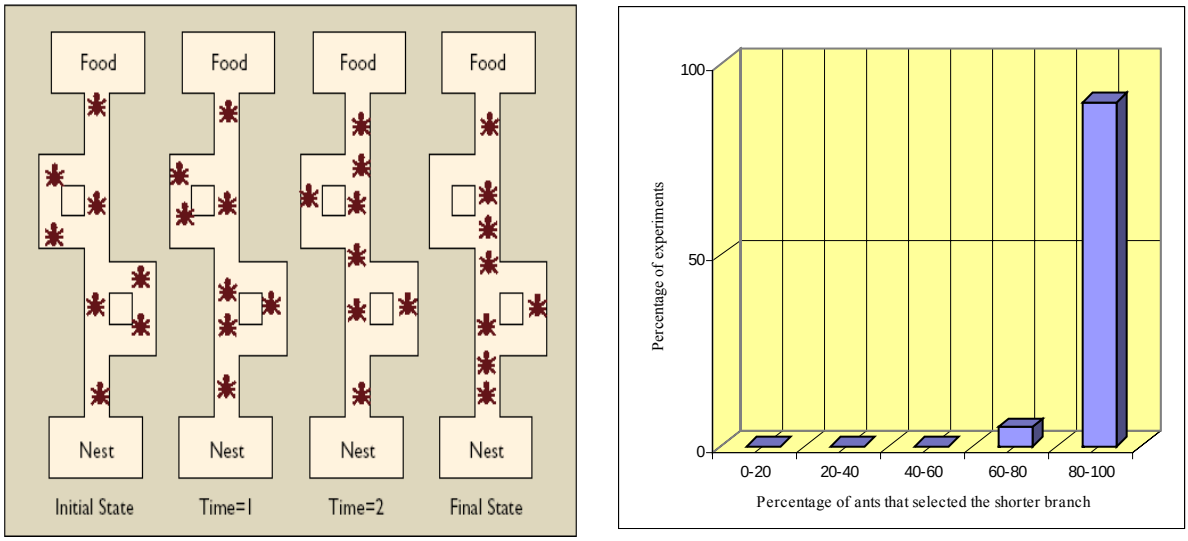
4.11.1.4 Negative Feedback

Negative feedback is often used to avoid premature convergence on a suboptimal solution (stagnation). If premature convergence happens, the ants are "satisfied" with the result found so far and stop looking for new – and may be better – solutions.
An example of negative feedback in ant algorithms is artificial pheromone evaporation. Artificial pheromone evaporation is usually implemented as a reduction in pheromone, which reduces the amount of pheromone to a percentage of the original amount. This implies that the amount of pheromone removed is proportionate to the (original) amount of pheromone. This

means that good paths need to be continually reinforced by ants in order to ensure that they retain their pheromone value since it evaporates faster.

4.11.2 Double Bridge Experiment

The double bridge experiment [Goss et al., 1989] is an important experiment in the field of ant algorithms. In fact, it gave the initial inspiration to all the research work that led to the definition of the ACO metaheuristic [Dorigo et al., 1991]. In the double bridge experiment, see Figure 4-7, an ant nest is connected to a food source via paths of different length.



(a) (b)

Figures adapted from [Goss et al., 1989] and [Bonabeau et al., 2000]

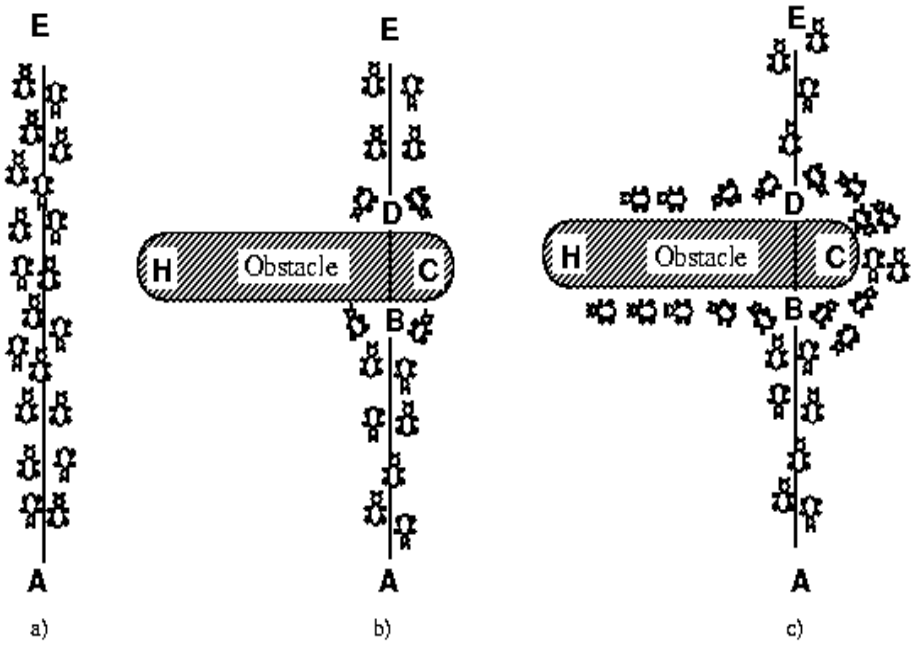
(a) Ants start exploring the double bridge. Eventually, most of the ants choose the shortest path.
(b) Distribution of the percentage of ants that selected the shorter path.

Figure 4-7. Double bridge experiment

At start time, all ants are in the nest and they are left free to move. The experimental apparatus is built in such a way that the only way for the ants to reach the food is by using one of the bridge branches. In the initial phase, the ants move randomly and they choose between the shorter and the longer branch with equal probability. While walking, ants deposit on the ground a pheromone trail. Those ants choosing the shortest branch will be the first to find the food and go back to the nest, so the pheromone trail on the shortest branch will grow faster. Forthcoming ants choose with higher probability those directions marked by stronger pheromone concentration. This autocatalytic (positive feedback) process is at the heart of the auto-organizing behavior that very quickly leads all the ants to choose the shortest branch. A

similar mechanism can be used by opportunely defined artificial ants to find minimum cost paths on graphs.

Consider for example, other experimental setting shown in Figure 4-8. There is a path along which ants are walking for example from the nest A to food source E and vice versa (Figure 4-8a). Suddenly, an obstacle appears and the path is cut off. Then at position B, the ants walking from A to E, or at position D those walking in the opposite direction have to decide whether to turn right or left (Figure 4-8b).



a) Ants follow a path between points A and E
b) An obstacle is interposed on the path; ants can choose to go around it following two different paths with equal probability
c) On the shorter path more pheromone is laid down increasing the probability of that route being preferred by following ants

Figure 4-8. Ants facing an obstacle

Figure is taken from Dorigo 1996

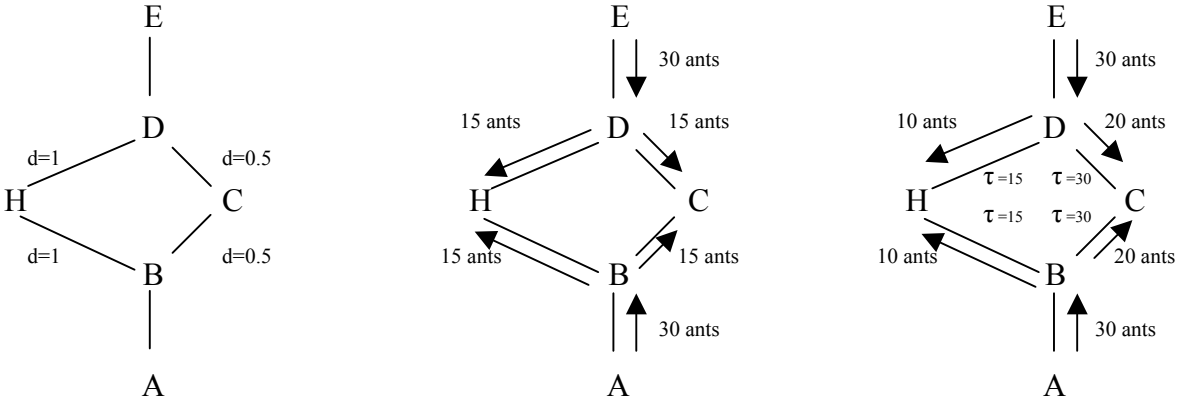
The choice is influenced by the intensity of the pheromone trails left by preceding ants. A higher level of pheromone on the right or left path gives ants a stronger stimulus and thus a higher probability to turn right or left.The first ant reaching point B or D has the same probability to turn right or left as there was no previous pheromone on the two altemative paths. Because path BCD is shorter than BHD, the first ant following it will reach D before the first ant following BHD (Figure 4-8c). The result is that an ant returning from E to D will find a stronger trail on path DCB caused by the half of all the ants that by chance decided to approach the obstacle via DCBA and by the already arrived ones coming via BCD: they will

therefore prefer in probability path DCB to path DHB. As a consequence, the number of ants following path BCD per unit of time will be higher than the number of ants following BHD. This causes the quantity of pheromone on the shorter path to grow faster than on the longer one and therefore, the probability with which any single ant chooses the path to follow is quickly biased toward the shorter one. The final result is that all ants will quickly choose the shorter path.

4.11.3 A Bit More Formal

We have described above how ants act in the real world. We can formalize it a little as follows: Simple mathematical modeling of Figure 4-8 is represented in Figure 4-9 using the same points A, B, C, D, E and H.

This diagram represents the map that the ants have to traverse. At each time unit *t*, each ant moves a distance *d*=1. All ants are assumed to move at the same time. At the end of each time step, each ant lays down a pheromone trail of intensity 1 on the edge (route) it has just travelled along.



a) At time t=1 there is no trail on the graph edges; therefore, ants choose to turn right or left with equal probability.
b) At time t=2 the trail τ is stronger on the shorter edge DCB, which is therefore, in the average, preferred by ants.

Figure 4-9. A mathematical modeling of ants experiment

- At *t*=0 there is no pheromone on any edges of the graph (we can represent this map as a graph). Assume that 30 ants are moving from E to A and another 30 ants are moving from A to E.
- At *t*=1 there will be 30 ants at B and 30 ants at D. At this point they have a 0.5 probability to each way they will turn. We assume that half of ants go one way and the other half of ants go the other way.

- At *t=2* there will be 15 ants at D (who have travelled from B, via C) and 15 ants at B (who have travelled from D, via C). There will be 30 ants at H (15 from D and 15 from B).

The intensities on the edges will be as follows.

ED = 30, AB = 30, BH = 15, HD = 15, BC = 30 and CD = 30

The process continues until all of the ants will eventually choose the shortest path.

If we now introduce another 60 ants into the system (30 from each direction) more ants are likely to follow the BCD rather than BHD as the pheromone trail is more intense on the BCD route. But, as we are interested in exploring the search space rather than simply plotting a route, we need to allow the ants to explore paths and follow the best paths with some probability in proportion to the intensity of the pheromone trail. That means we do not want them simply to follow the route with the highest amount of pheromone on it, else our search will quickly settle on a suboptimal (and probably very suboptimal) solution.

Therefore, the probability of an ant following a certain route is a function, not only of the pheromone intensity but also a function of what the ant can see (which Dorigo calls *visibility*). Therefore, if an ant is at D (in the above graph) and the pheromone trail is more intense on the route to H then the ant would follow that route if we only considered the pheromone intensity. However, by looking at the distances to the possible routes, it would see that the shorter route is to C. It would take this into account when deciding (probabilistically) which route to choose.

We also need to stop the pheromone trail building up without bound, because that will ultimately lead to all the ants following the same route. Therefore, we need to implement some form of "*evaporation*" that reduces the pheromone intensity at each time unit. Pheromone evaporation is the process by means of which the pheromone trail intensity on the components decreases over time. From a practical point of view, pheromone evaporation is needed to avoid a too rapid convergence of the algorithm toward a suboptimal region. It implements a useful form of forgetting and favouring the exploration of new areas in the search space.

4.12 Applications of Ant Colony Optimization Algorithms

There are now available numerous successful implementations of the ACO meta-heuristic applied to a number of different combinatorial optimization problems [Dorigo et al., 1999]. The most studied problems have been the traveling salesman, the quadratic assignment and routing in telecommunication networks. Dorigo and Gambardella have also proposed new hybrid versions of ACO with local search. In problems like the quadratic assignment problem

and the sequential ordering problem, their hybrid ant colony system outperforms all known algorithms on vast classes of benchmark problem. ACO alogrithms give reasonably good results in comparison with the best available heuristic approaches.

Looking at these implementations, it is possible to distinguish among two classes of applications (see list of different applications in appendix C):

- Static combinatorial optimization problems.
- Dynamic combinatorial optimization problems.

In static problems, the key-points of the problem are defined at the beginning and do not change while the problem is being solved. A typical example of such problems is the classic traveling salesman problem, in which city locations and their relative distances are part of the problem definition and do not change at run time.

In dynamic problems, the problem changes as a function of itself, thus the algorithms used to solve such problems must be able to adapt "online" to the changes. The typical example of such problem is the network routing problem.

Examples of applications to static combinatorial optimization problems are:

- **Traveling Salesman Problem,** where a salesman must find the shortest route by which he can visit a given number of cities, each city exactly once.
- **Quadratic Assignment Problem**, the problem of assigning *n* facilities to *n* locations so that the costs of the assignment are minimized.
- **Job-Shop Scheduling Problem,** where a given set of machines and set of job operations must be assigned to time intervals in such a way that no two jobs are processed at the same time on the same machine and the maximum time of completion of all operations is minimized.
- **Vehicle Routing Problem**, the objective is to find minimum cost vehicle routes such that:

(a) Every customer is visited exactly once by exactly one vehicle;
(b) For every vehicle the total demand does not exceed the vehicle capacity;
(c) The total tour length of each vehicle does not exceed a given limit;
(d) Every vehicle starts and ends its tour at the same position.

- **Shortest Common Super sequence Problem**, where – given a set of strings over an alphabet – a string of minimal length that is a super sequence of each string of the given set has to be found (a super sequence *S* of string *A* can be obtained from *A* by inserting zero or more characters in A).
- **Graph-Coloring Problem,** which is the problem of finding a coloring of a graph so that the number of colors used is minimal.

- **Sequential Ordering Problem**, which consists of finding a minimum weight Hamiltonian path 2 on a directed graph with weights on the arcs and on the nodes, subject to precedent constraints among the nodes.

The main focus of applications to the second class of problems (dynamic combinatorial optimization problems) is on communication networks, in particular on routing problems. Routing answers the question "how to direct data traffic (e.g. phone calls) through a network?" i.e. which node to choose next by a data packet entering the network. Routing mainly consists of building, using and updating routing tables.

Implementations for communication networks can be divided in two classes:

- **Connection-Oriented Network Routing**, where all packets of the same session follow the same path selected by a preliminary setup phase.
- **Connectionless Network Routing** where data packets of the same session can follow different paths (Internet-type networks).

4.13 Steps to Solve a Problem Using ACO

From the currently known ACO applications, we can identify some guidelines of how to study new problems by ACO. These guidelines can be summarized by the following six design tasks [Cordon et al., 2002]:

- Represent the problem in the form of sets of components and transitions or by means of a weighted graph that is travelled by the ants to build solutions.
- Appropriately define the meaning of the pheromone trails, i.e., the type of decision they bias. This is a crucial step in the implementation of an ACO algorithm. A good definition of the pheromone trails is not a trivial task and it typically requires insight into the problem being solved.
- Appropriately define the heuristic preference to each decision that an ant has to take while constructing a solution, i.e., define the heuristic information associated to each component or transition. Notice that heuristic information is crucial for good performance if local search algorithms are not available or can not be applied.
- If possible, implement an efficient local search algorithm for the problem under consideration, because the results of many ACO applications to NP-hard combinatorial optimization problems show that the best performance is achieved when coupling ACO with local optimizers [Dorigo and Di Caro, 1999] [Dorigo and Stützle, 2003].
- Choose a specific ACO algorithm and apply it to the problem being solved, taking the previous aspects into consideration.

- Tune the parameters of the ACO algorithm. A good starting point for parameter tuning is to use parameter settings that were found to be good when applying the ACO algorithm to similar problems or to a variety of other problems. An alternative to time-consuming personal involvement in the tuning task is to use automatic procedures for parameter tuning [Birattari et al., 2002].

It should be clear that the above steps can only give a very rough guide to the implementation of ACO algorithms. In addition, the implementation is often an iterative process, where with some further insight into the problem and the behavior of the algorithm, some initially taken choices need to be revised. Finally, we want to insist on the fact that probably the most important of these steps are the first four, because a poor choice at this stage typically can not be made up with pure parameter fine-tuning.

In [Bonabeau et al., 1999], it is mentioned that the following points must be defined to build an ACO algorithm:

- An appropriate representation of the problem, which allows the ants to incrementally construct/modify solutions through the use of a probabilistic transition rule based on the amount of pheromone in the trail and on a local heuristic;
- A heuristic function that measures the quality of items that can be added to the current partial solution;
- A method to enforce the construction of valid solutions, i.e. solutions that are legal in the real-world situation corresponding to the problem definition;
- A rule for pheromone updating, which specifies the pheromone trail;
- A probabilistic rule of transition based on the value of the heuristic function and on the contents of the pheromone trail.

An ACO algorithm iteratively performs a loop containing two basic procedures, namely:

- A procedure specifying how the ants construct/modify solutions of the problem to be solved;
- A procedure to update the pheromone trails.

The construction/modification of a solution is performed in a probabilistic way. The probability of adding a new item to the current partial solution is given by a function that depends on a problem-dependent heuristic and on the amount of pheromone deposited by ants on the trail in the past. The updates in the pheromone trail are implemented as a function that depends on the rate of pheromone evaporation and on the quality of the produced solution.

Figure 4-10 presents the generic ant algorithm.

- The first step consists mainly of the pheromone trail initialization.
- The second step is the iteration step, where each ant constructs a complete solution to the problem according to a probabilistic state transition rule. The state transition rule depends mainly on the state of the pheromone.

Step1: Initialization
- Initialize the pheromone trail

Step 2: Iteration
- For each Ant Repeat
  - Solution construction using the current pheromone trail
  - Evaluate the solutions constructed
  - Update the pheromone trail
- Until stopping criteria

Figure 4-10. A generic ant algorithm

Once all ants generate a solution, a global pheromone updating rule is applied in two phases:

- An evaporation phase where a fraction of the pheromone evaporates,
- A reinforcement phase where each ant deposits an amount of pheromone that is proportional to the fitness of its solution. This process is iterated until reaching stopping criteria.

The following Figure 4-11 represents a generalized flowchart of the ant algorithm.

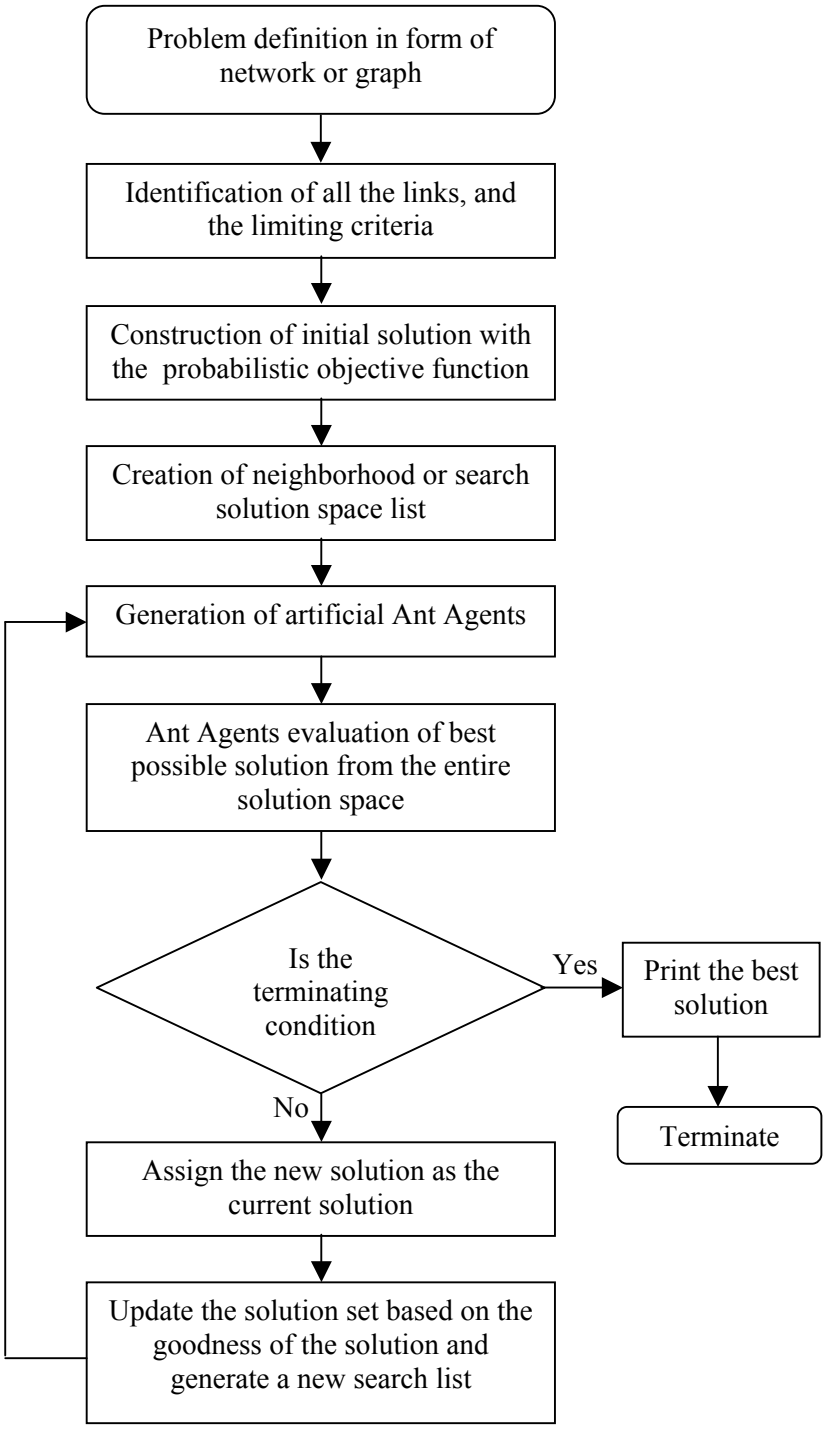


Figure 4-11. Generalized flowchart for ant algorithm

Bibliography

[Birattari et al., 2002] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A Racing Algorithm for Configuring Metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 11-18, 2002.

[Blum and Roli, 2003] C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, Vol. 35, No. 3, pages 268–308, 2003.

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

[Bonabeau et al., 2000] E. Bonabeau, M. Dorigo, and G. Theraulazß. *Inspiration for Optimization from Social Insect Behaviour*. Nature, vol. 406, 2000.

[Bullnheimer et al., 1997] B. Bullnheimer, R. F. Hartl, and C. Strauß. A New Rank Based Version of the Ant System – A Computational Study. *Technical Report POM-10/97*, *Institute of Management Science*, University of Vienna, 1997.

[Colorni et al., 1991] A. Colorni , M. Dorigo , and V. Maniezzo. Distributed Optimization by Ant-Colonies. In *Proceedings of the European Conference on Artificial Life (ECAL'91)*, edited by F. Varela and P. Bourgine, Cambridge, MIT Press, pages134-142, 1991.

[Colorni et al., 1992a] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by Ant Colonies. *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 134 - 142, 1992.

[Cordon et al., 2002] O. Cordon, F. Herrera, and T. Stützle. A Review on Ant Colony Optimization Metaheuristic: Basis, Models and New Trends. *Mathware & Soft Computing*, vol. 9, pages 141-175, 2002.

[Dorigo and Glover, 1999] M. Dorigo and F. Glover (Eds.). *New Ideas in Optimization*. McGraw Hill, London, UK, pages 11–32, 1999.

[Dorigo and Stützle, 2003] M. Dorigo and T. Stützle. The Ant Colony Optimization Metaheuristic: Algorithms, Applications and Advances. In F. Glover and G. Kochenberger, (Eds.), *Handbook of Metaheuristics*, Kluwer Academic Publishers, pages 251-285, 2003.

[Dorigo et al., 1991] M. Dorigo, V. Maniezzo, and A. Colorni. Positive Feedback as a Search Strategy. *Technical Report 91-016*, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

[Dorigo et al., 1991a] M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: An Autocatalytic Optimizing Process. *Technical Report 91-016*, *Dipartimento di Elettronica*, Politechnico di Milano, Italy, 1991.

[Dorigo et al., 1991b] M. Dorigo, V. Maniezzo, and A. Colorni. Positive Feedback as a Search Strategy. *Technical Report 91-016*, *Dipartimento di Elettronica*, Politecnico di Milano, Italy, 1991.

[Dorigo et al., 1996] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1), pages 29–41, 1996.

[Dorigo et al., 1999] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, MIT Press, 1999.

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

[Feldman and Ballard, 1982] J. A. Feldman and V. Ballard. Connectionist Models and their Properties. *Cognitive Science*, vol. 6, pages 205–254, 1982.

[Gambardella and Dorigo, 1995] L. M. Gambardella and M. Dorigo. Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. In *Proceedings of ML-95, Twelfth Intern. Conf. on Machine Learning*, pages 252–260, 1995.

[Gambardella, 2004] L. M. Gambardella. Engineering Complex Systems: Ant Colony Optimization (and in General Natural Inspired Metaheuristic) to Model and to Solve Complex Dynamic Problems, 2004.

[Glover, 1986] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Comput. Oper. Res. 13*, pages 533–549, 1986.

[GoldBerg,1991] D. E. GoldBerg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1991.

[Goss et al., 1989] S. Goss, S. Aron, J.L. Deneubourg, and J.M. Pasteels. Self-Organized Shortcuts in the Argentine Ant. *Naturwissenschaften*, vol. 76, pages 579–581, 1989.

[Grassé, 1959] P. P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez Bellicositermes natalensis et Cubitermes sp. La theorie de la stigmergie: Essai d'interpretation des termites constructeurs. *Ins. Soc*. 6, pages 41-83, 1959.

[Hertz, 1991] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*, Addison-Weseley, 1991.

[M. Dorigo et al., 1996] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transaction on System, Man, and Cybernetics-Part B*, vol. 26, No. 2, pages 656-665, 1996.

[Metaheuristics Network Website, 2000] *Metaheuristics Network Website 2000*. http://www.metaheuristics.net.

[Narendra and Thathachar, 1989] K. Narendra and M. Thathachar. *Learning Automata: An Introduction*. Prentice-Hall, 1989.

[Osman and Laporte, 1996] I. H. Osman, G. Laporte. Metaheuristics: A bibliography. *Ann. Oper. Res. 63*, pages 513–623, 1996.

[Reeves, 1993] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publishing, Oxford, England, 1993.

[Rubistein, 1981] R.Y. Rubistein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, 1981.

[Stützle, 1999a] T. Stützle. Iterated Local Search for The Quadratic Assignment Problem. *Tech. rep. aida-99-03*, FG Intellektik, TU Darmstadt. 1999.

[Stützle, 1999b] T. Stützle. *Local Search Algorithms for Combinatorial Problems - Analysis, Algorithms and New Applications*. DISKI - Dissertationen zur Künstliken Intelligenz. infix, Sankt Augustin, Germany, 1999.

[Voß et al. 1999] S. Voß, S. Martello, I. H. Osman, and C. Roucairol. Meta-heuristics Advances and Trends in Local Search Paradigms for Optimization. *Kluwer Academic Publishers*, Dordrecht, The Netherlands, 1999.

[Watkins,1989] Watkins. *Learning with Delayed Rewards*. PhD thesis, University of Cambridge, Psychology Department, University of Cambridge, England, 1989.

[Yovits et al., 1962] M.C.Yovits, G.T. Jacobi, and G.D. Goldstein. Self-Organizing Systems. Washington D.C., McGregor & Werner, 1962.

Chapter
5

HYBRID SIMULATION FOR ROUTING USING ANT COLONY WITH QUEUING ANALYSIS ALGORITHMS

5.1 Introduction

Ad hoc networks can operate without fixed infrastructure and survive rapid changes in the network topology. Usually nodes are mobile and use wireless communication links. In recent years, many routing algorithms for ad hoc networks have been proposed. The algorithms are most often compared using simulation.

There are three different ways to model and evaluate networks; *formal analysis*, *real life measurements* and *simulation* [Särelä, 2004]. The dynamic nature of ad hoc networks makes them hard to be studied by formal analysis. Some formal techniques that have been used in static networks include Petri nets, stochastic processes, queuing theory and graph theory. Since ad hoc networks are still mainly a research subject, most scenarios that will be used in are still unknown. One of those scenarios that are known is military networks. Thus, use of real life measurements is currently almost impossible and certainly costly. The commonly used alternative is to study the behavior of the protocols in a simulated environment. The purpose of simulation is to create an artificial environment, usually a computer program, that captures the essential characteristics of the phenomena that is being studied. Simulation is an economically viable way to create a statistically significant amount of test runs. For these reasons, simulation is a much used tool for comparing ad hoc routing protocols. There are

several network simulators that can be used for studying mobile ad hoc networks such as *ANSim* (Ad hoc Network Simulation), *Glomosim* and *NS-2*.

The main goal of this chapter is to present a novel complete hybrid simulation model for routing in *Mobile Ad hoc Networks* (MANETs) inspired by *Ant Algorithm* in combination with *queuing network analysis*. The routing algorithm is based on a type of learning algorithm, similar to one described in AntNet, that provides deterministic forwarding of message packets (ant packets) from source node to destination node. Our algorithm assumes that all links in the network are bi-directional and all the nodes in the network fully cooperate in the operation of the algorithm.

5.2 Mobility Models Used in Simulation

Ad hoc network researchers face the problem that it is unknown how nodes will operate in real life situations, especially, how they will move. In order to create meaningful simulation results, good understanding of mobility and its effects on ad hoc routing is required. Ad hoc mobility models are used to describe the movement of nodes, so they play a key part in simulating ad hoc networks. Each model yields an algorithm that is used to randomize the movements of nodes.

Currently, there are two types of mobility models used in simulation of networks, *traces* and *synthetic* models [Camp et al., 2002]. Traces are those mobility patterns that are observed in real life systems. Traces provide accurate information, especially when they involve a large number of mobile nodes MNs and an appropriate long observation period. On the other hand, synthetic models attempt to realistically represent the behavior of MNs without the use of traces.

The synthetic models are divided into two categories, *entity mobility models* and *group mobility models* [Davies, 2000]. The entity mobility models randomize the movements of each individual node and represent MNs whose movements are independent of each other. The group mobility models have groups of nodes that stay close to each other and then randomize the movements of the group and represent MNs whose movements are dependent on each other. The node positions also vary randomly around the group reference point.

In our simulation process, we will simulate two of these mobility models; *Random Waypoint Mobility* (RWM) model and *Boundless Simulation Area Mobility* (BSAM) model. There are a number of other mobility models to be found in the Appendix D.

5.2.1 Random Waypoint Mobility (RWM)

The Random Waypoint Mobility (RWM) model includes pause times between changes in direction and/or speed [Johnson and Maltz, 1996]. A MN begins by staying in one location for a certain period of time (i.e., a pause time). Once this time expires, the MN chooses a random destination in the simulation area and a speed that is uniformly distributed between [min_{speed}, max_{speed}]. The MN then travels toward the newly chosen destination with the selected speed. Upon arrival, the MN pauses for a specified time period before starting the process again. Figure 5-1 shows a traveling pattern example of a MN using the RWM model starting at a randomly chosen point or position. RWM model is also a widely used mobility model (e.g., [Broch et al., 1998], [Chiang and Gerla, 1998], [Garcia and Spohn, 1999] and [Johansson et al., 1999]).

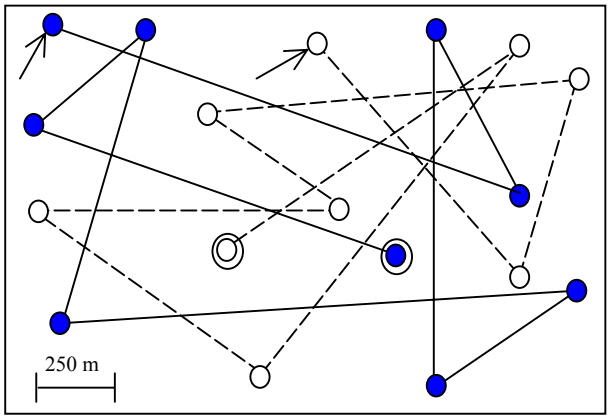


Figure 5-1. Random waypoint mobility model

5.2.2 Boundless Simulation Area Mobility (BSAM)

In the *Boundless Simulation Area Mobility* (BSAM) model there exists a relationship between the previous direction θ, velocity v of a MN and its current direction, velocity. This relationship limits the change in direction and speed per time unit that generates more realistic movement patterns. Both the velocity vector $V = (v,\theta)$ and the MN's position (x,y) are updated at every Δt time step according to the following formulas:

$$v(t + \Delta t) = min[max(v(t) + \Delta v, 0), V_{max}];$$

$$\theta(t + \Delta t) = \theta(t) + \Delta\theta;$$

$$x(t + \Delta t) = x(t) + v(t) * \cos\theta(t);$$

$$y(t + \Delta t) = y(t) + v(t) * \sin\theta(t);$$

where V_{max} is the maximum velocity defined in the simulation, Δv is the change in velocity which is uniformly distributed between $[-A_{max}*\Delta t, A_{max}*\Delta t]$, A_{max} is the maximum acceleration

of a given MN, $\Delta\theta$ is the change in direction which is uniformly distributed between [$-\alpha*\Delta t$, $\alpha*\Delta t$] and α is the maximum angular change in the direction travelled by a MN. Another specialty of this mobility model has given it its name, the rectangular simulation area is folded to form a torus with the effect that any node moves out of the simulation area enters it again at the opposite side thus, creating a boundless simulation area (e.g. a node leaving the simulation area at the top enters at the bottom again) as shown in Figure 5-2. Since the changes in direction and speed are limited, the resulting moves lack of abrupt changes in direction and speed, which makes the movement patterns more realistic. This realism only persists, if the components working with the resulting movements are aware of the specialty of the boundless simulation area. Figure 5-3 illustrates an example of a MN's path using the Boundless Simulation Area Mobility model.

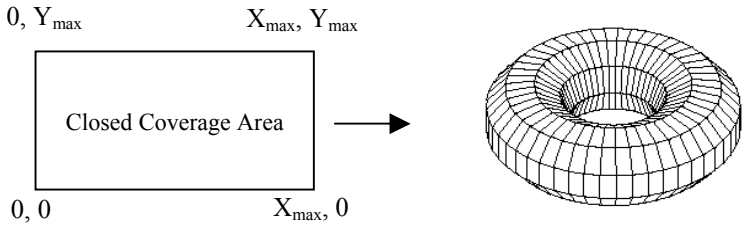


Figure 5-2. Rectangular simulation area mapped to a torus in the BSAM model

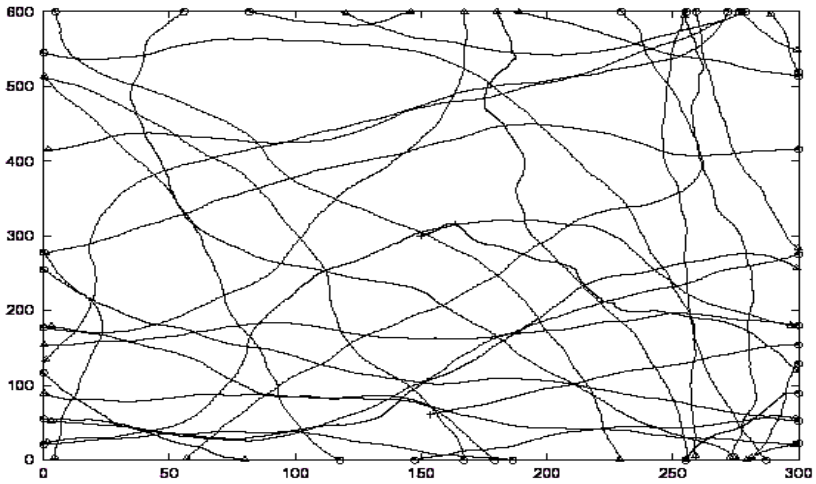


Figure 5-3. Traveling pattern of a MN using the BSAM model

5.3 Network Configuration

Our MANET is mapped on a directed graph with *N* nodes and *M* links. All the links are viewed as bit pipes characterized by a bandwidth (bits/sec) and a transmission delay (sec). For this purpose, every node of type store-and-forward (i.e., switch element) holds a buffer

space (queue) where the incoming and the outcoming packets are stored as indicated in Figure 5-4.

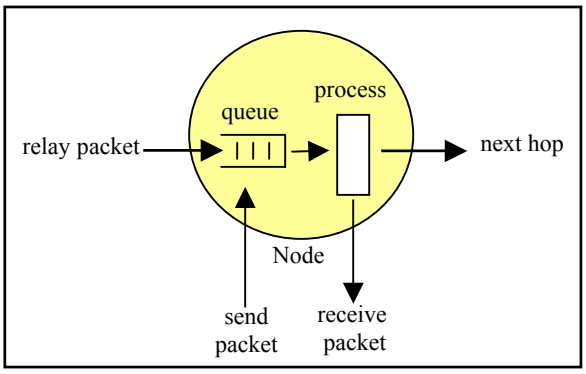


Figure 5-4. Node model

This buffer is a shared resource among all the queues attached to every incoming and outgoing link of the node. Traveling packets can be data packets or routing packets. Packets are queued and served on the basis of a First-In-First-Out (FIFO) policy. A packet reads from the routing table the information about which link to use to follow the path toward its target node. When link resources are available, they are reserved and the transfer is set up. The time a packet takes to move from one node to a neighboring one depends on its size and on the link transmission characteristics. On packet's arrival, if there is not enough buffer space to hold it, the packet will be discarded.

The links between any two mobile nodes *i* and *j* in our network are bi-directional links and illustrated in Figure 5-5.

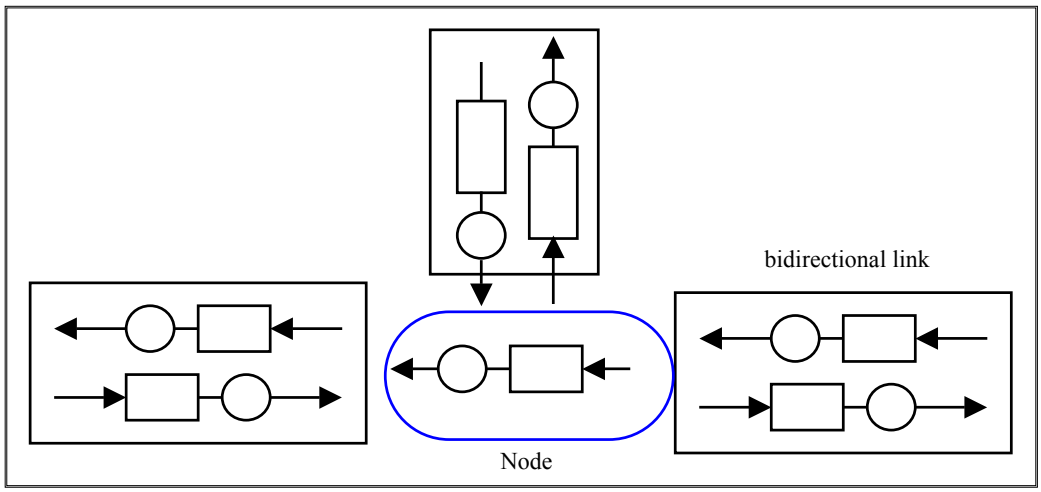


Figure 5-5. A mobile node and its associated bi-directional links

5.4 A Complete Scenario of the Simulation Algorithm

The following flowchart in Figure 5-6 shows the suggested model to minimize end-to-end delay in a MANET [Tarek and Mueller-Clostermann, 2004].

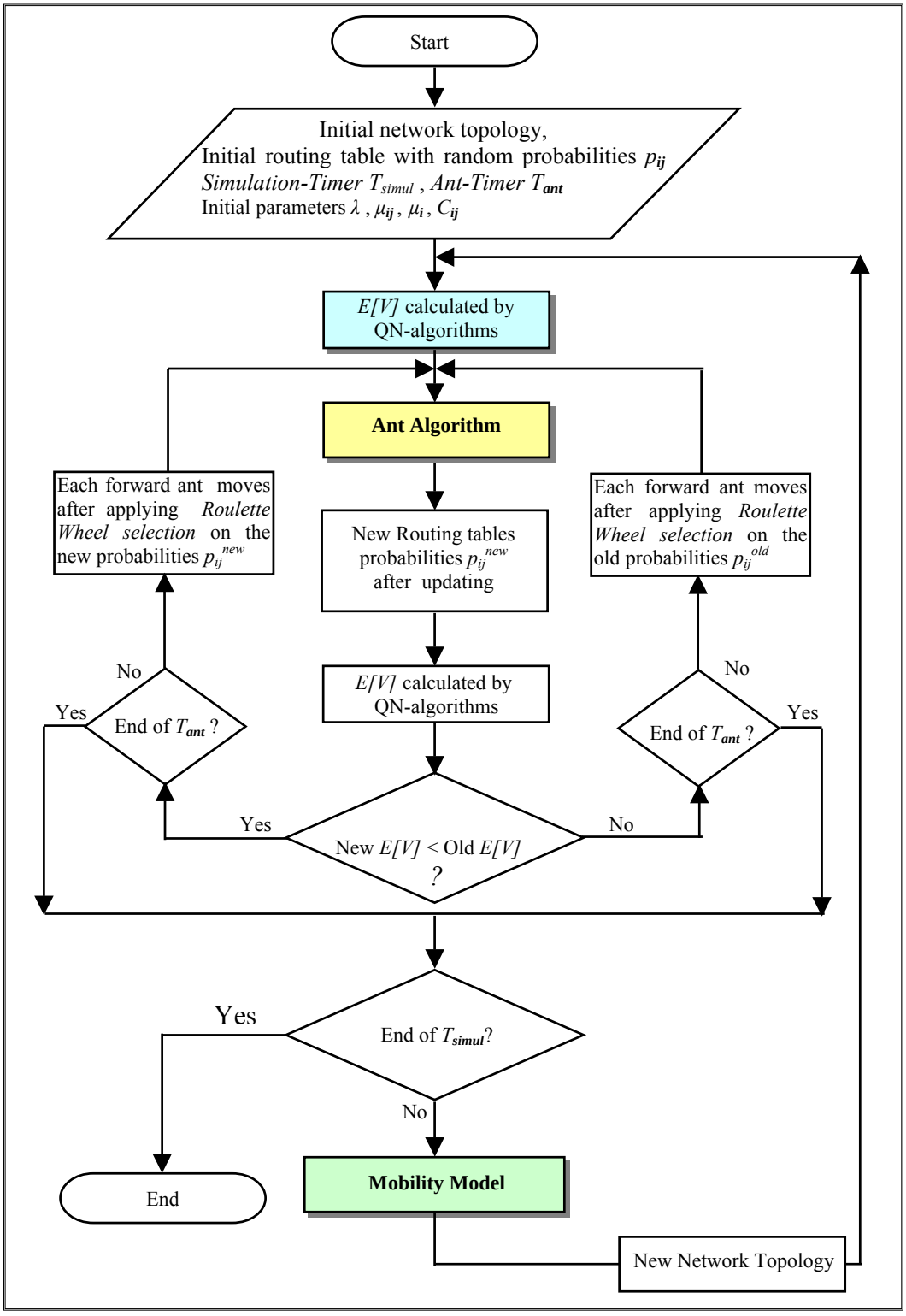


Figure 5-6. Flowchart of the simulation algorithm

5.4.1. Simulation Environment and Assumptions Initialization

- **Topology Initialization**

The simulation environment consists of mobile nodes that are constantly moving on a simulation rectangular area 1000m×800m that is chosen to have longer distances between the nodes than in a quadratic area, i.e. packets are sent over more hops. MNs are moving according to one of the simulated mobility models; *Random Waypoint Mobility* (RWM) model and *Boundless Simulation Area Mobility* (BSAM) model. Figure 5-7 shows an example for the initial topology of a set of MNs that are located in a simulation area and an ant will begin to discover its route from source to destination node.

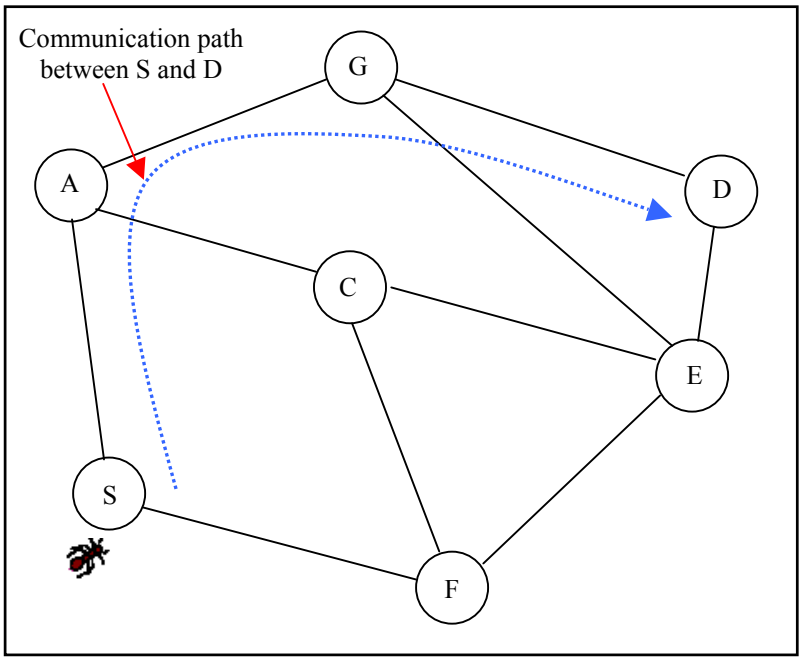


Figure 5-7. Initial network topology in a field of 1000×800 m^2 at T_{ant}=0 sec

- **Routing Table Initialization**

A routing table is a locally stored conventional table with rows and columns and it exists for every node of the network. The routing table of a node contains columns which represent the neighbor nodes that can be reached from that node. For every node of the network, except the node itself, the routing table contains rows which represent the destinations. The entries in the table are numeric values between 0 and 1 which represent the probability to reach other nodes represented by the rows along the neighbor nodes in shortest time. The probabilities in each row sum up to 1. Its entries are initialized with probabilities 1/N for each neighbor as the next hop for the respective destination, where N is the number of neighbors of the node. The probability value P_{in} which expresses the goodness of choosing n as next node when the destination node is i, is stored for each pair (i, n) with the constraint:

$$\sum_{n \in N_k} P_{in} = 1 \quad \in [1, N_k],\ N_k = \{\text{neighbors}(k)\};$$

The uniform probabilities assigned to all the neighbors indicate that nothing is known about the state of the network.

The routing table at each node is organized in the form (*Destination, Next hop, Probability*) on a per-destination basis. It contains the goodness values for a particular neighbor to be selected as the next hop for a particular destination. A special notation is used to give each entry a unique name: $P_{destinationNode,\ neighborNode}$. The first variable, *destinationNode*, defines the row in the routing table and the second variable, *neighborNode*, defines the column. The probability $P_{destinationNode,\ neighborNode}$, is the probability to reach the destination node in shortest time by going along the neighbor node from the current node. The information that will be read and written in the data structure stored in each network node *I* is indicated in Figure 5-8.

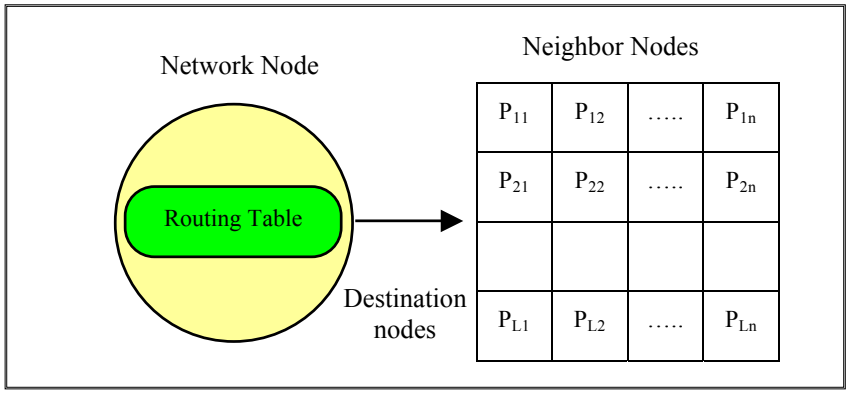


Figure 5-8. Routing table structure imbedded in a network node

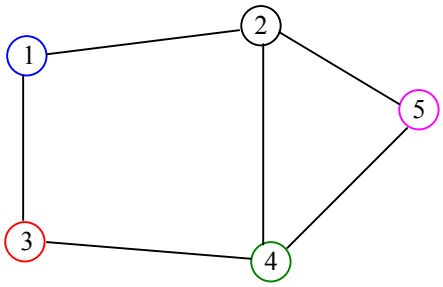


Figure 5-9. An example of topology with varying connectivity

Let's consider an example of a routing table. Table 5-1 is the routing table of node 5 from the network shown in Figure 5-9. Because the neighbor nodes of node 5 are the nodes 2 and 4, there exists a column for each of them. For every other node, except node 5, there is a row in the routing table. The probability to reach node 2 in shortest time by directly going to node 2

($P_{node2, node2}$) is very high, namely 0.95. It is also possible to go to node 2 along node 4, but the probability to go there in shortest time ($P_{node2, node4}$) is very small, only 0.05. The probabilities to go to node 1 in shortest time do not deviate so strongly from each other. ($P_{node1, node2}$) is 0.59 and ($P_{node1, node4}$) is 0.41. That is because the route to node 1 along node 2 is not much shorter than the route to node 1 along node 4.

Table 5-1. Routing table of node 5 from the network of Figure 5-9

Destination node ↓ Next node →	node 2	node 4
node 1	0.59	0.41
node 2	0.95	0.05
node 3	0.27	0.73
node 4	0.06	0.94

The probabilities in the routing tables can be compared with the strength of the pheromone trail. The higher the probability the stronger the pheromone trail. Note that the routing tables only contain local information on the best routes and no global information. To compute the shortest routes in time needed to send a packet from one node in the network to another node, the information of all routing tables on the path from source to destination node is needed. The next step from a node toward the destination node is always determined by using the routing table of the node. The next node is the node with highest probability in the row that represents the destination. By that, packets are always routed to their destination along the path with highest probability. Since this path represents the strongest pheromone trail to the destination, it should be the shortest known path in time.

- **Parameter Initialization**

We must also at the beginning of our simulation process define the *arrival rate* λ, which is the average number of packets arrived in each unit of time. Another parameter to be defined is the *service rate* μ_{ij} (Bandwidth) between each two nodes i and j in the network, which is the average number of packets serviced in each time unit. This parameter μ_{ij} will be given in a matrix called service rate table.

- **Timer Initialization**

At the beginning of the simulation process, we initialize two different timers. Ant timer T_{ant} is the time for the ant algorithm to work on a given network topology (termination condition for ant algorithm). For example, T_{ant} =30 sec. means that the ant algorithm will finish its work for every new network topology in 30 sec. The Simulation timer T_{simul} is the whole time for the

simulation process in which the ant algorithm will be repeated each T_{ant} time. For example, if T_{simul} =180 sec. and T_{ant} =30 sec. then, the ant algorithm will be repeated 6 times on 6 different network topologies.

5.4.2 Queuing Network Analysis

Kleinrock [Kleinrock, 1964] [Wong, 1978] first derived an expression for the mean End-to-End delay in a message-switched network. He chose to model a data network as a network of communication channels whose purpose was to move data messages from their origin to their destination. Each channel was modeled as a server serving a queue of data messages awaiting transmission. The main metric used for the performance of the network was *T*, the average time messages took to move across the network.

One of the first general results was an exact expression for the mean delay experienced by a message as it passed through a network. The evaluation of this delay required the introduction of an assumption (*Kleinrock's Independence Assumption*) without which the analysis remains intractable, and with which the analysis becomes quite straightforward. The independence assumption assumes that "*the length of a message is chosen independently from the exponential distribution each time it enters a switching node in the computer network*".

In our hybrid simulation, we will use *Kleinrocks's Independence Assumption* to calculate the delay of sending a packet through a mobile ad hoc network.

The End-to-End delay is defined as time between the moment in which the source wants to send a packet and the moment the packet reaches its destination.

End-to-End delay = $T_{destination\ receives\ packet} - T_{source\ wants\ to\ send\ packet}$

The End-to-End delay is important because nowadays, many applications (e.g. IP telephony) need a small latency to deliver usable results. It shows the suitability of the protocol for these applications. The expected response time delay to send packets from a source node *S* to a destination node *D* is the sum over the response times at all links and nodes visited along the way [Haverkort 1998]:

$$E[R(S,D)] = \sum E[R_{ij}] + \sum E[R_i] \qquad (1)$$

Where $E[R_{ij}] = \frac{1}{\mu C_{ij} - \lambda_{ij}}$, is the expected delay at all links, μc_{ij} is the number of packets that can be transmitted over a link *ij* (packets/sec.), and λ_{ij} is the arrival rate over a link *ij*.

$E[R_i] = \frac{\rho_i E[S_i]}{1-\rho_i} + E[S_i]$, is the expected delay for node *i* with $E[S_i]=1/\mu_i$ and $\rho_i=\lambda_i/\mu_i$.

i =1,…, N. The previous calculations of delay are shown in Figure 5-10.

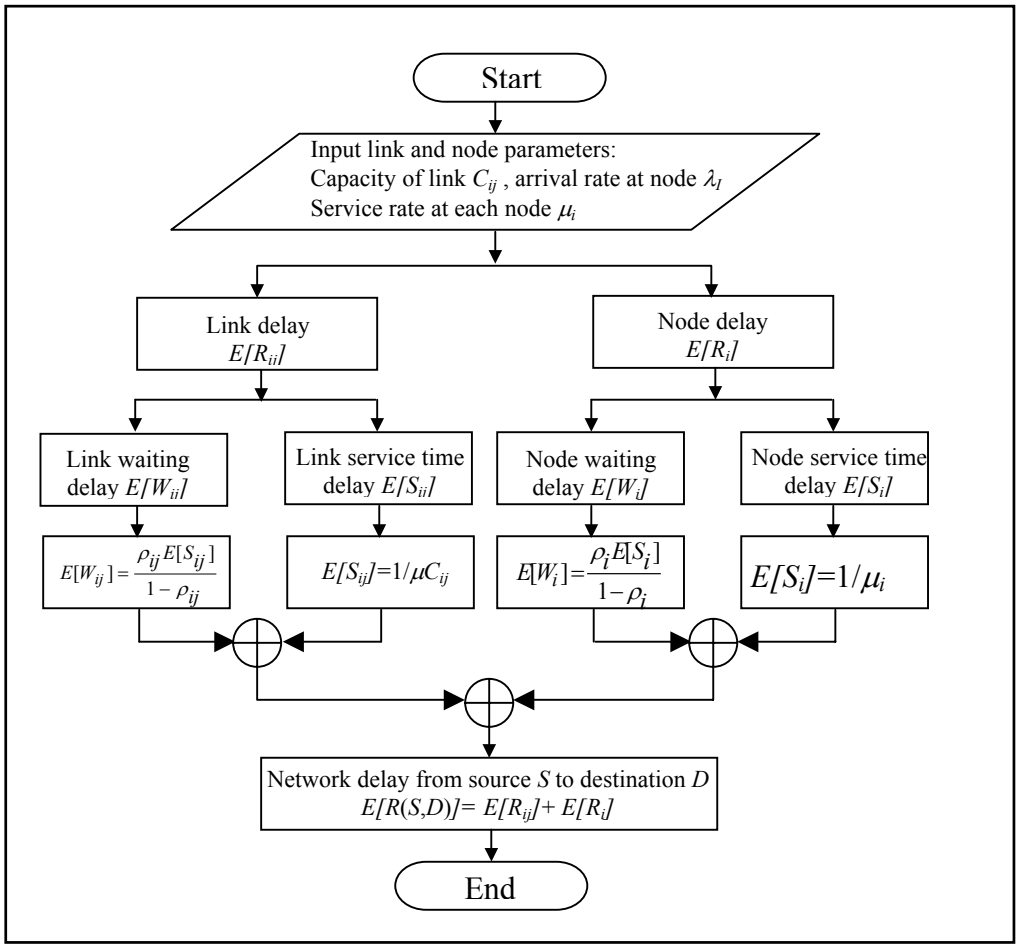


Figure 5-10. Calculation of end-to-end delay

5.4.3 Ant Algorithm

This section describes ant algorithm in detail. Firstly, we will explain the intelligent agents that will be used in the ant algorithm.

What has not been considered so far is how the probabilities in the routing tables are computed and updated. In the nature, ants lay pheromone and so they produce pheromone trails between the nest and a food source. On a computer, the pheromone has been replaced by *artificial stigmergy*, the probabilities in the routing tables. To compute and update the probabilities, intelligent agents are introduced to replace the ants. There exist two kinds of agents, the *forward agents* and the *backward agents*. All forward and backward agents have the same structure. The agents move inside the network by hopping at every time step from a node to the next node along the existing links. The agents communicate with each other in an indirect way by concurrently reading and writing the routing tables on their way.

The forward and backward agents receive percepts from the environment so they can do certain actions. Condition action rules are used to define what action should be chosen under which situation. The current situation is defined by the percept and the stored internal state. A description of both agent types is given in Table 5-2. The condition action rules of the agents are defined in Table 5-3.

Table 5-2. Description of forward and backward agents

Agent Type	Percepts	Actions	Goals	Environment
Forward agent	- ID of current node - IDs of the neighbor nodes and links that lead to them - routing table of current node	-Update memory -Remove cycle from memory -Determine and go to next node -Transform to backward agent	go to destination node, store the route and the travel times	network consisting of nodes and links connecting the nodes, only local information available at every node (routing tables)
Backward agent	- ID of current node - IDs of the neighbor nodes and links that lead to them	- update routing table - go to next node - kill the agent	go back to the source node along the stored route of the forward agent and update the routing tables	as forward agent

Table 5-3. Condition action rules of the agents

Agent Type	Condition action rules
Forward agent	IF current node already exists in memory THEN remove cycle from memory IF current node ≠ destination node THEN update memory AND go to next node IF current node = destination node THEN update memory AND transform to backward agent
Backward agent	IF current node ≠ source node THEN update routing table AND go to next node IF current node = source node THEN update routing table AND kill the agent

The internal state of the agents is a kind of memory which stores a list of (k,t_k) pairs. Every such pair represents a node that has been visited by the forward agent. k is the identifier of the visited node and t_k is the time a packet takes to travel the link from the last visited node to this node under the current traffic situation.

For further illustration of the ant algorithm function for individual ants as well as individual nodes, Figure 5-11 depicts the algorithm flow for each ant, while Figure 5-12 depicts the algorithm flow at each node.

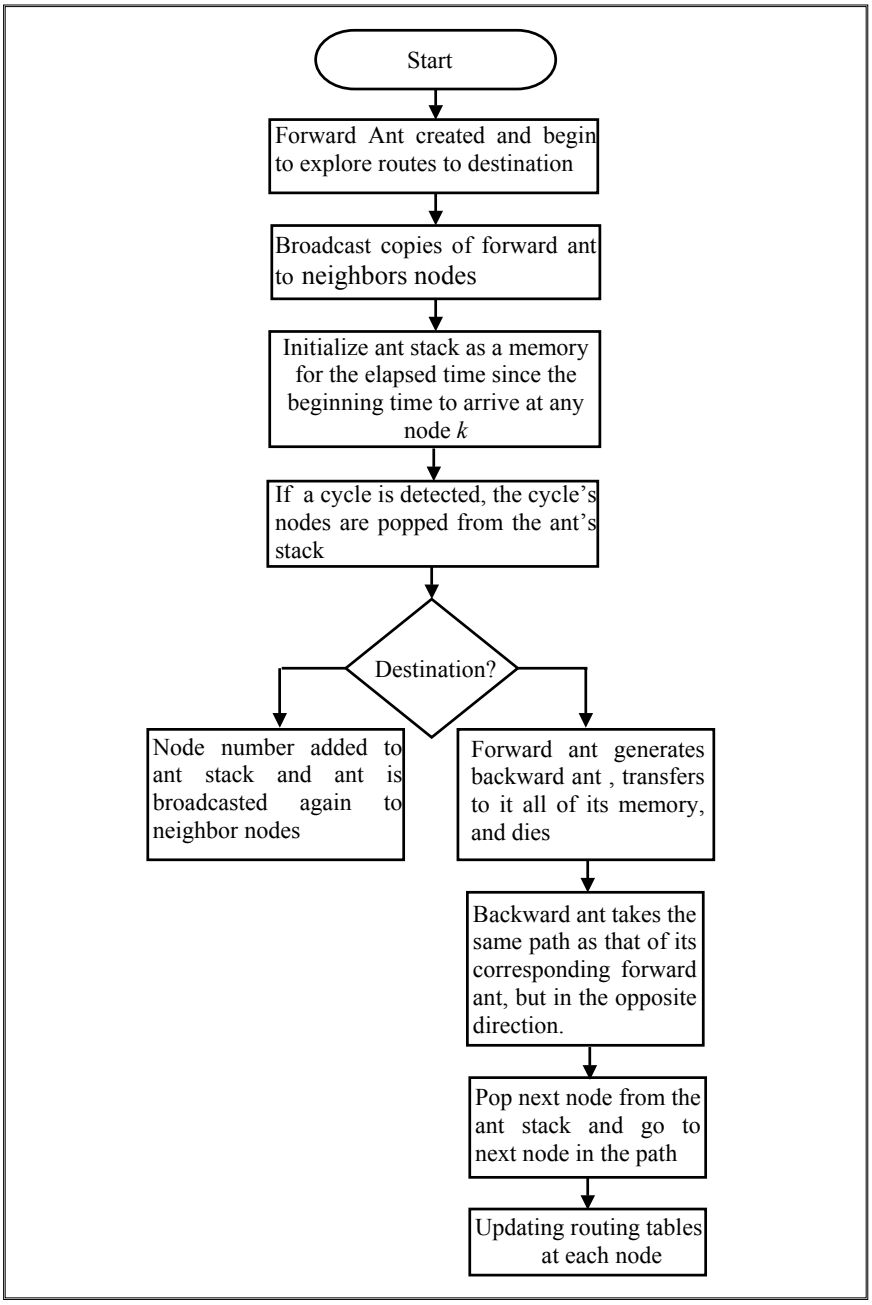


Figure 5-11. The algorithm for each ant

Using the description of the agents from Table 5-2 and the condition action rules from Table 5-3, the algorithm can be explained now.

5.4.3.1 Forward Ants

Each node s periodically sends a forward ant (forward agent) $F_{s\rightarrow d}$ to a randomly chosen destination node d throughout the network. The task of the forward ant is to discover a feasible, low-cost path to the destination and to gather useful information on its trip.

Every ant will represent a package sent from *s* to *d* through the network and each forward ant packet contains the following fields; *id number* of source node, *id number* of destination node and *stack memory*.

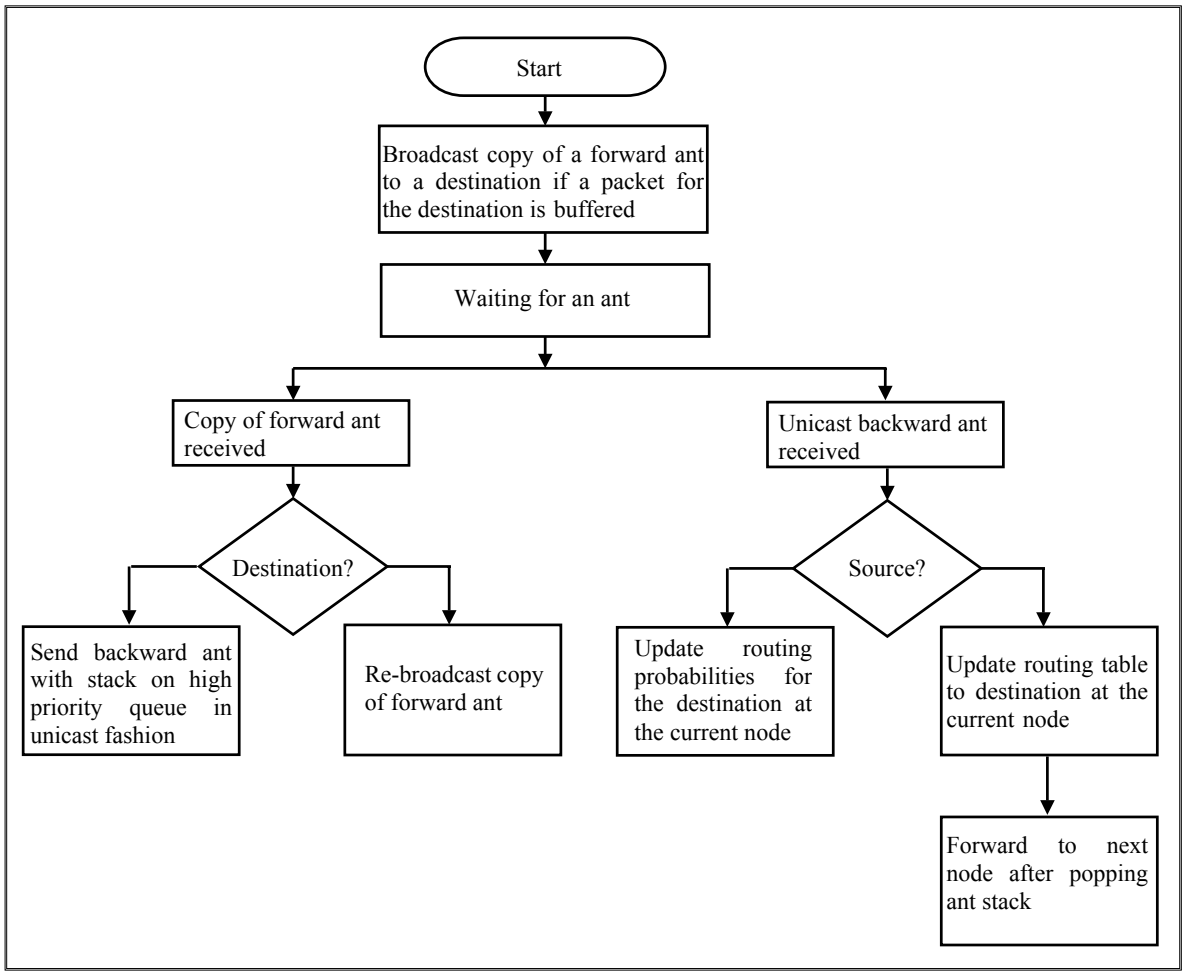


Figure 5-12. The algorithm at each node

At every visited node *k* on the way to the destination node, a forward ant does the following:

- The forward ant checks its stack memory whether node *k* has already been visited before or not. If it has been visited, there will be a cycle in the ant's path and this cycle will be deleted from the memory. The cycle's nodes are popped from the ant's stack and all the memory about them is destroyed as indicated in Figure (5-13b).
- The forward ant updates its memory by adding a new (k,t_k) pair to the memory as indicated in Figure (5-13a).
- If node *k* is not the destination node then the forward ant determines the next node to go to by using the probabilities in the row of the routing table of node *k* which represents the destination node.

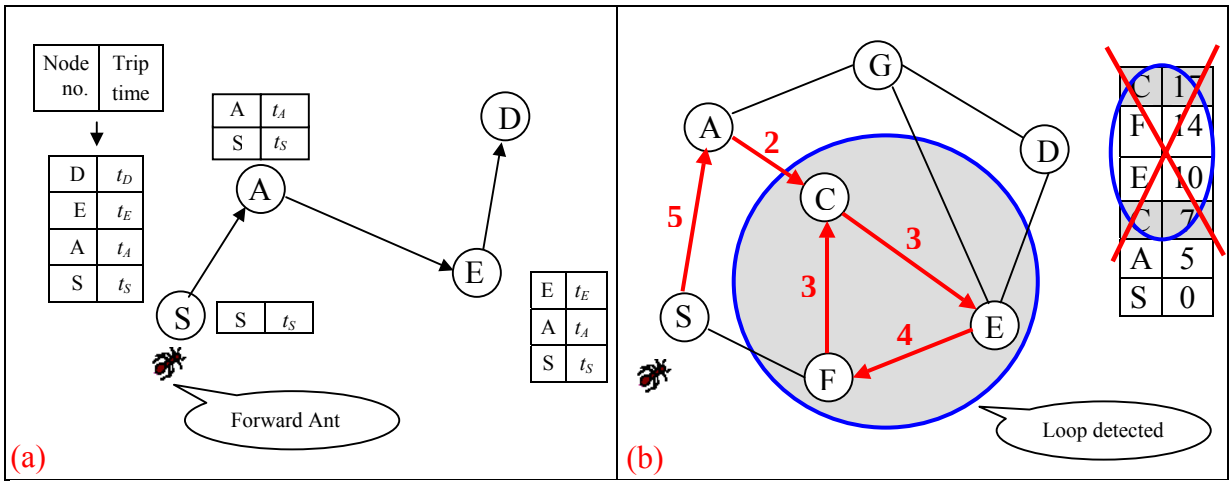


Figure 5-13. Trip time memory stack for one ant and cycle detection

In the first iteration of our algorithm, all the links from a node *k* to its neighbor nodes have equal probabilities, so the forward ant goes randomly to any one of its neighbor nodes.

In the successive iterations, the selection of the next hop is done according to *roulette wheel selection rule*. This simplest selection scheme also called *stochastic sampling with replacement*. This is a stochastic algorithm and involves the following technique:

The individuals (nodes of each routing table) are mapped to contiguous segments of a line, such that each node's segment is equal in size to its fitness (probability of routing). A random number between 0 and 1 is generated for every link to a neighbor node according to the magnitude of the probabilities and the node whose segment spans the random number is selected. This technique is analogous to a roulette wheel with each slice proportional in size to the fitness.

Table 5-4 shows an example of the next hop selection process from the routing table at a mobile node E. We remark that, the mobile node C has the highest probability to be chosen as a next hop node and occupies the largest interval, whereas node G is the least probability node that has the smallest interval on the line as shown in Figure 5-14.

Table 5-4. Selection probability at mobile node E

Next hop	D	G	F	C
Probability of routing	0.16	0.14	0.25	0.45
Cumulative probability	0.16	0.30	0.55	1.00

Figure 5-14 shows the selection process of the new nodes. For example, if the generated random number is 0.73, then node C will be selected as a next hop.

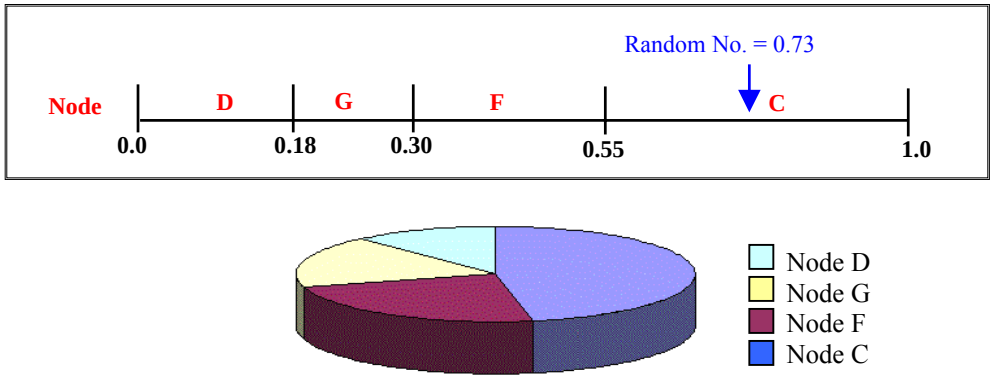


Figure 5-14. Roulette wheel selection

The individuals are selected according to their probabilities. The higher the probability, the higher the probability of being selected. The roulette can be described with the following algorithm:

[Sum] Calculate sum *S* of all neighbor nodes probabilities for a node *k*.

[Select] Generate random number *r* from the interval (0,*S*).

[Loop] Go through the neighbor nodes and sum the probabilities from 0. When the value of this sum exceeds the value of *r*, the procedure stops and the current node becomes the selected one.

Now, we go back to our discussion about the ant algorithm. The node where the ant has just come from is filtered out to avoid that the ant directly goes back to that node. With the generated probabilities, the next link is randomly selected. The forward ant goes to the next node along that link.

- If node *k* is the destination node then the forward ant transforms to a backward ant $B_{d \to s}$.

Now, all forward ants arrived at their destination node. Each ant has a stack $S_{s \to d}\ (k)$ of the virtual elapsed time for each ant trip. The stack of the forward ant is a dynamically growing data structure that contains the *id number* of the nodes that the forward ant has traversed as well as the elapsed time between its starting from *s* to its arriving to node *k*. For example, the stack for one of the ant trips from source *s* to destination node *d* shown in Figure 5-13a. The trip time to reach the desired neighbor is computed using the formula:

$$d_{ij} + (q_{ij} + S_a) / B_{ij}$$

Where d_{ij} is the link's propagation delay (distance/signal propagation speed) between two mobile nodes *i* and *j*. Note that this value can be neglected because the distance value is very small in comparison to the value of signal propagation speed.

q_{ij} is the number of bits of the data packets waiting in the queue between nodes *i* and *j*. This value is calculated by using this equation $q_{ij} = \frac{\rho_{ij}^2}{1-\rho_{ij}}$, where ρ_{ij} is the utilization of the link between two nodes *i* and *j*. The utilization can be calculated using this equation $\rho_{ij} = \frac{\lambda_j \times P_{ij}}{\mu_j}$, where λ_j and μ_j are the arrival rate and service rate at node *j* respectively. P_{ij} is the probability of routing from node *i* to *j*. S_a is the size of the ant packet, and B_{ij} is the bandwidth of the link between two nodes *i* and *j*. Bandwidth is the amount of data that can be transmitted in a fixed amount of time. It is expressed in bits (of data) per second (bps).

These estimates are computed at each node *k*, using a local statistical model L^{ij}_k capturing the depletion dynamics of each of the local links *ij* between any two nodes *i* and *j*.

Forward ants are routed on normal priority queues, that means, they use the same queues as normal data packets. As such, forward ants face the same network conditions (queuing and processing delays, network congestion) as data packets. Forward ants therefore contain information regarding the route that they have traversed.

We can summarize the work of the forward ants as follows; each forward ant selects the next hop node using the information stored in the routing table. The route is selected, following a random scheme, proportionally to the goodness (probability) of each path and to the local queues status. In case the selected link is not currently available, the forward ant waits its turn in the low-priority queue of the data packets, where it is served on the basis of a FIFO policy.

5.4.3.2 Backward Ants

The backward ant (backward agent) inherits the memory from the forward ant. The task of the backward ant is to go back to the source node *s* along the same path as the forward agent but in the opposite direction and to update the routing tables on this path.

At every visited node *k* on the way back to the source node a backward agent does the following:

- The backward agent updates the routing table of node *k* by using the travel times stored in its memory. The update process is explained in the next section.
- If node *k* is not the source node then the backward agent uses its memory to determine the next link of the path back to the source node. The backward agent goes to the next node along that link.
- If node *k* is the source node then the backward agent is killed.

Let's summarize the indirect communication of the intelligent agents via artificial stigmergy: The forward agents use the routing tables (artificial stigmergy) of the nodes to find their way to the destination node. They only have reading access to the routing tables and can not change the probabilities. The backward agents use their memory to find the way back to the source node. On their way back, they update the routing tables by changing some of the probabilities. By that, the probabilities in the routing tables change very fast, possibly several times a second. The next forward agents that visit the nodes will find other values in the routing tables and therefore other conditions under which they react. By this indirect communication between the forward ants and the backward ants, an emergent behavior of the quite simple ants arises like in ant colonies that leads to optimize routes.

5.4.3.3 Updating Routing Tables

Updating in routing tables at each node will be done by backward ants using ants trip times. As shown in Figure 5-15, at every visited node k on the way back to the source node, a backward ant updates some of the probabilities in the routing table of node k by using the travel information in its memory which was collected by the forward ant.

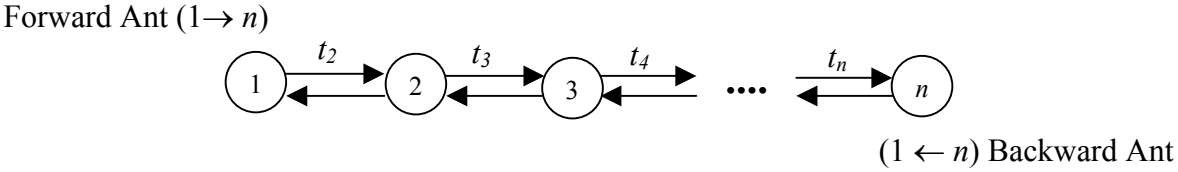


Figure 5-15. Forward and backward ant's path

The backward ant retraces the path of the forward ant by popping the stack, making modifications in the routing tables at each intermediate node according to the following learning rules:

IF (node was in the path of the ant)

THEN $p^{new}(i) = p^{old}(i) + r\,[1 - p^{old}(i)]$ (2)

ELSE $p^{new}(i) = p^{old}(i) - r\,p^{old}(i)$ (3)

where $r \in (0,1]$ is the *reinforcement factor* which is central to express path quality. The reinforcement factor should be a factor of trip time of the node neighborhood. This factor is given by the relationship $r=t_1/t_2$ where t_1 is the minimum trip time of all the forward ants, and t_2 is the trip time of the current forward ant from a node to the destination node.

$p^{new}(i)$ is the new probability and $p^{old}(i)$ is the old probability for a node i in the routing table.

Note that equation (2) always increases the probability $p^{new}(i)$, what means that every update is a positive feedback. This resembles the pheromone of the ants where every dropped pheromone increases the strength of a pheromone trail but not weakens it.

The reinforcement factor ensures that the probability increase is inversely proportional to the travel time of the forward ant. The higher the time t_2, the lower the probability and the lower the time t_2, the higher the probability. By that, good paths receive a strong update while bad paths receive a low update only.

For a given value of r, the absolute and relative increase of $p^{new}(i)$ is much larger for small values of $p^{old}(i)$ than for large values of $p^{old}(i)$. By that weighted change, low probabilities go up very fast to adapt to the new traffic situation, whereas probabilities which are already high are increased only a little bit.

Whenever a probability in the routing table is increased by equation (2), the other entries in the same row must be decreased by equation (3). This is necessary to ensure that the sum of the probabilities in each row remains 1.

Equation (3) is used to normalize the values. This means that a probability can only decrease if another probability in the same row increases. A probability can approach zero if other probabilities in the same row of the routing table increase much more often or much stronger. This would mean that a forward ant hardly takes the route associated with this low probability, what means that this probability receives nearly no further updates. But indeed, this route can become the fastest route in time under changed traffic conditions. After that, the probabilities per destination are normalized again for all the nodes, so that the sum of the probabilities is 1, i.e. $\sum_i P_i = 1$ which ensures that every link is used by forward agents from time to time and so new routes are explored regularly.

The value of a probability in a routing table finally depends on two facts: the trip times experienced by the forward ants that use the route associated with the probability and the frequency of the updates (the number of ants that use the route). For example, if the trip times are very low, the reinforcement factor r is very high and so the probability is strongly increased with every update. So the probability will be very high even if the frequency of the update is low. But on the other hand, also high trip times, which mean a low reinforcement factor r, can lead to a high probability if the frequency of the updates is high. In that way, high frequent updates with low travel times lead to the highest probabilities.

What has to be mentioned is that the equations (2) and (3) to update the probabilities are very simple. They only take the old probability and the experienced trip time to compute the new probability. The adaptation to suddenly occurring new traffic situations is very slow.
Now after this step of updating routing tables, the probabilities in routing tables are altered a little bit and then, the delay is recalculated using equation (1). If the new delay is better, it definitely becomes the starting point for the next iteration and each forward ant will move by applying *roulette wheel selection* on the new probabilities after updating. If the delay is worse, each forward ant will move by applying r*oulette wheel selection* on the old probabilities before updating. This has the effect that nearly every new solution is adopted, while over time it becomes more and more likely that only better solutions are accepted.
This comparison between the delay of each successive iteration repeats until the ant timer T_{ant} finishes, then if the simulation timer T_{Simul} hasn't finished yet, a new topology of the network will be established according to the used mobility model and repeats again the same procedure until the simulation timer expires.

5.5 Performance Evaluation

In this section, we evaluate the End-to-End delay *E[V]* for the packets through the MANET using the framework described in the preceeding sections.

5.5.1 Simulation Model

The simulation scenario consists of a number of nodes that are initially placed randomly and constantly moving in a simulation rectangular area 1000×800 m^2 according to RWM or BSAM models.
During the simulation, nodes are free to move anywhere within this area. Each node travels toward a random spot, then takes a rest period of time in second. After the rest period, the node travels toward another randomly selected spot. This process repeats throughout the simulation, causing continuous changes in the topology of the underlying network, followed by a simulation of the ant behavior yielding an improvement of the routing tables, which is evaluated by Kleinrock´s delay analysis. Finally, we get the minimum delay from source to destination node. The simulation program has been executed on standard 350 Mhz PC using Visual Basic 6.0. It needs few seconds of CPU time for a single simulation run.
The model parameters that have been used in the following experiments are summarized in Table 5-5.

Table 5-5. Simulation parameters

Parameter	Value
Number of nodes	10, 20, …, 150
Arrival rate	150 Kbps
Transmission range	250 m
Velocity / Direction	10 m/sec & 45 degree
Packet size	64 byte
Pause time (Ant time)	10, 20, …, 100 sec.
Simulation time	180 sec.
Link bandwidth	1 Mbps
Simulation area	1000 m × 800 m
Mobility model	RWM & BSAM models
Routing protocol	Ant algorithm

We will use a simple scenario for evaluating our ad hoc routing algorithm with the following assumptions:

- 2-d rectangular area
- No obstacles
- Bi-directional links
- Fixed number of nodes
- Nodes operate for whole simulation
- Random waypoint mobility (RWM) model: Nodes pause for a random amount of time before picking uniformly distributed points from the simulation area. Nodes then move to new points in the simulation area.
- Boundless simulation area mobility (BSAM) model: There exists a relationship between the previous and the current movement direction θ and velocity v. Both direction and velocity are deterministic through the whole simulation time.

5.5.2 Experimental Results

The first experiment shows the relation between increasing the number of nodes and the End-to-End delay in case of using different numbers of nodes (10, 20, …, 140, 150) while ant time is constant at 30 sec.

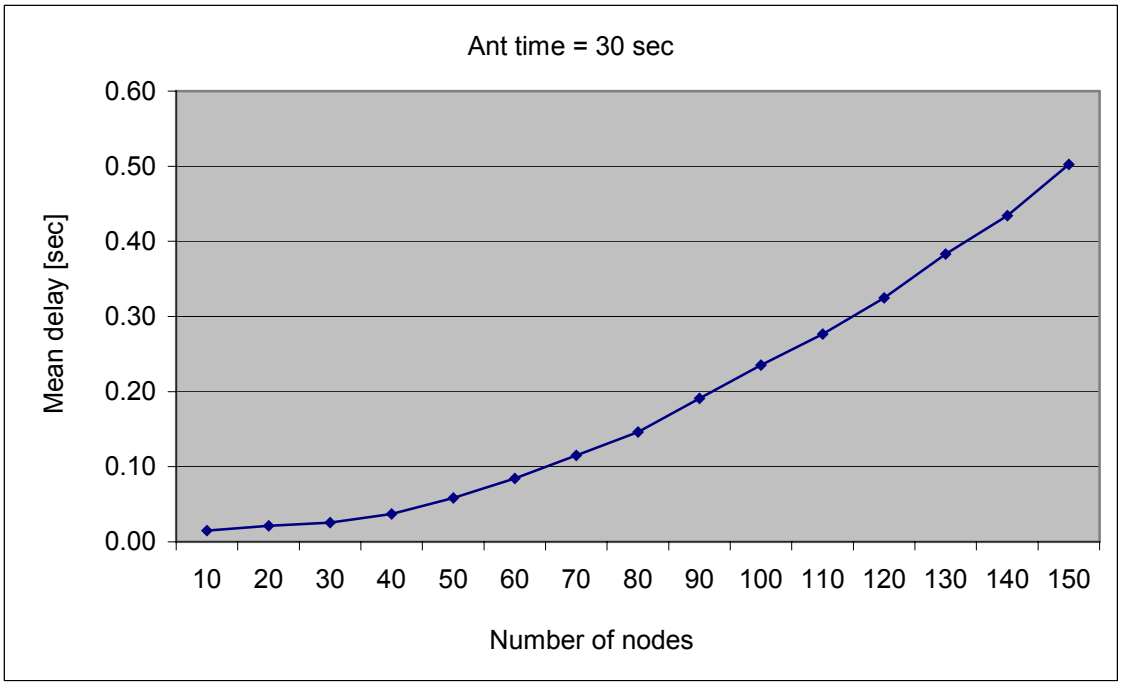


Figure 5-16. Number of nodes vs. mean delay

Figure 5-16 shows that increasing the number of nodes results in an increase in the delay, because each hop can contribute a substantial amount of delay in forwarding traffic. Furthermore, the more nodes, the more congestion and the longer it takes to discover routes. The second experiment investigates the relation between increasing the ant time and the End-to-End delay in case of using different pause times (10,20, …,100 seconds) and 60 nodes.

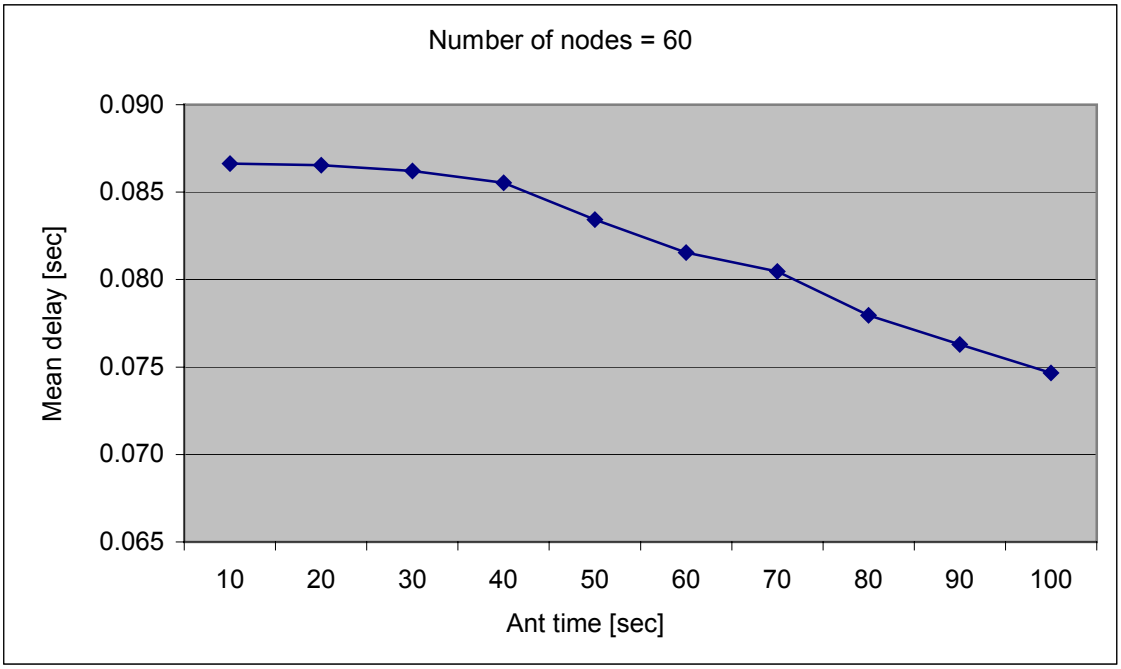


Figure 5-17. Ant time vs. mean delay

Figure 5-17 shows that increasing the pause time leads to a decrease in the delay, because the ant algorithm performs more iterations that help to approach the minimum delay.

The third experiment is done for a scenario with 60 nodes distributed randomly and an ant time of 60 seconds. The whole simulation time is 180 seconds and we observe the packet delays over this time period.

Figure 5-18 shows the mean delay for the first topology. Figures 5-19 and 5-20 show the mean delay for the topologies, which has been established from the preceding topology by applying the mobility model. These figures do display 95% confidence intervals.

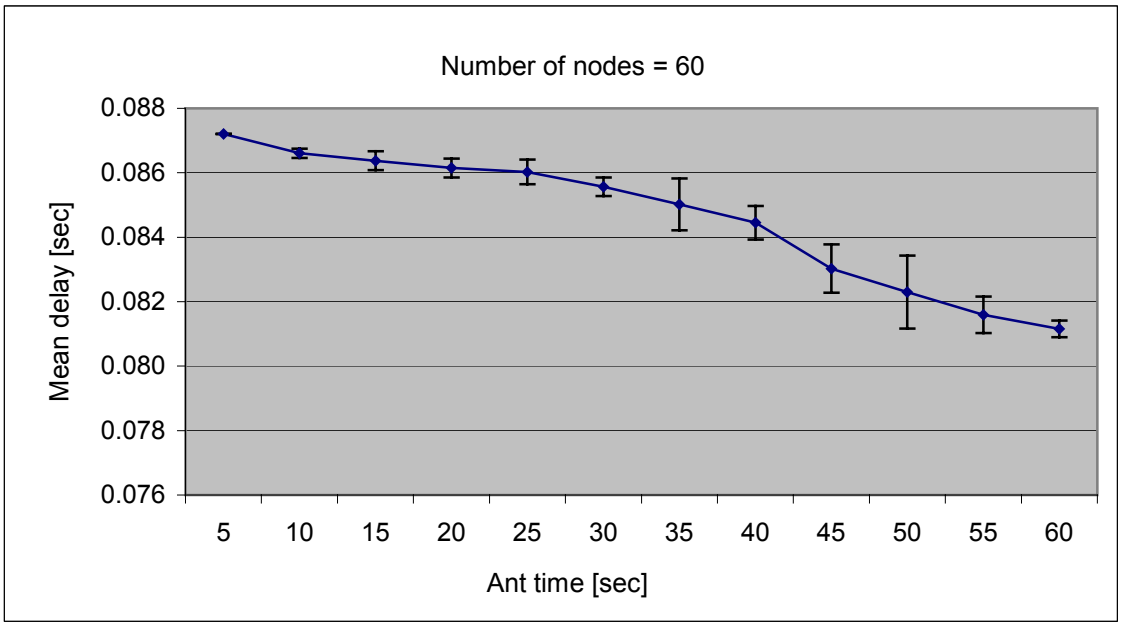


Figure 5-18. Mean delay through first topology

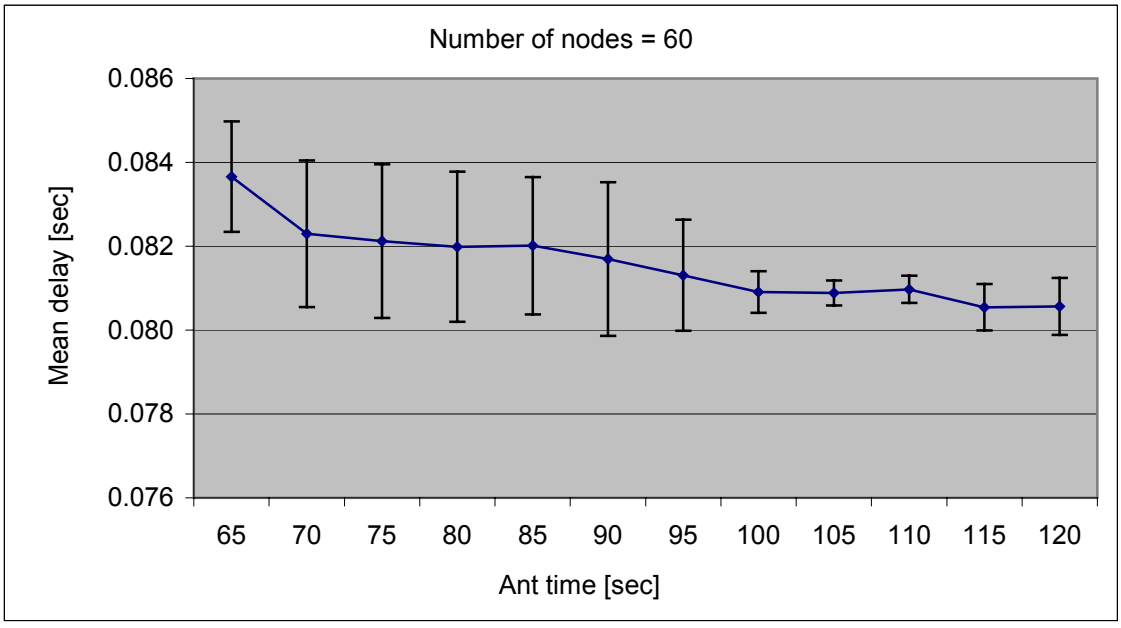


Figure 5-19. Mean delay through second topology

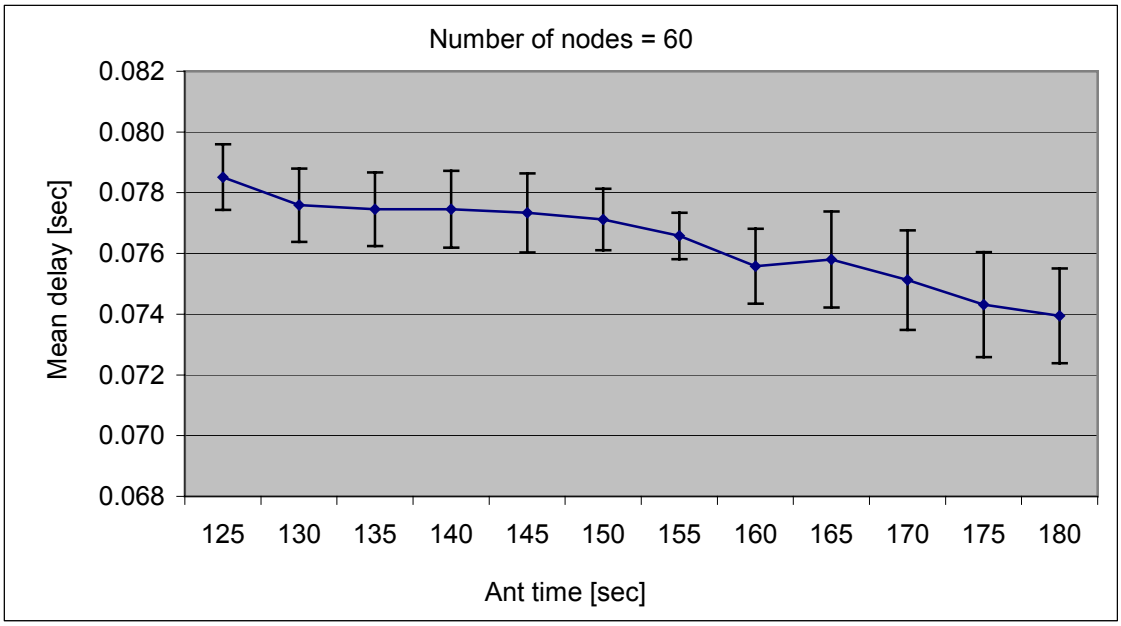


Figure 5-20. Mean delay through third topology

Figure 5-21 summarizes the tracing of the delay over the whole simulation time of 180 seconds for three different topologies. We note from the figure that there is no significant effect on the delay between RWM and BSAM models, but the performance can vary significantly with other different mobility models as stated in [Camp, 2002].

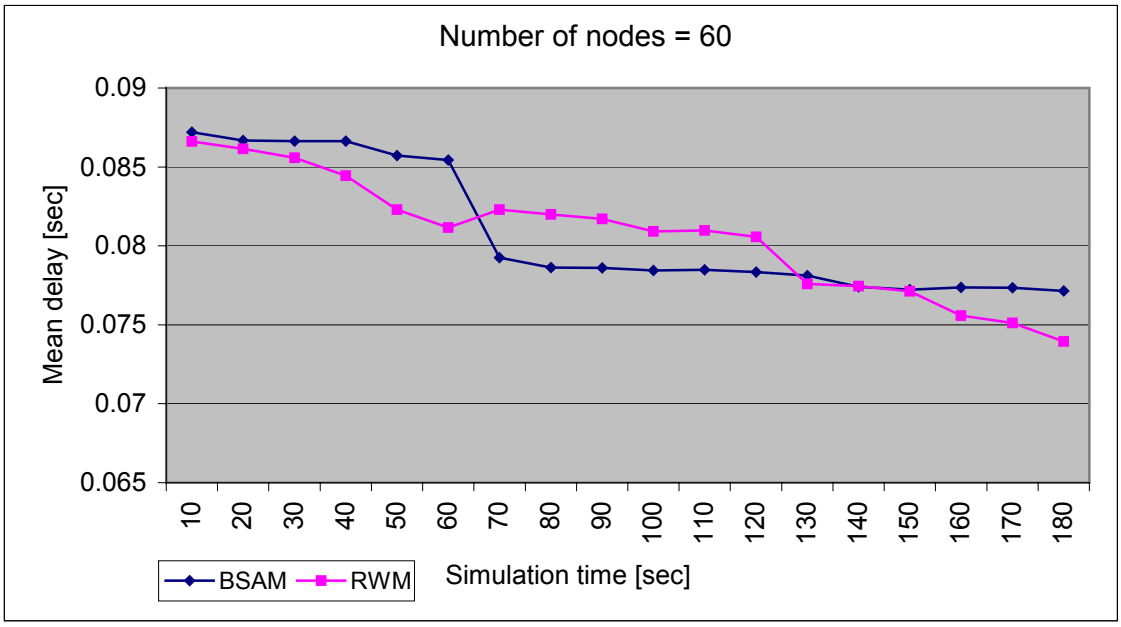


Figure 5-21. Packet delay over 180 sec. simulation time

After a change in the topology, we observe a degradation of the performance (i.e. an increase of the delay) followed by a performance improvement due to the work of the ant algorithm which converges step-by-step toward the minimum possible delay.

The last experiment is done to remark the relation between the number of nodes and loop detection time. Figure 5-22 indicates that the increasing in the number of nodes will cause increasing in the mean time for loop detection [Tarek, 2004].

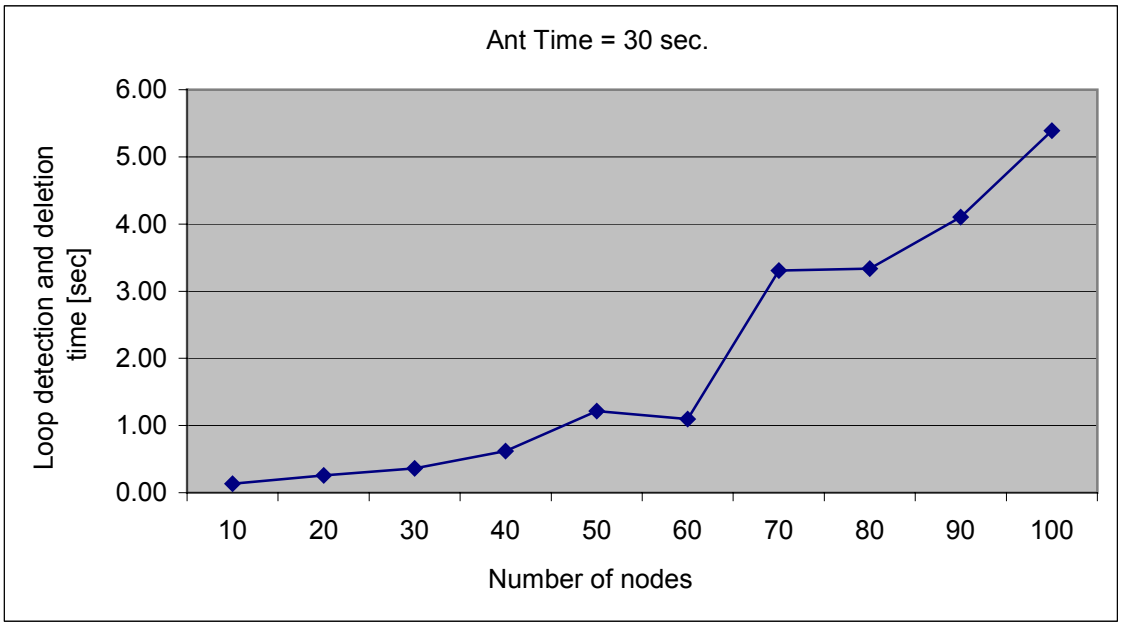


Figure 5-22. Number of nodes vs. loop detection and deletion time

The higher the node density, the higher the possibility of loops formation, which in turn causes the time of loop detection and deletion to increase.

Bibliography

[Broch et al., 1998] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 85–97, 1998.

[Camp et al., 2002] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research, In *Wireless Communication and Mobile Computing (WCMC)*, Vol. 2, No. 5, pages 483-502, 2002.

[Chiang and Gerla, 1998] C. Chiang and M. Gerla. On-Demand Multicast in Mobile Wireless Networks. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 1998.

[Davies, 2000] V. Davies. *Evaluating Mobility Models within an Ad Hoc Network*. Master's Thesis, Colorado School of Mines, 2000.

[Garcia and Spohn, 1999] J. J. Garcia and M. Spohn. Source-Tree Routing in Wireless Networks. In *Proceedings of the 7th International Conference on Network Protocols (ICNP)*, 1999.

[Haverkort, 1998] B. Haverkort. *Performance of Computer Communication Systems, A Model-Based Approach*, John Wiley & Sons, Ltd., 1998.

[Johansson et al., 1999] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Routing Protocols for Mobile Ad-Hoc Networks - A Comparative Performance Analysis. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, pages 195–206, 1999.

[Johnson and Maltz, 1996] D. Johnson and D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In T. Imelinsky and H. Korth, editors, *Mobile Computing*, Kluwer Academic Publishers, pages 153–181, 1996.

[Kleinrock, 1964] L. Kleinrock. *Communication Nets: Stochastic Message Flow and Delays*. McGraw-Hill, New York, 1964.

[Särelä, 2004] M. Särelä. Measuring the Effects of Mobility on Reactive Ad Hoc Routing Protocols. *Helsinki University of Technology Laboratory for Theoretical Computer Science, Research Reports 91*, 2004.

[Tarek and Mueller-Clostermann, 2004] A. Tarek and B. Mueller-Clostermann. Modelling and Simulation of a Routing Protocol for Ad Hoc Networks Combining Queuing Network Analysis and Ant Colony Algorithms. *11th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA'04),* Magdeburg, Germany, 2004.

[Tarek, 2004] A. Tarek. Ant Colony Optimization for Routing in Mobile Ad Hoc Networks. *5th International Conference on Recent Advances in Soft Computing (RASC'04)*, Nottingham, United Kingdom, 2004.

[Wong, 1978] J. W. WONG. Queueing Network Modeling of Computer Communication Networks, *Computing Surveys*, vol. 10, No. 3, 1978.

Chapter
6

COMPARISON STUDY

6.1 Introduction

On-demand routing protocols in ad hoc networks are called reactive routing protocols which means, routes are determined only when needed. In contrast to proactive (table-driven) routing protocols, all up-to-date routes are not maintained at every node; instead the routes are created as and when required. When a source wants to send a packet to a destination, it invokes the route discovery mechanisms to find the path to the destination. The route remains valid as long as the destination is reachable or until the route is no longer needed. The Ad hoc On-demand Distance Vector (AODV) routing protocol is one of several published routing protocols for mobile ad hoc networks [Perkins and Royer, 1999]. Wireless ad hoc routing protocols such as AODV are currently an area of much research among the networking community. Thus, tools for simulating these protocols are very important. In this chapter, we will discuss in detail AODV routing protocol. Our aim of this chapter is to make a comparison study between AODV and our proposed ant algorithm.

6.2 Ad hoc On-Demand Distance Vector Routing (AODV)

The Ad hoc On-demand Distance Vector (AODV) routing protocol is a reactive protocol designed for wireless ad hoc networks [Perkins, 2001]. It combines the use of destination sequence numbers in Destination Sequenced Distance Vector (DSDV) routing with the on-demand route discovery technique in Dynamic Source Routing (DSR) protocols to formulate a loop-free, on-demand, single path, distance vector protocol. Unlike DSR, which uses source routing, AODV is based on hop-by-hop routing approach. AODV is designed to improve upon the performance characteristics of DSDV in the creation and maintenance of routes. The primary objectives of the AODV protocol are:

1. To broadcast discovery packets only when necessary,
2. To distinguish between local connectivity management (neighborhood detection) and general topology maintenance,
3. To disseminate information about changes in local connectivity to those neighboring mobile nodes which are likely to need the information.

In AODV, each node maintains two separate counters:

1. *Sequence Number*, a monotonically increasing counter used to maintain freshness information about the reverse route to the source.
2. *Broadcast-ID*, which is incremented whenever the source issues a new Route Request (*RREQ*) message.

Each node also maintains information about its reachable neighbors with bi-directional connectivity. Whenever a node (router) receives a request to send a message, it checks its *routing table* to see if a route exists. Each routing table entry consists of the following fields:

- Destination address
- Next hop address
- Destination sequence number
- Hop count

AODV protocol is composed of two mechanisms: *Route Discovery* and *Route Maintenance*.

6.2.1 AODV Route Discovery

When a node needs to determine a route to a destination node, it floods the network with a *Route Request (RREQ) message* as shown in Figure 6-1. If a route exists, the originating node broadcasts a RREQ message to its neighboring nodes, which broadcast the message to their neighbors and so on. Otherwise, it saves the message in a *message queue*, and then it initiates a route request to determine a route. Figure 6-2 shows a flowchart to illustrate this process. To prevent cycles, each node remembers recently forwarded route requests in a route request buffer (see next section).

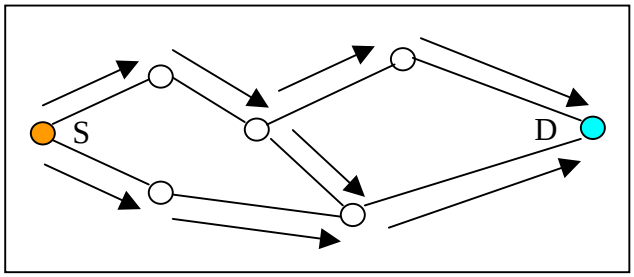


Figure 6-1. Propagation of the RREQ from source to destination node

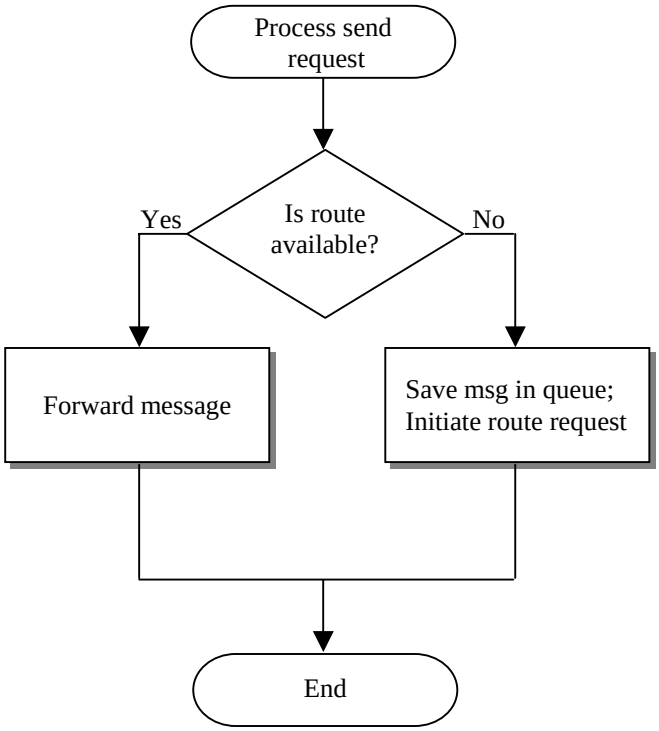


Figure 6-2. Flowchart for broadcasting a RREQ message

As these requests spread through the network, intermediate nodes store reverse routes back to the originating node. Since an intermediate node could have many reverse routes, it always picks the route with the smallest hop count as shown in Figure 6-3. When a node receiving the request either, it knows a "fresh enough" route to the destination (see section 6.2.3), or it is the destination itself, the node generates a *Route Reply (RREP) message* and sends this message along the reverse path back toward the originating node. As the RREP message passes through intermediate nodes, these nodes update their routing tables, so that in the future, messages can be routed through these nodes to the destination.

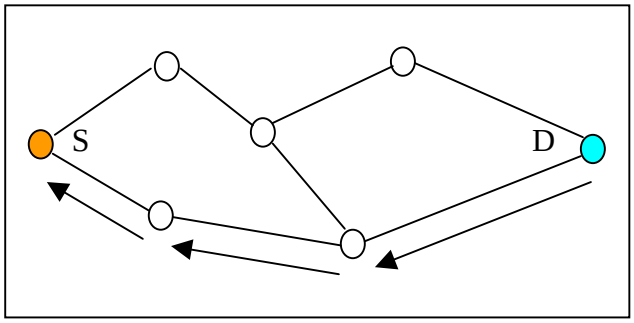


Figure 6-3. Propagation of the RREQ from destination to source node

Notice that it is possible for the RREQ originator to receive a RREP message from more than one node. In this case, the RREQ originator will update its routing table with the most

"recent" routing information; that means, it uses the route with the greatest destination sequence number (see section 6.2.3).

6.2.2 The Route Request Buffer

In the flooding protocol described above, when a node originates or forwards a route request message to its neighbors, the node will likely receive the same route request message back from its neighbors. To prevent nodes from resending the same RREQs (causing infinite cycles), each node maintains a *route request buffer*, which contains a list of recently broadcasted route requests. Before forwarding a RREQ message, a node always checks the buffer to make sure it has not already forwarded this request before.

RREQ messages are also stored in the buffer by a node that originates a RREP message. The purpose of this process is to insure that a node does not send multiple RREPs for duplicate RREQs that may have arrived from different paths. The exception is if the node receives a RREQ with a better route (i.e. smaller hop count), a new RREP will be sent.

Each entry in the route request buffer consists of a pair of values: the address of the node that originated the request, and a route request identification number (RREQ id). The RREQ id uniquely identifies a request originated by a given node. Therefore, the pair uniquely identifies a request across all nodes in the network.

To prevent the route request buffers from growing indefinitely, each entry expires after a certain period of time, and then is removed. Furthermore, each node's buffer has a maximum size. If nodes are to be added beyond this maximum, then the oldest entries will be removed to make room.

6.2.3 Sequence Numbers

Each destination (node) maintains a monotonically increasing sequence number, which serves as a logical time at that node. Also, every route entry includes a destination sequence number, which indicates the "time" at the destination node when the route was created. The protocol uses sequence numbers to ensure that nodes only update routes with "newer" ones. Doing so, we also ensure loop freedom for all routes to a destination.

All RREQ messages include the originator's sequence number, and its (latest known) destination sequence number. Nodes receiving the RREQ add/update routes to the originator with the originator sequence number, assuming this new number is greater than that of any existing entry. If the node receives an identical RREQ message via another path, the

originator sequence numbers would be the same, so in this case, the node would pick the route with the smaller hop count.

If a node receiving the RREQ message has a route to the desired destination, then we use sequence numbers to determine whether this route is “fresh enough” to use as a reply to the route request. To do this, we check if this node’s destination sequence number is at least as great as the maximum destination sequence number of all nodes through which the RREQ message has passed. If this is the case, then we can roughly guess that this route is not terribly out-of-date, and we send a RREP back to the originator.

As with RREQ messages, RREP messages also include destination sequence numbers. This is so that nodes along the route path can update their routing table entries with the latest destination sequence number.

6.2.4 Link Monitoring and Route Maintenance

Each node keeps track of a *precursor list* and an *outgoing list*. A precursor list is a set of nodes that route through the given node. The outgoing list is the set of next hops that this node routes through. In networks where all routes are bi-directional, these lists are essentially the same.

Each node periodically sends HELLO messages to its precursors. A node decides to send a HELLO message to a given precursor only if no message has been sent to that precursor recently. Correspondingly, each node expects to periodically receive messages (not limited to HELLO messages) from each of its outgoing nodes. If a node has received no messages from some outgoing nodes for an extended period of time, then that node is presumed to be no longer reachable.

Whenever a node determines one of its next hops to be unreachable, it removes all affected route entries and generates a Route Error (RERR) message. This RERR message contains a list of all destinations that have become unreachable as a result of the broken link. The node sends the RERR to each of its precursors. These precursors update their routing tables and in turn forward the RERR to their precursors, and so on. To prevent RERR message loops, a node only forwards a RERR message if at least one route has been removed.

The following Figure 6-4 displays a flowchart which summarizes the action of an AODV node when processing an incoming message. HELLO messages are excluded from the diagram for brevity.

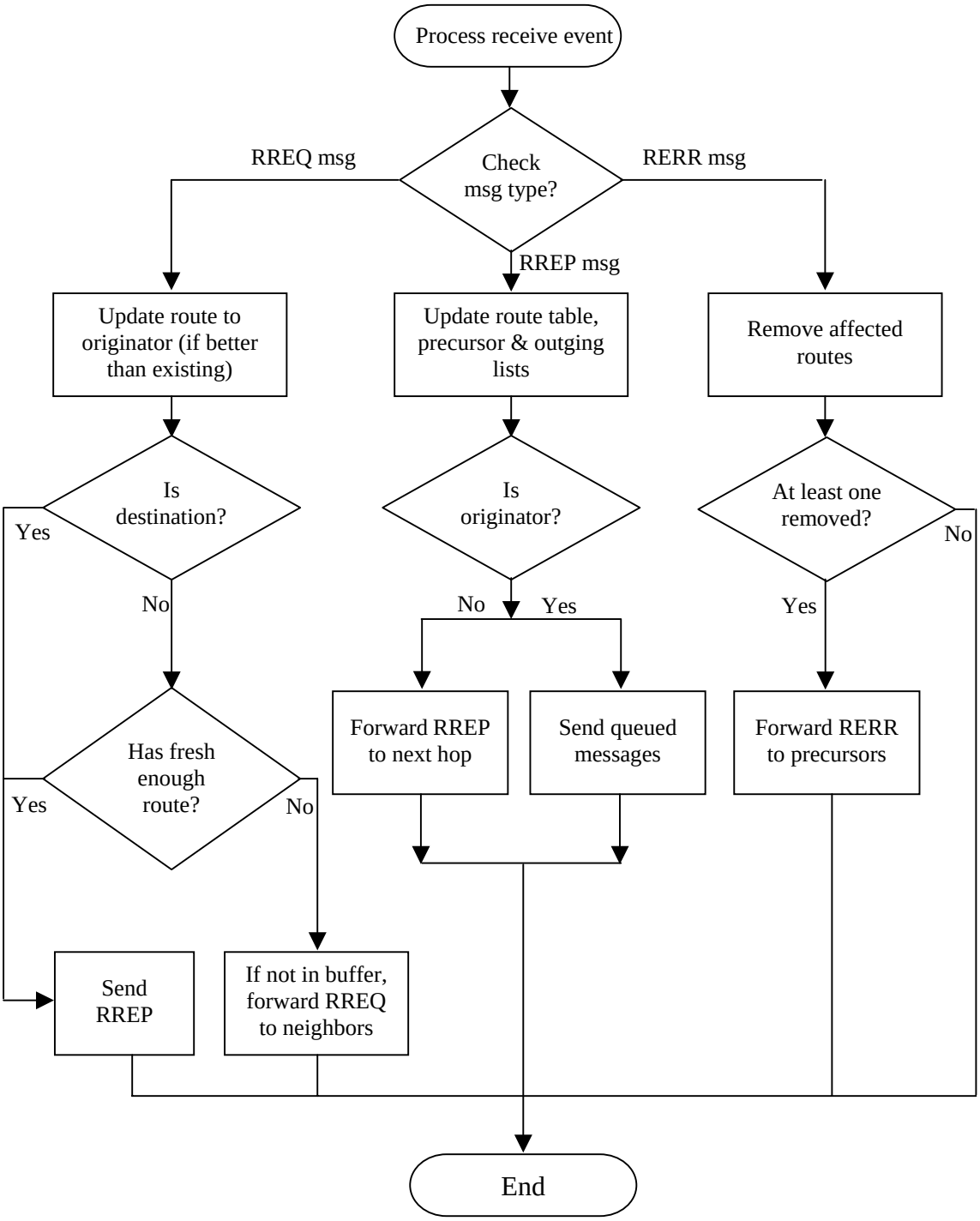


Figure 6-4. Flowchart for an AODV node when processing an incoming message

6.2.5 Summary of AODV Routing Protocol

AODV is a method of routing messages between mobile computers. It allows these mobile computers or nodes to pass messages through their neighbors to nodes with which they cannot directly communicate. AODV does this by discovering the routes along which messages can

be passed. AODV makes sure that these routes do not contain loops and tries to find the shortest route possible. AODV is also able to handle changes in routes and can create new routes if there is an error.

6.3 AODV Example

The following Figure 6-5 shows a set up of five nodes on a wireless network. The circles illustrate the range of communication for each node. Because of the limited range, each node can only communicate with the nodes next to it.

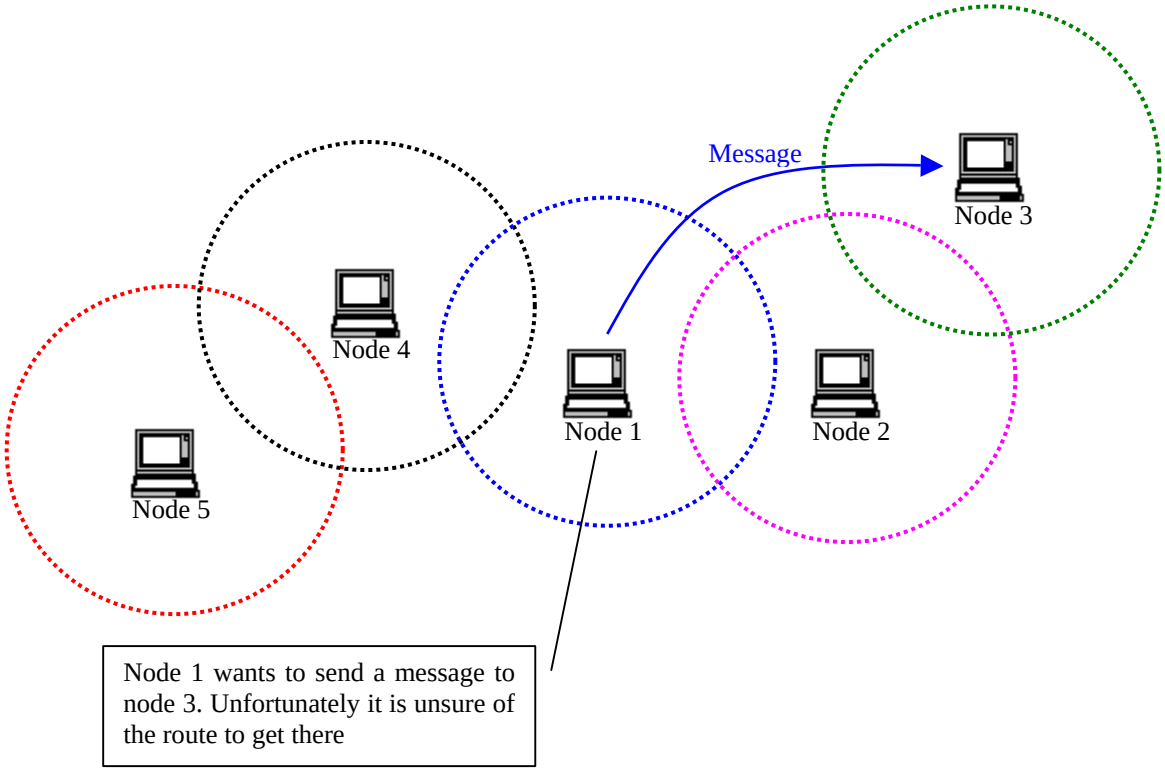


Figure 6-5. Setup a mobile ad hoc network consists of five nodes

Nodes which can communicate with the other nodes directly are considered to be neighbors. A node keeps track of its neighbors by listening for a HELLO message that each node broadcasts at set intervals. When one node needs to send a message to another node that is not its neighbor, it broadcasts a Route Request (RREQ) message. The RREQ message contains several key bits of information: the source, the destination, the lifespan of the message and a *Sequence Number* which serves as a unique ID.

Node 1 wishes to send a message to node 3. Node 1's neighbors are nodes 2 and 4. Since node 1 can not directly communicate with node 3, node 1 sends out a RREQ. The RREQ is heard by nodes 4 and 2 as shown in Figure 6-6.

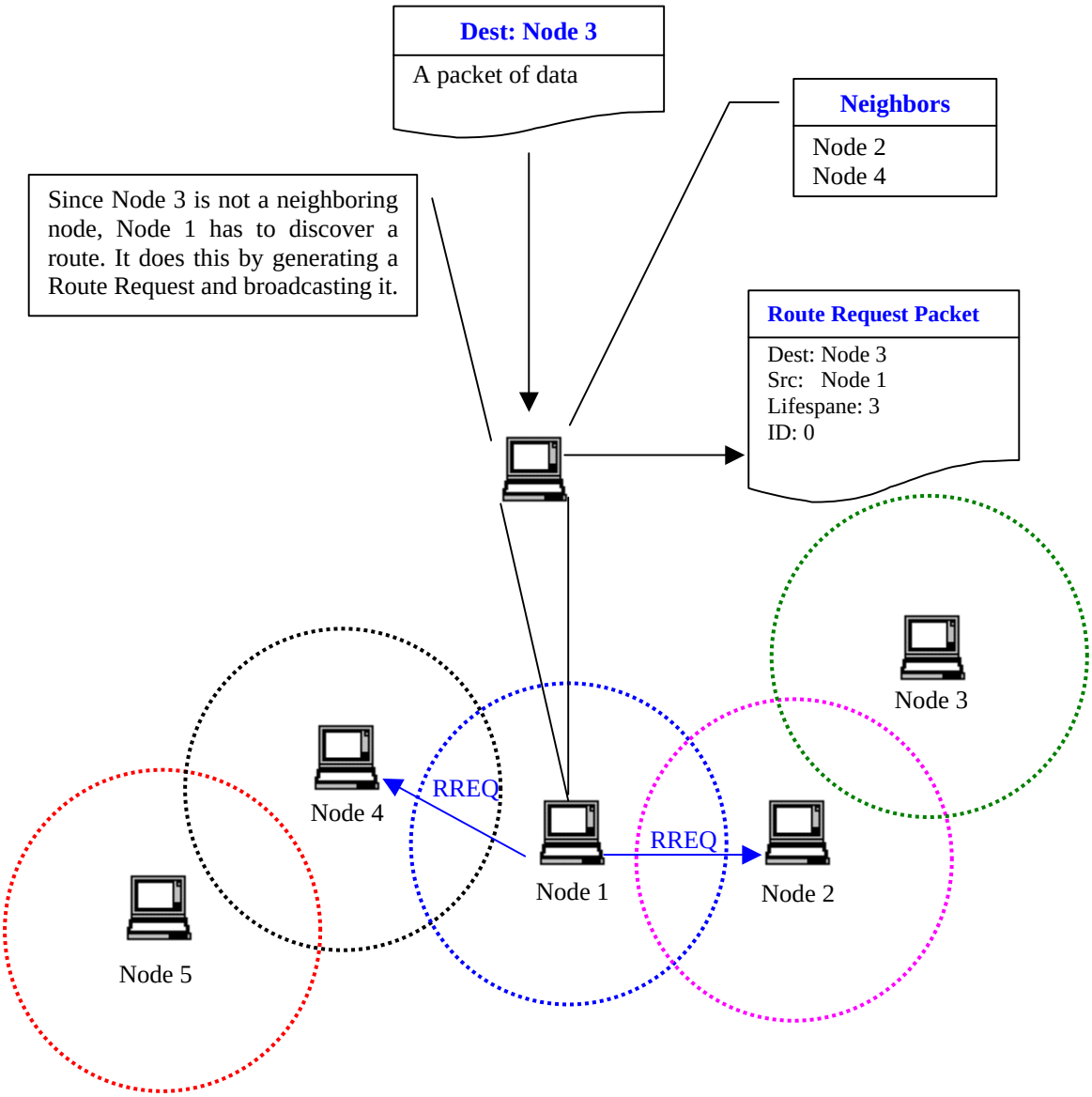


Figure 6-6. Node 1 sends out a RREQ to its neighbor nodes

When Node 1's neighbors receive the RREQ message they have two choices; if they know a route to the destination or if they are the destination they can send a Route Reply (RREP) message back to node 1, otherwise they will rebroadcast the RREQ to their set of neighbors. The message keeps getting rebroadcasted until its lifespan is up. If node 1 does not receive a reply in a set amount of time, it will rebroadcast the request but this time the RREQ message will have a longer lifespan and a new ID number. All of the nodes use the *sequence number* in the RREQ to insure that they do not rebroadcast the same RREQ. In Figure 6-7, node 2 has a route to node 3 and replies to the RREQ by sending out a RREP. Node 4 on the other hand does not have a route to node 3 so it rebroadcasts the RREQ.

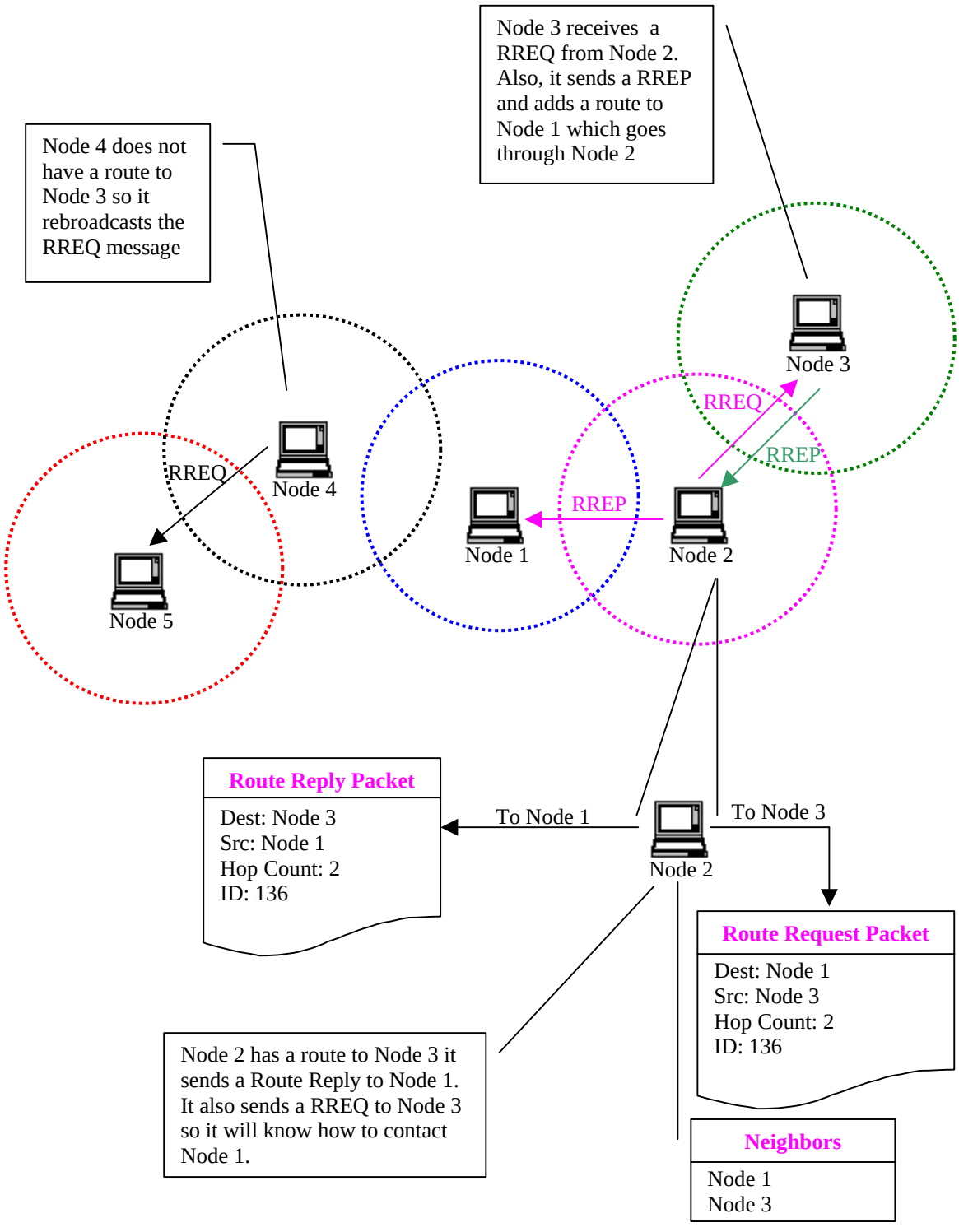


Figure 6-7. Node 2 has a route to node 3 and sends out a RREP

Sequence numbers serve as time stamps. They allow nodes to compare how “fresh” their information on other nodes is. Every time a node sends out any type of message it increases its own sequence number. Each node records the sequence number of all the other nodes it talks to. A higher sequence numbers signifies a fresher route. Thus it is possible for other

nodes to figure out which one has more accurate information. In Figure 6-8, node 1 is forwarding a RREP to node 4. It notices that the route in the RREP has a better sequence number than the route in its routing list. Node 1 then replaces the route it currently has with the route in the RREP.

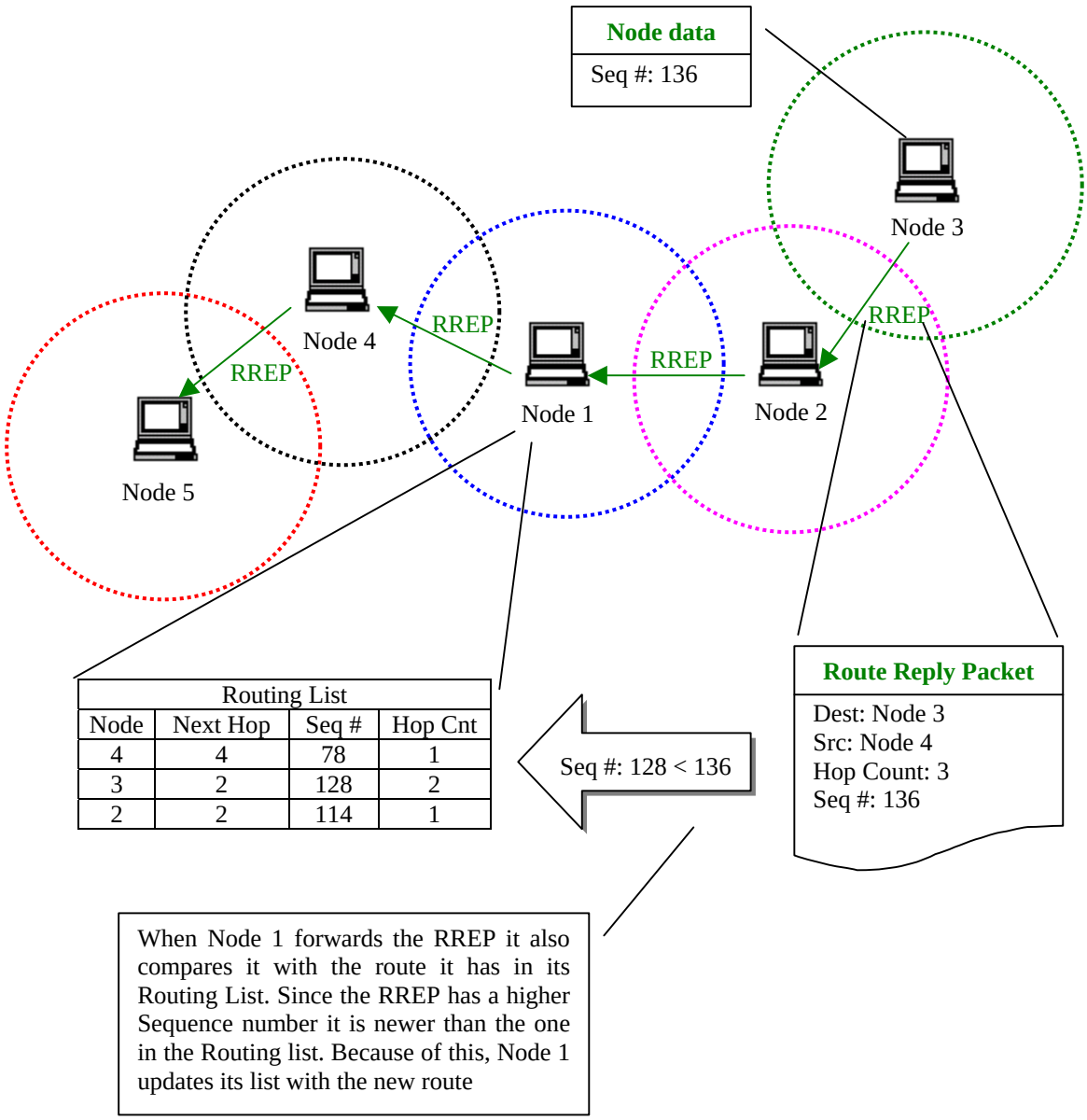


Figure 6-8. Node 1 is forwarding a RREP to node 4

The Route Error Message (RERR) allows AODV to adjust routes when nodes move around. Whenever a node receives RERR, it looks at the routing table and removes all the routes that contain the bad nodes. Figures 6-9 illustrates the three cases under which a node would broadcast a RERR to its neighbors. In the first scenario, the node receives a data packet that it is supposed to forward but it does not have a route to the destination.

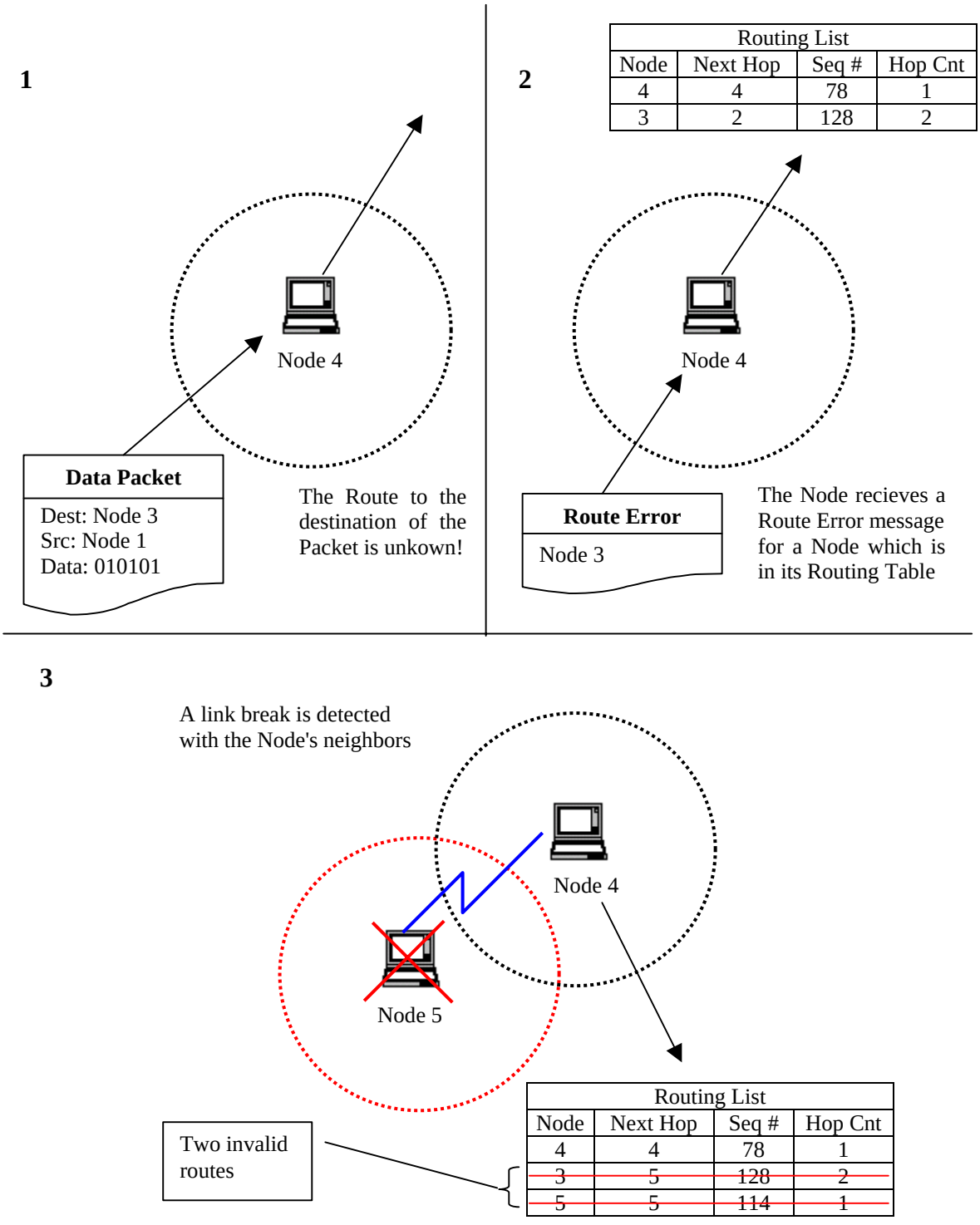


Figure 6-9. Different cases of a node broadcasting a RERR to its neighbors

The real problem is not that the node does not have a route; the problem is that some other nodes may assume that the correct route to the destination is through that node. In the second scenario, the node receives a RERR that causes at least one of its routes to become invalid. If

it happens, the node would then send out a RERR with all the new nodes which are now unreachable. In the third scenario, the node detects that it cannot communicate with one of its neighbors. When this happens, it looks at the routing table for routes that use the neighbor as a next hop and marks them as invalid. Then, it sends out a RERR for the neighbors with the invalid routes.

6.4 Experimental Results

The goal of this experimental study is a delay comparison of a MANET between AODV algorithm and our proposed ant algorithm. Figure 6-10 shows the mean delay when changing the node density. Increasing the node density results in a monotonically increase of the mean delay in case of our proposed ant algorithm, because each hop can contribute a substantial amount of delay in forwarding traffic. Furthermore, the more nodes, the more congestion, and the longer it takes to discover routes. Our proposed ant algorithm results in a considerable reduction in the mean End-to-End delay up to 100 node when compared with AODV but the later gives better results for more than 100 node. Ant algorithm helps in maintaining high connectivity, hence the packets need not to wait in the send buffer until the routes are discovered. Also, the dynamic nature of our ant algorithm keeps routes updated by the ants. This updating process done by the ants leads the source node to switch from a longer route to newer and shorter ones hence reducing mean End-to-End delay for active routes. Lastly, on the other hand side, the overhead of AODV routing protocol increases the mean End-to-End delay in comparison with our ant algorithm.

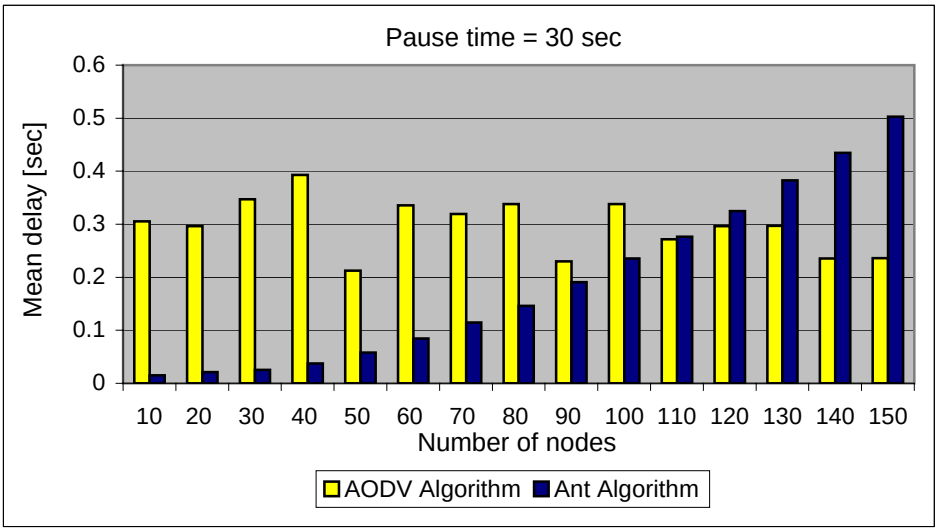


Figure 6-10. Ant algorithm vs. AODV algorithm

Bibliography

[Perkins and Royer, 1999] C. E. Perkins and E. M. Royer. Ad Hoc On-Demand Distance Vector Routing. *Proc. of 2nd IEEE workshop on Mobile Computing Systems and Applications*, pages 90-100, 1999.

[Perkins, 2001] C. E. Perkins. *Ad Hoc Networking*. Addison-Wesley, 2001.

Chapter 7

CONCLUSION & FUTURE WORK

7.1 Introduction

Routing in MANETs is a hard work and actually it is an interesting research area that has been growing in recent years. Its difficulty is mainly generated because of the continuous changes in the network. There exist some traditional solutions such as *proactive* protocols and *reactive* protocols, each one with its advantages and disadvantages. In spite of this, these solutions have to be improved to offer better performance. In fact, there is a new generation of *bio-inspired* routing protocols that have the potential to provide higher performance than pure reactive or proactive protocols and moreover to maintain routing information much longer because of the collaboration between nodes.

In this chapter, we will present and summarize the conclusion of our thesis research, highlight our contribution and discuss potential future research.

7.2 Conclusion

As a special type of network, Mobile Ad hoc Networks (MANETs) have received increasing research attention in recent years. There are many active research projects concerned with MANETs. Mobile ad hoc networks are wireless networks that use multi-hop routing instead of static networks infrastructure to provide network connectivity. MANETs have applications in rapidly deployed and dynamic military and civilian systems. The network topology in MANETs usually changes with time. Therefore, there are new challenges for routing protocols in MANETs since traditional routing protocols may not be suitable for MANETs. For example, one assumption is that a node can receive any broadcast message sent by others

in the same subnet. However, this may not be true for nodes in a wireless mobile network. The bandwidth in this kind of network is usually limited. Thus, this network model introduces great challenges for routing protocols. Moreover, some protocols cannot efficiently handle topology changes.

Researchers are designing new MANETs routing protocols, comparing and improving existing MANETs routing protocols before any routing protocols are standardized using simulations.

As has been explored and described earlier in chapter 4, Ant Colony Optimization (ACO) metaheuristic is one of the most recent bioinspired metaheuristics. It simulates ants behavior when they search for food to solve complex combinatorial problems. Ant systems are self-organizing systems which have many of the characteristics of self-organizing systems. Ant colony optimization (ACO) algorithms are a method for optimization and reinforcement learning. Most importantly, they are decentralized, robust and resilient. These features make ants a good choice for dynamic environments where short paths need to be found, such as in MANETs. New routing protocols based on ACO algorithms are being developed to provide higher performance and greater reliability on MANETs over the traditional protocols. There are many areas in which ant algorithms improved over traditional network routing techniques such as scalability, fault tolerance, adaptation, speed, modularity, autonomy and parallelism. The main purpose of our thesis is to study ant algorithms applied to a dynamic environment such as mobile ad hoc network. We have implemented a routing algorithm working on a simulated network and conducted simulations to test the algorithm. The main question that was to be answered in this thesis was:

Can an ant-inspired routing algorithm improve delay in a dynamic environment such as in a simulated mobile ad hoc network?

We have demonstrated that by combining positive feedback and negative feedback of ant algorithm with queuing network delay analysis, it yields an algorithm which can distribute the package transfer over several paths in a simulated network. This algorithm ensures a lower node to node delay when compared with the traditional routing protocols.

Our contribution includes the presentation of a new ant-based algorithm for routing in mobile ad hoc networks. It is an algorithm, combining ant algorithms with queuing network delay analysis.

Our study shows that ant colony optimization algorithms are suitable for mobile ad hoc networks. These ant colony optimization algorithms illustrate different reasons about why this kind of algorithms could perform well in mobile ad hoc networks. Firstly, the ant colony

optimization algorithms are based on agent systems and work with individual ants. This allows a high adaptation to the current topology of the network. The dynamic topology property is responsible for the bad performance of several routing algorithms in mobile ad hoc networks. Secondly, in contrast to other routing approaches, the ant colony optimization algorithms are based only on local information, i.e., no routing tables or other information blocks have to be transmitted to neighbors or to all nodes of the network. Finally, each node has a routing table with entries for all its neighbors, which contains also the pheromone concentration. The decision rule, to select the next node, is based on the pheromone concentration on the current node, which is provided for each possible link. Thus, the approach supports multipath routing.

We have designed an ad hoc routing algorithm which operates using reinforcement learning to define a model of optimal routing behavior in an ad hoc network. In this model, optimal behavior is not merely searching shortest-hop paths, but also considers the quality of the links which make up those paths.

We have devised a learning strategy for continuous monitoring of the links in the network and movement of routing information throughout the network. The learning strategy we have designed is based on work in ant colony optimization (ACO) algorithms. We adapt some of the unique features of ant colony optimization algorithms which are applicable to the ad hoc routing problem. In particular, each packet routed by the protocol is equivalent to an ant in ant colony optimization techniques: the progress of the packet through the network incrementally changes the routing policy and the paths which are used by future packets.

We have implemented this routing algorithm using simulation, and compared its performance to that of AODV in a simple network scenario.

We have found that our model which is based on ant algorithm is extremely valuable in comparison with AODV.

We can summarize our final conclusion from our experimental results as follows:

- Increasing the density of nodes yields to an increasing in the mean End-to-End delay.
- Increasing the pause time leads to a decrease in the mean End-to-End delay.
- There is no significant effect on the delay between RWM and BSAM models.
- Increasing in the number of nodes will cause increasing in the mean time for loop detection.
- Our proposed ant algorithm results in a considerable reduction in the mean End-to-End delay up to 100 nodes when compared with AODV but the later gives better results for more than 100 nodes.

Finally, we can conclude that the considered ant algorithm is able to cope with this type of dynamic networks, in particular its ability to improve the system performance which has been reflected in our model.

7.3 Future Work

We identify the following areas of future research:

- Study the operation of our proposed ant routing algorithm under various metrics such as available energy at each node in the network.
- We have discussed the behavior of our proposed ant algorithm with mobility models that represent multiple mobile nodes MNs whose actions are completely independent of each other. In an ad hoc network, there are many other situations where it is necessary to model the behavior of MNs as they move together. One of our future research studies is the study of the behavior of our proposed algorithm with other mobility models such as *Reference Point Group Mobility* (RPGM) model which represents multiple MNs moving together. Finally, we can compare between these different mobility models to show its effects on the performance of our proposed ant routing algorithm.
- Create more realistic movement models through the incorporation of obstacles. These obstacles are utilized to both restrict node movements as well as wireless transmissions. The aim of this study is to show how the use of obstacles has a significant impact on the performance of our proposed ant routing algorithm for ad hoc networks.
- A study of different traffic scenarios such as using multiple source nodes and multiple destination nodes are necessary in order to show the general applicability of our proposed ant routing algorithm.
- Despite the differences between both ant colony optimization (ACO) algorithms and the new evolutionary computation (EC) algorithms such as in genetic algorithms, both share properties that allow to construct a new special hybrid ant algorithm which includes some EC components by integrating of both algorithms. This integration of both algorithms can improve the performance of our algorithm. It will be necessary to have a deeper insight into EC and ACO algorithms to identify all exploitation and exploration components in both techniques.
- Compare the performance of our proposed ant routing algorithm with one of the proactive (table driven) routing protocols such as *Destination Sequenced Distance Vector* (DSDV) routing protocol.

APPENDIX

A. Mobile Ad hoc Network Applications

The following Table A-1 provides an overview of present and future MANET applications [Chlamtac et al., 2003].

Table A-1. Mobile ad hoc network applications

Applications	Possible scenarios/services
Tactical networks	• Military communication and operations • Automated battlefields
Sensor networks	• Home applications: smart sensor nodes and actuators embedded in consumer electronics to allow end users to manage home devices locally and remotely. • Environmental applications include tracking the movements of animals (e.g., birds and insects), chemical/biological detection, precision agriculture, etc. • Tracking data highly correlated in time and space, e.g., remote sensors for weather, earth activities. • Body area networks (BAN).
Emergency services	• Search and rescue operations. • Disaster recovery. • Replacement of a fixed infrastructure in case of environmental disasters (e.g., earthquakes, hurricanes) • Policing and fire fighting. • Supporting doctors and nurses in hospitals, e.g., early retrieval and transmission of patient data (record, status, diagnosis) from/to the hospital.
Commercial and civilian environments	• E-Commerce: e.g., Electronic payments anytime and anywhere. • Business: dynamic database access to customer files, mobile offices. • Vehicular Services: road or accident guidance, transmission of road and weather conditions, taxi cab network, inter-vehicle networks. Local ad hoc network with nearby vehicles for road/accident

	guidance. • Sports stadiums, trade fairs, shopping malls. • Networks of visitors at airports.
Home and enterprise networking	• Home/Office Wireless Networking (WLAN) e.g., shared white-board application; use PDA to print anywhere; trade shows. • Personal Area Network (PAN). • Conferences, meeting rooms. • Networks at construction sites.
Educational applications	• Setup virtual classrooms or conference rooms. • Setup ad hoc communication during conferences, meetings or lectures. • Universities and campus settings.
Entertainment	• Multi-user games. • Robotic pets. • Outdoor Internet access. • Wireless P2P networking. • Theme parks.
Location aware services	• Follow-on services, e.g., automatic call-forwarding, transmission of the actual workspace to the current location. • Information services: push, e.g., advertise location specific service, like gas stations. pull, e.g., location dependent travel guide; services (printer, fax, phone, server, gas stations) availability information. • Touristic information.
Coverage extension	Extending cellular network access. Linking up with the Internet, intranets, etc.

B. Overview of Ad hoc Routing Protocols

We have compiled a list of every routing protocol we have found. It's far from certainty that we have mentioned every currently existing protocol, because there are so many new and different variations of protocols being developed all the time. The list of protocols is shown in Figure B-1. We have divided the protocols into different categories according to their characteristics. In this appendix, we just give a general overview about only some of reactive and proactive protocols.

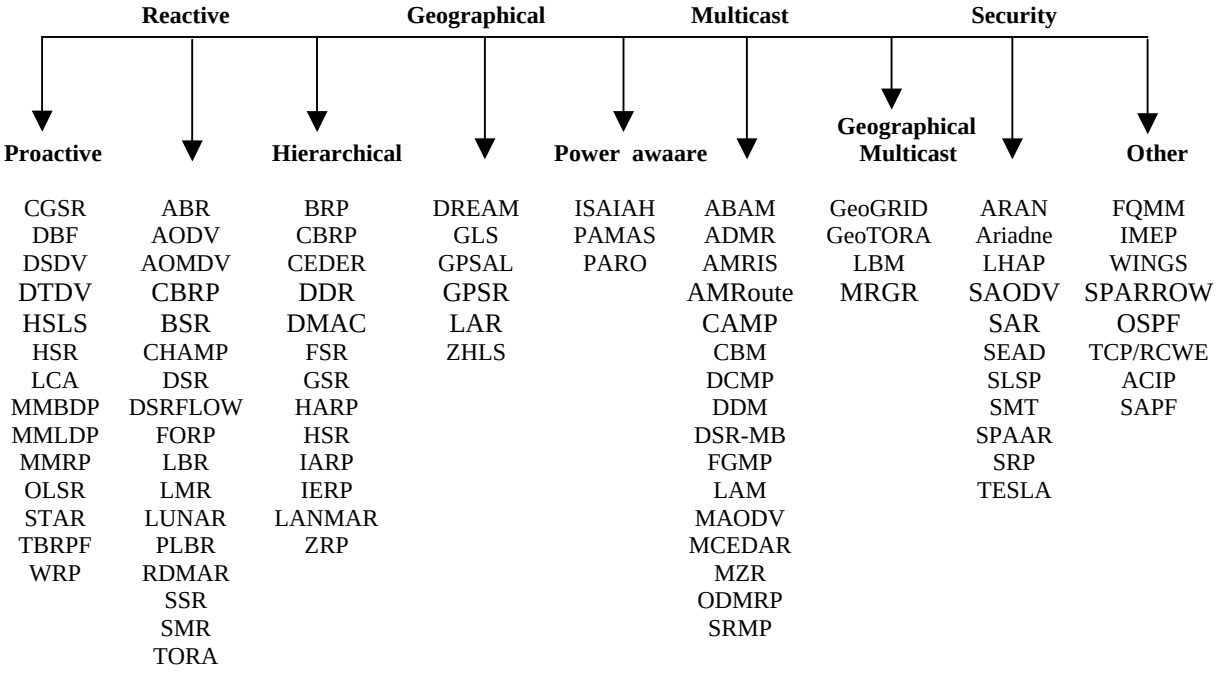


Figure B-1. List of routing protocols

B.1 Table Driven Routing Protocols (Proactive)

Protocols that keep track of routes for all destinations in the ad hoc have the advantage that communications with arbitrary destinations experience minimal initial delay from the point of view of the application. When the application starts, a route can be immediately selected from the routing table. Such protocols are called proactive because they store route information even before it is needed. They are also called table driven because routes are available as parts of a well-maintained table. Certain proactive routing protocols are Destination-Sequenced Distance Vector (DSDV), Wireless Routing Protocol (WRP), Global State Routing (GSR) and Clusterhead Gateway Switch Routing (CGSR). Proactive protocols suffer the disadvantage of additional control traffic that is needed to continually update stale route entries. Since the network topology is dynamic, when a link goes down, all paths that use that

link are broken and have to be repaired. If no applications are using these paths, then the effort done to repair them may be considered wasted. This wasted effort can cause scarce bandwidth resources to be wasted and can cause further congestion at intermediate network points. This strategy is appropriate for a network with low mobility.

B.1.1 Destination Sequenced Distance Vector (DSDV)

Destination sequenced distance vector routing protocol is derived from a classical distance vector algorithm, Distributed Bellman-Ford (DBF) algorithm. Enhancements are made in order to avoid the looping problem present in the basic DBF. Formation of loops is avoided by tagging each route table entry with a sequence number to order the routing information.
This is a proactive protocol, that updates routing information on a regular basis. To avoid routing loops, destination sequence numbers have been introduced. DSDV is one of the first attempts to adapt an established routing mechanism to work with mobile ad hoc networks.
Each routing table lists all destinations with their current hop count and sequence number. Routing information is broadcast or multicast. Each node transmits its routing table to its neighbors. Routes with more recent sequence numbers replace obsolete older routes. This mechanism provides loop freedom and prevents stale routes.
The routing information is transmitted every time a change in the topology has been detected (i.e. a change in the set of neighbors of a node). DSDV works only with bidirectional links. DSDV was presented in [Perkins and Bhagwat, 1994]. A more detailed description is available in [Perkins, 2001].

B.1.2 Wireless Routing Protocol (WRP)

The Wireless Routing Protocol WRP is one of the first suggestions of a routing algorithm for mobile ad hoc networks [Murthy and Garcia, 1996]. WRP is related to the DBF algorithm. Routing update messages are only sent locally to the neighbor set. They contain all the routing information known by the originating node. Of course, not the whole routing table is sent in each update. Only changes are transmitted, either by receiving an update from another node or from a changed link in the neighborhood. WRP is a proactive routing protocol, since routes are maintained all the time and no special route requests by source nodes need to be performed. The routing table consists of an entry for each destination with the next hop and a cost metric. The routes are selected by choosing the node from the neighbor set, which provides the path with the lowest cost (provided it’s loopfree), as next hop. The link costs will be kept in a separate table, but it is not specified how the cost for each link should be

determined. Various possiblities exist: hop count, end-to-end delay, utilization, etc. To keep the state of the neighbor links up to date, empty update messages (HELLO messages) are sent in a regular fashion, if no other updates would be sent anyway. Update messages which are not empty, need to be acknowledged.

B.1.3 Global State Routing (GSR)

Global State Routing was developed by Tsu-wei Chen and Mario Gerla [Gerla and Chen, 1998]. It is an early attempt to introduce link state routing in an ad hoc networking context. The main problem of traditional link state routing, is the high amount of topology information, which is sent from each router to each other router. Since in ad hoc networks, each node is also a router, this mechanism does not scale and needs to be limited. Therefore GSR adopts the information dissemination process from the Distributed Bellman-Ford algorithm. In GSR, topology information is exchanged periodically only among neighbors. If a topology change occurs, this change is transmitted further. Messages are send only in such triggered cases. GSR uses sequence numbers based on timestamps. GSR was evaluated in simulation and compared to traditional link state and Distributed Bellman-Ford.

B.1.4 Fisheye State Routing (FSR)

Fisheye State Routing was proposed by Mario Gerla et.al. to the MANET IETF working group [Gerla, 2000]. FSR wants to reduce unnecessary traffic by introducing a multi-level scope. By concept, FSR is a protocol that periodically updates link state information (table driven). FSR is derived from Global State Routing. The major drawback of GSR is the large message size and the propagation latency of the link state changes. FSR now helps by introducing scopes, which depend on the number of hops a packet has passed on from its source. Nodes within the smallest scope are considered most often in update packets; nodes, which are far away are considered much less frequent. This means, the message size can be greatly reduced, as information for most nodes can be omitted. Although routes may become inaccurate for distant destinations under increased mobility, packets will find more and more accurate routes while getting closer to the target, thus they don't suffer much from the inaccuracy.

B.1.5 Hierarchical State Routing (HSR)

HSR introduces a multilevel clustering infrastructure [Atsushi Iwata et al., 1999]. Clustering is done on a physical and logical basis. Physical clustering starts with level 0, the bottom

layer. On each level, nodes can form clusters, which are represented by a clusterhead each. The clusterhead itself can form another cluster at the next higher level. On higher levels, the clusterheads are connected via virtual links, which need to be mapped to physical links on the bottom layer. A virtual link will usually contain gateway nodes on the lowest level. Each clusterhead collects link state information of each cluster member, regarding its neighbors, and propagates a summary to its fellow clusterheads on the higher levels, possibly using gateway nodes. On the higher levels, the same happens with the link state information about the virtual links, which are computed from the lower level link states. A special hierarchical addressing scheme is used, which is sufficient to route a message from any node to any other node. A node passes a message up to the node of the highest level in its current hierarchy. This one will pass it to the destination cluster node (through a virtual link), which will pass it down the levels to the right node on the lowest level. Additionally to physical clustering, a logical partitioning is used, which works similar to Mobile IP. HSR shows its advantage as being the most scalable approach, FSR instead does not perform equally well.

B.1.6 Clusterhead Gateway Switch Routing (CGSR)

Clusterhead Gateway Switch Routing was proposed in [Chiang et al., 1997]. It consists of a clustering scheme, called *Least Cluster Change*, which is combined with either lowest ID or maximum links, to form clusters and elect clusterheads. The scheme focuses on cluster stability. CGSR explicitly specifies requirements on the link layer and medium access scheme:

- Inter-cluster communication requires a CDMA system, such that each cluster is assigned a different code.
- Within each cluster, TDMA is used. The allocation of time slots is done by a token passing method.

Gateway nodes are nodes, that are within more than one cluster, and therefore need to communicate in different codes. The protocol uses a sequence number scheme (as developed in DSDV) to gain loop-free routes and avoid stale routing entries. In CGSR, a packet is routed alternating between clusterheads and gateways, hence the name.

B.2 On-Demand Routing Protocols (Reactive)

To overcome the wasted work in maintaining unrequired routes, on-demand or reactive protocols have been designed. In these protocols, routing information is acquired only when it is actually needed. Reactive routing protocols save the overhead of maintaining unused routes

at each node, but the latency for many applications will drastically increase. Most applications are likely to suffer a long delay when they start because a route to the destination will be acquired before the communication can begin. Some reactive protocols are Cluster Based Routing Protocol (CBRP), Ad hoc On-Demand Distance Vector (AODV), Dynamic Source Routing (DSR), Temporally Ordered Routing Algorithm (TORA) and Associativity-Based Routing (ABR). Reactive protocols may not be optimal in terms of bandwidth utilization because of flooding of the route discovery request, but they remain scalable in the frequency of topology change.

B.2.1 Cluster Based Routing Protocol (CBRP)

CBRP maintains clusters of two hops diameter, with an elected clusterhead for each cluster [Martha, 2001]. Clusters may be overlapping, but each node must be part of at least one cluster. Clusterheads are not allowed to be direct neighbors, except for a short period called *contention period*. Nodes maintain a neighbor table which also includes the link type. Also, a cluster adjacency table is kept in each node. Source routing is used, with the route in the CBRP header. This allows a limited local repair mechanism and a route cache (much like DSR) to be used. For clustered routing, the key argument is that with a clustered hierarchy, it is again possible to channel information. Thus, scalability may be regained even if broadcasts need to be used.

B.2.2 Dynamic Source Routing (DSR)

Dynamic source routing is based on source routing, where the source specifies the complete path to the destination in the packet header and each node along this path simply forwards the packet to the next hop indicated in the path. It was first described in [Johnson and Maltz, 1996]. Since DSR works on demand, a route must be discovered through a Route Discovery Mechanism before use. Discovered routes may be cached and routes may be overheard by a node by parsing the source route information of packets that are relayed. If broken links are detected, a corresponding Route Error message is transmitted through the network and a route maintenance mechanism takes over to fix the broken routes, if possible.

For further reduction of unnecessary traffic, a node may reply to a route request with a locally cached route, even if it is not the destination node. Delays in these replys with promiscous observation (overhearing) of other routing traffic prevent multiple nodes replying with a cached entry all at once. The dynamic source routing protocol is also a very mature protocol. DSR is also one of the few ad hoc routing protocols that have been implemented and

evaluated in a real testbed. DSR was used in many performance comparisons, evaluating studies, and was used as a reference for a lot of other protocols. Further, it was used as a reference protocol for investigations to find general improvements for mobile ad hoc networks (like reduced energy consumption). The major advantage of DSR is that there is little or no routing overhead when a single or few sources communicate with infrequently accessed destinations. In such situation, it does not make sense to maintain routes from all sources to such destinations.

B.2.3 Temporally Ordered Routing Algorithm (TORA)

The Temporally Ordered Routing Algorithm (TORA) is a highly adaptive, efficient and scalable distributed routing algorithm based on the concept of link reversal [Park and Corson, 1997]. TORA is proposed for highly dynamic mobile, multihop wireless networks. It is a source-initiated on-demand routing protocol. It finds multiple routes from a source node to a destination node. The main feature of TORA is that the control messages are localized to a very small set of nodes near the occurrence of a topological change. To achive this, the nodes maintain routing information about adjacent nodes. The protocol has three basic functions: *Route creation*, *Route maintenance* and *Route erasure*. TORA can suffer from unbounded worst-case convergence time for a very stressful scenarios. TORA converges quickly and performs significantly better than *Idealized Link State* ILS algorithm.

B.2.4 Associativity Based Routing (ABR)

ABR is an on-demand routing protocol in which routes are discovered with a broadcast query request [Toh, 1996]. From these requests, the destination learns all possible routes and replies along a selected route to the source. If a route breaks, several route-reconstruction methods can be applied, depending on which node moves out of reach: the source, the destination or an intermediate node. Further, ABR maintains a *degree of associativity* in form of associativity ticks. These are not clearly defined, but from context it appears that every node maintains a tick-value for every one of its neighbors. Every time interval, a link-layer hello message from each neighbor is received and the tick value is increased. If the neighbor moves out of reach, the value is reset to zero. A tick level above a certain threshold indicates a stable association between those two nodes. On selecting a route, the destination (which does the selection) prefers most stable routes, i.e. those with the highest associativity tick value. Hence, this degree of associativity is used as a metric of mobility.

C. Ant Colony Optimization (ACO) Applications

The following Tables C-1 and C-2 list the available implementations of ACO algorithms.

Table C-1. List of applications of ACO algorithms to static optimization problems

Problem name	Authors	Year	Algorithm name
Traveling salesman	Dorigo, Maniezzo, & Colorni	1991	AS
	Gambardella & Dorigo	1995	Ant-Q
	Dorigo & Gambardella	1996	ACS & ACS-3-opt
Quadratic assignment	Stützle & Hoos	1997	MM AS
	Bullnheimer, Hartl, & Strauss	1997	A*Srank*
	Maniezzo, Colorni, & Dorigo	1994	AS-QAP
	Gambardella, Taillard, & Dorigo	1997	HAS-QAP[a]
	Stützle & Hoos	1998	MM AS-QAP
	Maniezzo & Colorni	1998	AS-QAP[b]
	Maniezzo	1998	ANTS-QAP
Job-shop scheduling	Colorni, Dorigo, & Maniezzo	1994	AS-JSP
Vehicle routing	Bullnheimer, Hartl, & Strauss	1996	AS-VRP
	Gambardella, Taillard, & Agazzi	1999	HAS-VRP
Sequential ordering	Gambardella & Dorigo	1997	HAS-SOP
Graph coloring	Costa & Hertz	1997	ANTCOL
Shortest common super sequence	Michel & Middendorf	1998	AS-SCS

Table C-2. List of applications of ACO algorithms to dynamic optimization problems

Problem name	Authors	Year	Algorithm name
Connection-oriented network routing	Schoonderwoerd, Holland, Bruten, & Rothkrantz	1996	ABC
	White, Pagurek, & Oppacher	1998	ASGA
	Di Caro & Dorigo	1998	AntNet-FS
	Bonabeau, Henaux, Guérin, Snyers, Kuntz, & Theraulaz	1998	ABC-smart ants
Connectionless network routing	Di Caro & Dorigo Subramanian	1997	AntNet & AntNet-FA
	Druschel, & Chen	1997	Regular ants
	Heusse, Guérin, Snyers, & Kuntz	1998	CAF
	van der Put & Rothkrantz	1998	ABC-backward

[a] HAS-QAP is an ant algorithm that does not follow all the aspects of the ACO metaheuristic.

[b] This is a variant of the original AS-QAP.

D. Mobility Models Used in Simulations

To evaluate the performance of a protocol for an ad hoc network, it is necessary to test the protocol under realistic conditions, especially including the movement of the mobile nodes. We will present a survey of different mobility models including the Random Waypoint Mobility (RWM) and Boundless Simulation Area Mobility (BSAM) models which are used in our thesis.

D.1 Random Walk Mobility Model

Random walk is a simple mobility model which based on random directions and speeds [Davies, 2000]. By randomly choosing a direction between *0* and 2π and a speed between 0 and V_{max}, the mobile node moves from its current position. A recalculation of speed and direction occurs after a given time or a given walked distance. The random walk mobility model is memoryless. Future directions and speeds are independent of the past directions and speeds. This can cause unrealistic movement such as sharp turns or sudden stops. If the specified time or distance is short, the nodes are only walking on a very restricted area of the simulation area.

D.2 Random Waypoint Model (RWM)

The Random Waypoint Mobility (RWM) model used by Johnson includes pause times between changes in direction and/or speed [Johnson and Maltz, 1996].

RWM works as follows: all nodes are uniformly distributed around the simulation area at starting time. Each node then chooses a random destination and moves toward it with a speed uniformly distributed over [*0*, V_{max}]. Then, it stops for a pause time which could be selected as 0 to give continuous motion. The random waypoint mobility model is very widely used in simulation studies of MANET.

D.3 Random Direction Model (RDM)

The random direction mobility (RDM) model was created in order to overcome a flaw discovered in RWM model [Davies, 2000]. This a more *stable* model than a random waypoint model. At start, the nodes select random directions and start to move along them. Since the area of simulation is confined, the nodes may end up reaching one of the boundaries during the simulation. When a boundary is reached, the nodes pause for a given time and then choose new directions to travel. Since the node is on a boundary, the selectable angle is 180 degrees.

The result of this model is a more stable distribution of the nodes than the RWM. The behavior can be thought as a micro-cell of a larger area which is a useful property.

D.4 Modified Random Direction Model (MRDM)

Modified Random direction model (MRDM) is more advanced version described in [Royer et al., 2001]. To give an even more realistic simulation, the RDM was extended with an extra choice for the nodes when their pause time is over. The nodes don't have to travel all the way to the boundary but could stop anywhere along the path.

D.5 Brownian Model (BM)

Hu and Johnson describe in [Hu and Johnson, 2000] another way of modeling the speed of the nodes. At the beginning of each time interval, each node changes speed and direction by choosing $r \in [0, V_{max}]$ and $\theta \in [-\pi, \pi]$ and moves with velocity vector $(r \sin\theta, r \cos\theta)$. This model is very similar to the random direction model except for the speed which is smooth in this model.

D.6 Column Model (CM)

A mobility model suited for experiments is described by Sanchez in [Sanchez, 2001]. Nodes are only moving along the x-axis. The initial position of node i is $(10i, \ 10i)$ and the node changes the speed $v \in [0, V_{max}]$ at the discrete intervals. This will produce a simpler mobility pattern than the random mobility model since the nodes only move along the x-axis (one dimension).

D.7 Random Gauss-Markov Model (RGM)

This model uses discrete time intervals to divide up the motion. The nodes update their velocity and direction vectors at the beginning of each interval according to:

$$v_n = \alpha \ v_{n-1} + (1-\alpha) * \bar{v} + R \ \sqrt{1-\alpha^2}$$

$$d_n = \alpha \ d_{n-1} + (1-\alpha) * \bar{d} + R \ \sqrt{1-\alpha^2}$$

where v_n and d_n are the new velocity and direction of the MN at time interval n; α, where $0 \leq \alpha \leq 1$, is the tuning parameter used to vary the randomness; $\bar{v}$ and $\bar{d}$ are constants representing the mean value of speed and direction. R is a random variable with mean 0 and variance σ. This model is described by Sanchez [Sanchez, 2001].

D.8 Pursue Model (PM)

Another model done by Sanchez [Sanchez, 2001] is to create group of movements. One node in each group is moving according to the random waypoint model. The rest of the group is moving toward the target that the leading is aiming for. The speed of the pursuing nodes is chosen uniform random in the range [Vp_{min}, Vp_{max}].

D.9 Exponential Correlated Random Model (ECR)

The ECR is able to model all possible movements of individuals and groups [Bergamo, 1996]. This is done by changing the parameters of a motion function. A new position *b(t+1)* is a function of the previous position *b* to which a random deviation is added. The function $b(t) = (r_t, 0_t)$ can be defined either for a single node or a group at time t. *r* is a random Gaussian variable with variance σ. The parameters are then changed to give different mobility patterns. Very hard to create a predefined motion pattern by selecting the parameters.

D.10 Reference Point Group Mobility Model (RPGM)

Hong et al. describe another way to simulate group behavior in [Hong et al., 1999] where each node belongs to a group in which every node follows a logical center reference point. The nodes in a group are usually randomly distributed around the reference point. The different nodes use their own mobility model and are then added to the reference point which drives them in the direction of the group. This general description of group mobility can be used to create a variety of models for different kinds of mobility applications.

D.11 Individual Simulated Behavioral Model (ISB)

This is another new and different idea about how to do more accurate and better simulations. They use a theory about an individually simulated behavioral model where all objects have their own properties. They verified their idea with DSR and proved that it generates reproducible and realistic mobility patterns [Tan et al., 2002].

Bibliography

[Atsushi Iwata et al., 1999] Atsushi Iwata, Ching-Chuan Chiang, Guangyu Pei, Mario Gerla, and Tsu wei Chen. Scalable routing strategies for ad hoc wireless networks. *Tech. Rep., Department of Computer Science University of California, Los Angeles*, 1999.

[Bergamo, 1996] M. Bergamo. System design specification for mobile multimedia wireless network(MMWN). *Technical report, DARPA project*,1996.

[Chiang et al., 1997] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in clustered multihop, mobile wireless networks with fading channel. *Tech. Rep., University of California at Los Angeles Computer Science Department*, 1997.

[Chlamtac et al., 2003] I. Chlamtac, M. Conti and J. J.-N. Liu. Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, Vol.(1), pages 13–64, 2003.

[Davies, 2000] V. Davies. Evaluating mobility models within an ad hoc network. *Master's thesis, Colorado School of Mines*, 2000.

[Gerla and Chen, 1998] Mario Gerla and Tsu-Wei Chen. Global state routing: A new routing scheme for ad-hoc wireless networks. *Tech. Rep., Computer Science Department, University of California, Los Angeles*, 1998.

[Gerla, 2000] Mario Gerla, Guangyu Pei, Xiaoyan Hong, and Tsu-Wei Chen. Fisheye state routing protocol (fsr) for ad hoc networks. *Internet Draft http://www.ietf.org/internet-drafts/draft-ietf-manet-fsr-00.txt*, November 2000.

[Hong et al., 1999] X. Hong, M. Gerla, G. Pei, and C. Chiang. A group mobility model for ad hoc wireless networks. *In Proceedings of the ACM International Workshop on Modeling and Simulation of Wireless and Mobile Systems (MSWiM)*, 1999.

[Hu and Johnson, 2000] Yih-Chun Hu, David B. Johnson. Caching Strategies in On-Demand Routing Protocols for Wireless Ad Hoc Networks. *In Proceedings of The Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, 2000.

[Johnson and Maltz, 1996] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In T. Imelinsky and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[Johnson and Maltz, 1996] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks, *Tech. Rep., Computer Science Department Carnegie Mellon University, Pittsburgh*, 1996.

[Martha, 2001] Martha Steenstrup. Cluster-Based Networks, chapter 4, pp. 75-138, In [Perkins, 2001], 2001.

[Murthy and Garcia, 1996] Shree Murthy and J.J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Tech. Rep., Copmuter Engineering, University of California at Santa Cruz, Santa Cruz* 1996.

[Park and Corson, 1997] Vincent D. Park and M.Scott Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. *In Proceedings of INFOCOM 1997,* 1997.

[Perkins and Bhagwat, 1994] Charles E. Perkins and Pravin Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. *In ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pp. 234-244, 1994.

[Perkins, 2001] Charles E. Perkins, Ad Hoc Networking, Addison-Wesley, 2001.

[Royer et al., 2001] Elizabeth M. Royer, P. M. Melliar-Smith, Louise E. Moser. An Analysis of the Optimum Node Density for Ad hoc Mobile Networks. *Proceedings of the IEEE International Conference on Communications, Helsinki, Finland*, 2001.

[Sanchez, 2001] M. Sanchez. Mobility models.

[Tan et al., 2002] D.S.Tan, S. Zhou, J. Ho, J.S. Mehta, H. Tanabe. Design and Evaluation of an Individually Simulated Mobility Model in Wireless Ad Hoc Networks. Communication Networks and Distributed Systems Modeling and Simulation Conference, San Antonio, 2002.

[Toh, 1996] Chai-Keong Toh. Associativity-based routing for ad-hoc mobile networks. *Tech. Rep., University of Cambridge Computer Laboratory, Cambridge, United Kingdom*, 1996.

See URL: http://www.disca.upv.es/misan/mobmodel.htm, 2001.