ABSTRACT

Title of Thesis:	DYNAMIC ADAPTIVE ROUTING IN MOBILE AD HOC NETWORKS
Degree candidate:	Harsh Mehta
Degree and year:	Master of Science, 2002
Thesis directed by:	Professor John S. Baras Department of Electrical Engineering

Mobile Ad Hoc Networks are infrastructure-less networks consisting of wireless, possibly mobile nodes which are organized in peer-to-peer and autonomous fashion. They are ideal for use in scenarios such as battlefield operations, disaster relief and emergency situations where fixed infrastructure is not available for communications and rapid deployment is important. This environment provides substantial challenges for routing. The highly dynamic topologies, limited bandwidth availability and energy constraints make the routing problem a challenging one.

The Swarm Intelligence paradigm is derived from the study of group behavior in animals such as ants, bees and other insects and has been used to solve various hard optimization problems such as the travelling salesman problem and has recently been used in solving the routing problem in static computer networks with

encouraging results. These algorithms have proved to be resilient and robust to topology changes.

In this thesis we take a novel approach to the unicast routing problem in MANETs by using swarm intelligence-inspired algorithms. The proposed algorithm uses Ant-like agents to discover and maintain paths in a MANET with dynamic topology. We present simulation results that measure the performance of our algorithm with respect to the characteristics of a MANET, the varying parameters of the algorithm itself as well as performance comparison with other well-known MANET routing protocols.

DYNAMIC ADAPTIVE ROUTING IN MOBILE AD HOC NETWORKS

by

Harsh Mehta

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
2002

Advisory Committee:

Professor John S. Baras, Chair
Professor Virgil Gligor
Doctor Mark Fleischer

ACKNOWLEDGMENTS

I am grateful to my advisor, Professor John S. Baras for his advice, support and encouragement. I would also like to thank Dr. Mark Fleischer for constant encouragement and helpful discussions. I would like to thank Prof. Virgil Gligor and Dr. Mark Fleischer for agreeing to serve on my committee and reviewing the thesis. Sincere thanks to Dinesh Dharamaraju for help with the simulations, Tao Jiang, Prabha Ramachandran, Pedram Hovareshti and Karthikeyan Chandrasekaran for their valuable suggestions and comments. I am grateful for the support of my research work and graduate studies through the following contracts and grants (all with the University of Maryland College Park):

ARL DAAD190120011 (through the Collaborative Technology Alliance in Communications and Networks), Lockheed Martin Corporation and the Maryland Industrial Partnerships Program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background and Motivation

1.1.1 Mobile Ad hoc NETworks (MANETs)

Ad hoc networks (MANETs) are infrastructure-less, autonomous networks comprised of wireless mobile computing devices. All the nodes in the network have the same capability to communicate with each other without the intervention or need of a centralized access point or base-station. The mobile nodes or devices are equipped with wireless transmitters and receivers. These antennas could be omni-directional resulting in a broadcast medium or highly directional resulting in a point-to-point network. Due to limited transmission range of wireless interfaces, these networks are multi-hop networks i.e., a node may have to relay a message through several intermediate nodes for the message to reach the destination. Thus every node is a router as well as a host in a MANET. The arbitrary movement of the nodes in such networks results in highly dynamic or ad hoc topologies. A

MANET can thus be considered as a dynamic multi-hop graph. Lower capacity of wireless channels as compared to wired links, coupled with effects of interference, fading and noise reduce the effective available bandwidth for communication. Thus bandwidth is at a premium in MANETs. Moreover, since the mobile devices are dependent on batteries for their operation, these networks are also energy constrained.

MANETs are attractive as they provide instant network setup without any fixed infrastructure. The ease and speed of deployment of these networks makes them ideal for battlefield communications, disaster recovery, conferencing, electronic classrooms, etc.

1.1.2 Routing in MANETs

1.1.2.1 Routing in MANETs

Routing in Mobile Ad Hoc Networks is a difficult problem due to the bandwidth and 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 by a number of nodes before reaching the destination from the source.

1.1.2.2 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. The presence of stale or out-dated routes results in large packet losses [27]. To alleviate this problem, the protocol should be resilient to topology changes. Reactive protocols that try to mend disconnected paths incur large control overhead [27] 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, variable capacity links

MANETs use wireless links which offer limited bandwidth for transmission. Further, these links maybe 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.

1.2 Contributions

This thesis presents a new, dynamic, adaptive routing algorithm for mobile ad hoc networks inspired by the swarm intelligence paradigm [4]. Swarm intelligence algorithms have been thoroughly investigated in the past for dynamic problems such as the Travelling Salesman problem and even the routing problem in communication networks. Their performance has been found to comparable or better than existing algorithms. Further, in the MANET environment, Swarm Intelligence algorithms, due to their distributed and emergent nature, provide robustness under tough operating conditions and resistance to router state corruption.

A routing scheme is proposed that exhibits emergent behavior. This behavior is based on a type of learning algorithm, similar to AntNet [6], which provides natural resistance to the corruption of routing table entries. The algorithm uses Ant-like agents, *forward and backward ants*, as routing packets that explore available paths to a destination in the network through a number of nodes. These agents deposit

pheromone or a "goodness value", a probability distribution for taking a certain next hop to reach the destination. The agents reinforce good routes and reduce the goodness value of routes which do not perform well based on certain criteria; e.g. delay or number of hops. This is similar to the behavior of ants in nature when they forage for food as a colony. The communication between the ants is through the environment (pheromone, a chemical in this case). Global information, that is a complete route, emerges from local information, that is, the goodness value of a certain next hop for the packets destination.

We evaluate and quantify the overhead, delay, goodput and throughput of these algorithms. Based on trade-off curves, we suggest a range of operating values for various parameters that characterize a MANET vis-a-vis mobility speed and algorithm parameters.

1.3 Organization of the Thesis

The rest of the dissertation is as follows. Chapter 2 introduces the swarm intelligence paradigm and previous work on routing in communication networks using swarm intelligence heuristics. In chapter 3, some of the existing routing protocols for MANETs are discussed. Chapter 4 introduces a unicast, swarm intelligence based routing algorithm for MANETs. Chapter 5 introduces a dynamic adaptive routing algorithm based on the swarm intelligence paradigm which uses a flooding-based approach for route discovery. In chapter 6, the important parameters of the

algorithm are discussed, and chapter 7 presents the simulation methodology and the results. Chapter 8 concludes the dissertation.

Chapter 2

The Swarm Intelligence paradigm

2.1 Swarm Intelligence heuristics for solving hard optimization problems

Several animal species, such as bees and ants use communication via the environment. Many ant species have trail-laying trail-following behavior when foraging: individual ants deposit a chemical substance called *pheromone* as they move from a food source to their nest, and foragers follow such pheromone trails. As an emergent result of the actions of many ants, the shortest path between two or more locations (often between a food source and a nest) is discovered and maintained. This process of emerging global information from local actions through small, independent agents not communicating with each other is called *Stigmergy*. This property is used in algorithms and heuristics to solve various NP-hard problems. Several such approaches and problems are discussed in [4]. The following sections briefly discuss some such approaches for solving problems such as the travelling

salesman problem (TSP) and the routing problem in communication networks.

2.1.1 The Travelling Salesman Problem

Since the interaction of Ants in the natural environment produces and maintains shortest paths between a food source and a nest, the first natural application was to a path optimization problem, the travelling salesman problem (TSP) [8], [7]. The TSP is a very difficult NP-hard problem, has been studied and was found to be easily adaptable to the ant colony optimization metaphor. The Ant Colony System [7] was an important contribution by Dorigo and Gambardella towards solving the TSP using Swarm Intelligence paradigm.

2.1.1.1 Ant System

Dorigo and Gambardella [7] proposed algorithms based on the ant colony metaphor for solving the Travelling Salesman Problem. The underlying idea in this approach is to use a *positive feedback* mechanism, similar to the ants in nature which lay a pheromone trail which is followed by other ants. A "virtual pheromone" or "goodness value" allows good solutions or parts of a good solution to be reinforced. Care has to be taken to not reinforce some of the good solutions too much, so that they do not stagnate. To avoid this, negative reinforcement is built into the algorithm with a time scale, similar to the evaporation of the pheromone in the case of foraging ants. This time scale should be long enough for cooperative behavior between the ants to emerge. Ants that explore good paths and perform

well influence ants that are sent out later more than the ants not performing so well.

The Traveling Salesman Problem: In the TSP [8], the goal is to find a closed tour of minimal length connecting n given cities. Each city must be visited once and only once. Let d_{ij} be the distance between cities i and j.

The problem can then be defined on a graph (N, E) where the cities are the nodes N and the connections between the cities are the edges between the set of nodes E. The distance matrix need not be symmetric.

In the Ant System algorithm, ants build solutions to the TSP by moving on the problem graph from one city to another until they complete a tour. During an iteration of the AS algorithm each ant k, $k=1,\ldots,m$, builds a tour executing $n = \|N\|$ steps in which a probabilistic transition rule is applied. Iterations are indexed by t, $1 \leq t \leq t_{max}$, where t_{max} is the maximum number of iterations allowed.

Each ant goes from city i to city j depending on:

- Whether or not the city has already been visited. Each ant carries a *memory* or *tabu list*. This grows with a tour and is then emptied at the end of each tour. The memory is used to define the set J_i^k of cities that the ant still has to visit when it is on city i. At the beginning, J_i^k contains all cities except the originating city. Visiting the same city more than once is avoided using the memory.

- The inverse of the distance $\eta_{ij} = \frac{1}{d_{ij}}$, is also called visibility. Visibility is based on strictly local information and represents the *desirability* of choosing city j when in city i. It is used to direct the ants' search.

- The amount of virtual *pheromone trail* τ_{ij} on the edge connecting city i to city j. Pheromone trail is updated on-line and is used to represent the *learned desirability* of choosing city j when in city i. Thus, the pheromone trail is emerging global information. The pheromone is adjusted during problem solution to reflect the experience acquired by the ants while problem solving as feedback.

The *transition rule*, that is, the probability for ant k to go from city i to city j while building its t^{th} tour is called the *random proportional transition rule* and is given by:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\Sigma_{l \epsilon J_i^k} [\tau_{il}(t)]^\alpha [\eta_{il}(t)]^\beta} \tag{2.1}$$

α and β are two adjustable parameters that control the relative weight of pheromone trail $\tau_{i,j}(t)$ and visibility $\eta_{i,j}$. If $\alpha = 0$, the closest cities are more likely to be selected. If $\beta = 0$, then the pheromone gets reinforced rapidly. This will lead quickly to optimal routes.

After the completion of a tour, each ant k lays a quantity of pheromone $\Delta\tau_{ij}^k(t)$ on each edge (i, j) that it has used depending on how well the ant has performed. At iteration t (an iteration is over when each ant has completed a tour), ant k lays $\Delta\tau_{ij}^k(t)$ on edge (i, j):

$$\Delta\tau_{ij}^{k}(t) = Q/L^{k}(t)\ if\ (i,j)\epsilon T^{k}(t), \tag{2.2}$$

$$\Delta\tau_{ij}^{k}(t) = 0,\ otherwise \tag{2.3}$$

where $T^k(t)$ is the tour done by ant k at iteration t, $L^k(t)$ is its length, and Q is a parameter (chosen to be of the order of magnitude of the optimal tour length by running a simple heuristic like the nearest neighbor heuristic).

For good performance of this method, the pheromone needs to decay, or all the ants end up taking the same route eventually with the solution possibly converging to a non-optimal. Pheromone decay is introduced by introducing a parameter of decay ρ, $0 \leq \rho \leq 1$. The resulting pheromone update rule is then given by:

$$\tau_{ij}(t) \leftarrow (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{2.4}$$

where $\Delta\tau_{ij}(t) = \Sigma_{k=1}^{m}\Delta\tau_{ij}^{k}(t)$, and m is the number of ants. The initial amount of pheromone is a small quantity, τ_0.

The total number of ants is an important parameter which influences the quality of the solution produced significantly.

Table 2.1.1.1 shows a performance comparison in terms of the number of iterations of the Ant System heuristic with Tabu Search and Simulated Annealing performed in [7]. AS performs as well as Tabu search and significantly better than Simulated Annealing.

A number of experiments by Dorigo and DiCaro [7] showed that the population

	Best tour	Average	Std. Dev.
Ant System for TSP	420	420.4	1.3
Tabu Search	420	420.6	1.5
Simulated Annealing	422	459.8	25.1

Table 2.1: A performance comparison of Ant Systems against Tabu Search and Simulated Annealing for the TSP

of solutions generated by AS does not converge to a single common solution. The algorithm produces a new, possibly improved solution with each iteration. This is interesting and promising for dynamical problems because it *tends to avoid the algorithm getting trapped in local optima.*

2.1.2 Applications of Ant Algorithms to Communication Networks

A desirable feature of Ant algorithms is that they may allow for increased efficiencies in the case when the problem under consideration involves spatial distribution and time-varying parameters. Thus, one of the first problems to be investigated for applicability was the routing problem in communicatin networks. This problem is both dynamic and distributed.

The problem to be solved by an ant routing algorithm is to direct traffic from sources to destinations maximizing performance while minimizing costs.

	Avg. call failures	Std. Dev.
Shortest path	12.57	2.16
Mobile agents	9.19	0.78
Improved mobile agents	4.22	0.77
ABC	1.79	0.54

Table 2.2: A performance comparison of Ant-Based Control with standard algorithms for call routing in Telecommunication networks

2.1.2.1 Ant Based Control for Telephone networks

Schoonderwooerd *et al.* [10] proposed an adaptive routing algorithm based on the use of many simple agents, called ants, that 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 network resources available. A performance comparison of Ant Based Control (ABC) performed well as compared to existing routing algorithms for telecommunication networks [10].

2.1.2.2 AntNet: Routing in Communication Networks

DiCaro and Dorigo [6] introduced an adaptive routing algorithm based on ant colonies that explore the network with the goal of building routing tables and

keeping them adapted to traffic conditions.

Some unique aspects of this work are as follows:

- The algorithm is designed for operation in both connection less and connection-oriented networks.
- Ants collect information which is used to build local parametric models of the network status. These models are used to compute reinforcements to change the probabilistic routing tables.
- Most importantly, the algorithm has been tested on a packet-switching network model [6] and was found to perform well against well-known exisiting algorithms in terms of throughput and delivery delays.

Since this work is an important inspiration for our work, we discuss this in detail here.

Consider a network with N nodes where each node i is characterized by its number of neighbors k_i and by a routing table $R_i = [r^i_{n,d}(t)]_{k_i,N-1}$. The entry $r^i_{n,d}(t)$ in the routing table of node i represents the probability at time t of choosing node n (which is a neighbor of node i) as the next node to move to when the destination is node d. $r^i_{n,d}(t)$ is then the *desirability* of going through n when trying to reach d from i. There are two data structures at each node:

- The routing table as described above and,

- A set of estimates for trip times from the current node i to all other nodes d. This entry consists of the average trip times $\mu_{i\to d}$ to go from node i to d and the associated variances $\sigma^2_{i\to d}$. This table of entries is denoted by $\Gamma_i = \{\mu_{i\to d}, \sigma^2_{i\to d}\}$. Γ_i is akin to a local picture of the global network's status at node i.

Further, the algorithm uses two types of agents (or ants) to collect and disseminate information:

- *Forward ants* $F_{s\to d}$, that go from source node s to destination d. They travel at the same priority level as data packets, therefore they experience the same traffic loads collecting information about the state of the network.

- *Backward ants* $B_{d\to s}$ that go from destination nodes to source nodes. They have a higher priority than both forward ants and data packets as they are required to backpropagate information as soon as possible.

The algorithm can be described as follows:

- Forward ants $F_{s\to d}$ are launched at regular intervals from every node s to a randomly selected destination d.

- Each forward ant selects the next hop among the not yet visited neighbor nodes following a random scheme. The probability of choosing a node is made proportional to the *desirability* of each neighbor node.

- The identifier of each visited node i and the time elapsed from its launching time to its arrival at this ith node are pushed into a memory stack $S_{s \to d}$ carried by the forward ant.

- If an ant visits a node it has already visited, a cycle has occured. The ant's stack is destroyed and it is not allowed to contribute to the pool of knowledge.

- When the ant $F_{s \to d}$ reaches the destination node d, it generates a backward ant $B_{d \to s}$. The stack of the forward ant is transferred to the backward ant.

- The backward ant takes the same path as that of the corresponding forward ant in the opposite direction. At each node i along the path it pops its stack $S_{s \to d}$ to know the next hop node. When it arrives at a node i, the backward ant updates the routing table R_i and the table Γ_i:

  - Γ_i (the local statistics at intermediate node i) is updated by using the values stored in the stack $S_{s \to d}(t)$. The times elapsed in arriving at every node f on the path $i \to d$ are used to update the means and variances at the node. An important point to be noted here is that only the subpaths that have a smaller trip time than the existing mean in the model are used to update Γ_i, while all the trip times are used at the source node s. Poor trip times are not used since they provide an incorrect estimate of the time required to reach a certain destination or intermediate node. Using trip times at node i obtained by ants routed

between source s and destination d allows the maintenance of a local picture at node i for destination d at no extra cost.

– The routing table R_i is changed by incrementing the probability $r^i_{i-1,d}(t)$ associated with the neighbor node $i-1$ and the destination node d, and decreasing by normalization the probabilities $r^i_{n,d}(t)$ associated with the other neighbor nodes, $n, n \neq i-1$, for the same destination.The proabability $r^i_{i-1,d}(t)$is increased by the reinforcement value as follows:

$$r^i_{i-1,d}(t+1) = r^i_{i-1,d}(t) + r \cdot (1 - r^i_{i-1,d}(t)) = r^i_{i-1,d}(t) \cdot (1-r) + r \quad (2.5)$$

Thus, the probability $r^i_{n,d}(t)$ will be increased by a value proportional to the reinforcement received and to the previous value of the node probability (small values are increased proportionally more).

The probabilities $r^i_{n,d}(t)$ for destination d of the other neighboring nodes n to node i receive a negative reinforcement,

$$r^i_{n,d}(t+1) = r^i_{n,d}(t) \cdot (1-r), \ n \neq i-1. \quad (2.6)$$

Every discovered path receives positive reinforcement. Thus, not only the assigned value r plays a role, but also the ant's arrival rate plays a role. This policy rewards paths that receive either high reinforcements independent of their frequency or low and frequent reinforcements.

The algorithm is first run on a network with no traffic for a limited period of time before it starts to control the network to build the routing tables and initial paths.

Routing data packets: Each node holds a buffer in which incoming and outgoing packets are stored with a different buffer for each outgoing link. Packets at the same priority level are served on a first-in-first-out basis and high-priority ants (backward ants) are served first. It is important to note that not only forward ants, but also *data packets are routed probabilistically.* This has been shown to improve the performance of AntNet [6].

Chapter 3

Routing Algorithms for Mobile Ad Hoc Networks

A number of routing protocols have been proposed for operation in Mobile, ad hoc networks. Most of these can be classified into one of the following categories:

- *Reactive protocols*; eg., AODV, DSR
- *Proactive protocols*; e.g., DSDV
- *Hybrid protocols*; e.g., ZRP
- *Location-based protocols*; e.g., LAR

We briefly discuss some of these protocols here, in particular, AODV, DSR and DSDV, which have recently become quite popular. Further, other protocols have also been proposed based on Hierarchical, Geographical and Power Aware schemes. Broch *et al.* [2] and [27] present in-depth performance analysis of common routing protocols AODV, DSR, TORA and DSDV.

3.1 Reactive protocols

Reactive, or on-demand, protocols have separate route discovery and maintenance processes. The discovery process is only triggered when a route is needed for a destination. The maintenance process is started when a node notices a link that has gone down. Broken routes are then deleted and a new discovery process is initiated. Reactive protocols therefore attempt to minimize the routing overhead required and in the process may face substantial delays for transmitting data packets.

3.1.1 Dynamic Source Routing

Source routing [14] refers to the technique where the packet itself contains the complete sequence of nodes that it is to traverse. The key advantage of source routing is that intermediate nodes do not need to maintain up to date routing information to forward the packets that they need to route. The Dynamic Source Routing protocol (DSR) [14], [2] uses two mechanisms - route discovery and route caching for routing.

Route discovery allows any node in the ad hoc network to dynamically discover a route to any other node in the network, whether directly reachable within the wireless transmission range or reachable through one or more intermediate hops through other nodes. A node initiating a route discovery broadcasts a route request packet which may be received by those nodes within its transmission range. The route request identifies the node, referred to as the target of the route discovery,

for which the route is requested. If the route discovery is successful, the initiating node receives a route reply packet listing a sequence of network hops through which it may reach the target. In addition to the address of the original initiator of the request and the target of the request, each route request packet contains a route record that accumulates a record of the sequence of hops taken by the route request as it is propagated through the ad hoc network during this route discovery. Each route request packet also contains a unique request id, set by the initiator from a locally maintained sequence number. In order to detect duplicate routes, each node in the network maintains a list of the $< initiator\ address,\ request\ id >$ ($< IA, RREQID >$) pairs that it has received on any route request.

When any node receives a route request packet, it goes through the following steps:

1. If the pair $< IA, RREQ\ ID >$ for its route is found in the nodes lists of recently seen requests, it discards the route request packet.

2. Otherwise, the packet is checked to see if the node is listed in its route record. If it is, the packet is discarded.

3. Otherwise, the node checks to see if it is the target. If it is, it sends a route reply packet to the initiator.

4. Otherwise, it appends its address and re-broadcasts the request.

Discarding already received request packets and request packets in which the

node is already contained, but is not the target, is intended to enable protection against loops. Route reversal is the preferred mode of returning a discovered route to the source node. However, an attempt to support unidirectional links is to allow for a route discovery of a specific type to be made by the destination node when a simple route reversal is not possible [14].

Route maintenance in DSR is built around caching discovered routes. There is no periodic checking of routes in DSR. This makes it very light on complexity, but it will definitely need enhancement if there are elements of QoS support that require some knowledge of the network topology.

3.1.2 Ad Hoc On-Demand Distance Vector (AODV)

The Ad Hoc On-Demand Distance Vector algorithm is designed to enable dynamic, self-starting, multi-hop routing. AODV is a combination of the Dynamic Source Routing (DSR) and Destination-Sequenced Distance Vector (DSDV) algorithms. It combines the Route Discovery and Route Maintenance mechanisms of DSR with the hop-by-hop routing of DSDV. As long as the endpoints in the communication connection have valid routes to one another, AODV does not play a role. It is intended for an ad hoc network whose links are frequently changing. As such, it can be termed to be a reactive protocol which works on demand. There are three message types defined by AODV for its operations:

- Route Request

- Route Reply
- Multicast Route Activation

These are UDP/IP messages. A node broadcasts a Route Request (RREQ) when a new route to a destination is needed. This destination can include either a single node or a multicast group. The RREQ message contains a Hop Count, which is the number of hops from the sources IP Address to the node currently handling the request. It also carries Broadcast ID, a sequence number identifying the particular RREQ when taken in conjunction with the sources IP address. Sequence numbers are used to select the chosen route by allowing the most recent information to be used. An RREQ contains a Destination as well as a Source Sequence Number. Given the choice between two routes to a destination, the requesting node always selects the node with the greatest sequence number. Each node receiving an RREQ caches a route back to the source node. If the receiving node has already received an RREQ with the same Broadcast ID and destination, it silently discards it. However, if the RREQ is new to the receiving node, it rebroadcasts the RREQ from its interface using its own IP address in the IP header of the outgoing RREQ. If the receiving node does have a route to the destination, the Destination Sequence number for that route is compared to the Destination sequence number field of the incoming RREQ. If the Destination Sequence number held by the node is less than the Destination Sequence number of the RREQ, the node rebroadcasts the RREQ as if it did not have a route to the destination. On

the other hand, if the node does not have a route to the destination, and the nodes existing destination sequence number is greater than or equal to the destination sequence number of the RREQ, the node generates a Route Reply.

The Route Reply message (RREP) contains the Hop Count and the Lifetime, the Lifetime being the time for which the nodes receiving the RREP consider the route to be valid. A RREP is unicast back to the source node if the receiving node has a fresh route to a destination.

AODV allows for the formation of multicast groups whose membership is free to change during the lifetime of the network. AODV supports multicast communication consisting of multicast trees. These are trees containing all nodes that are members of the multicast group, and all nodes that are needed to connect to the multicast group members. When a node receives a Multicast Route Activation (MACT) announcing it as the next hop, it will send its own MACT announcing the node it has chosen as the next hop, and so one up the tree. This continues until a node that is a part of the multicast tree is traversed once more.

For route maintenance, AODV requires nodes to transmit periodic HELLO messages. Alternatively, AODV briefly specifies an option that allows for the use of only lower level methods (physical/link layer) for response detection. This variant has been branded as AODVLL (AODV Link Layer).

3.2 Proactive Protocols

Proactive protocols attempt to maintain routes to all nodes in the network at all times. Further, route information is exchanged periodically or at topological changes. This ensures that the latency is low when data traffic is required to be sent. However, this might lead to high consumption of bandwidth, leading to inefficient performance in terms of overhead, as well as high consumption of other network and computing resources.

3.2.1 Highly Dynamic Distance-Sequenced Distance Vector Routing Protocol

DSDV [16] is a hop-by-hop distance vector routing protocol that requires each node to periodically broadcast routing updates.

Each DSDV node maintains a routing table listing the next hop for each reachable destination. DSDV tags each route with a sequence number and considers a route R more favorable than R' if R has a greater sequence number, or if the two routes have the same sequence number but R has a lower metric. Each node in the network advertises a monotonically increasing sequence number for itself. When a node B decides that its route to a destination D is broken, it advertises the route to D with an infinite metric and sequence number 1 greater that the sequence number for the route that has just broken. This causes a node A routing packets through B to incorporate the infinite metric in its route calculations until

it receives and advertisement for a route to D with a higher sequence number.

Chapter 4

A swarm intelligence based unicast algorithm for MANETs

In this chapter, we describe a routing algorithm for Mobile Ad Hoc Networks based on the swarm intelligence paradigm and similar to the swarm intelligence algorithms described in [6] and [11]. 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, 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. The algorithm assumes the existence of the IP layer and is independent of the MAC layer.

4.1 The Operation of the algorithm

4.1.1 Initialization and beaconing

Initialization is carried out at each node by establishing the identities of the neighboring nodes. Each node sends out single-hop 'Hello' (beacon) broadcast messages with its ID. All the receiving nodes that get this message add the sending node to their neighbor list. This neighbor list is then used to build routing tables and to update routes when a change in the topology occurs. All links are assumed to be bidirectional. Therefore, if a node A has a node B in its list of neighbors, then node B also has A in its list of neighbors. Hello messages are sent out periodically with an interval of $HELLO\_INTERVAL$ seconds.

4.1.2 Bootstrapping of the routing tables

The initial bootstrapping of the routing tables is done at a node when the first forward ant is being sent out to a certain destination.

At this time, there are no routing table entries (i.e. no probabilities for next hops) for that particular source-destination pair. The creation of the first forward ant at a node for the source-destination pair causes the routing table entries to be 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 where the routing table is being established.

The uniform probabilities assigned to all the neighbors indicate that nothing is known about the state of the network. These probabilities are then adjusted by backward ants, when backward ants from the destination are received at the source node.

4.1.3 The Routing Table

The routing table at each node is organized on a per-destination basis and is of the form (*Destination*, *Next hop*, *Probability*). It contains the goodness values for a particular neighbor to be selected as the next hop for a particular destination. Further, each node also maintains a table of statistics for each destination d to which a forward ant has been previously sent, the mean and the variance, $(\mu_{sd}, \sigma^2_{sd})$ for the routes between source node s and destination node d.

The routing tables then contain the following data structures:

- The probability (goodness value) of taking a next hop f at a node n , P^n_{fd} to eventually reach a certain destination d. These values are used by the forward

ants when they are routed from node to node based on the probability value at each node.

- The mean and the variance, (μ_{nd}, σ_{nd}) at node n to reach destination d.

Consider the simple network topology depicted in figure 4.1. In this example,

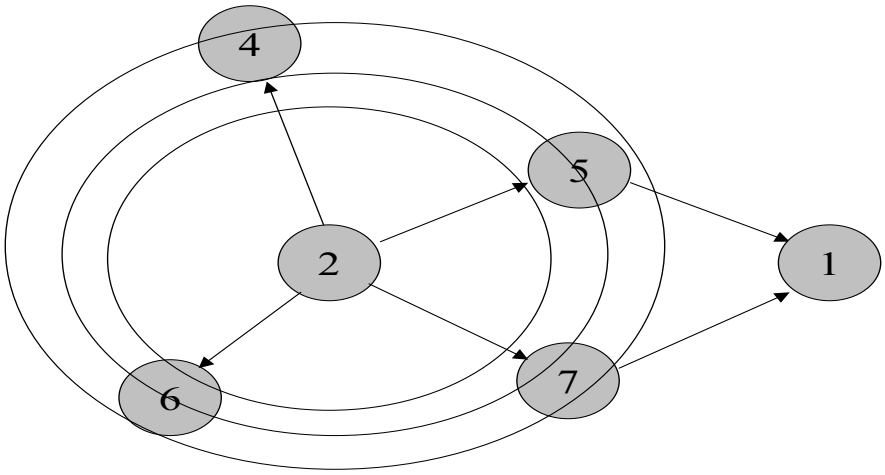

Figure 4.1: An example of topology with varying connectivity

nodes 4, 5, 6 and 7 are within the transmission range of node 2. Node 1 can be reached through nodes 5 or 7. Then, the routing table probabilities at node 2 for destination 1 look as follows when initialized:

Here,

Destination	Node 1
Neighbors	$(\mu_{21}, \sigma_{21}^2)$
4	0.25
5	0.25
6	0.25
7	0.25

Table 4.1: Initialization of the routing table at node 1

$$\sum_{i=1}^{m} P_{ni}^{1} = 1, \tag{4.1}$$

where P_{ni}^{1} is the probability of taking node i as the next hop at the current node n for destination node 1 and μ_{21} and σ_{21}^2 are the mean and the variance at node 2 for node 1.

4.1.4 Forward Ants

Each node periodically sends forward ants to randomly chosen destination nodes throughout the network. At the time of creation of the agent, if a routing table entry is not present at the node for that particular destination, a routing table entry is created. This is also true of the forwarding of ants at intermediate nodes. Each forward ant packet contains the following fields:

- Source node IP address

- Destination node IP address
- Next hop IP address
- Stack
- Hop count

Hence, the next hop of the forward ant is determined at the sending node and the forward ant is sent in unicast fashion. That is, though the forward ant is received at all the neighboring nodes, it is accepted (at the MAC layer) only by the node to which it has been addressed.

The stack of the forward ant is a dynamically growing data structure that contains the IP addresses of the nodes that the forward ant has traversed as well as the time at which the forward ant reached these nodes.

Forward ants are routed on normal priority queues, that is, 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.

4.1.5 Routing the Forward Ants

The forward ant is routed at each node according to the per-destination probabilities for the next hop in the routing table at the current node. Thus, the forwarding

of the forward ant is purely probabilistic and allows exploration of paths available in the network. As such, these agents are forwarded according to the current network status and hence take the network status in to account.

These agents are henceforth referred to as *Regular Ants*, similar to [11] to distinguish them from *Forward Uniform Ants* which are described underneath. As mentioned earlier, the forward ants (both regular and uniform) maintain a stack containing the IP addresses of the nodes that they have been through. When a forward ant is received at a node, it checks to see if it has previously traversed the node. If it has not previously traversed the node, the IP address of the node and the current time are pushed into the stack of the ant. In case the node IP address is found in its existing stack, the forward ant has gone into a loop. As such, the forward ant is not allowed to contribute to the pool of information regarding the state of the network and is destroyed.

4.1.6 Uniform Ants

Since forward regular ants are routed uniformly, and the resulting backward ants reinforce the routes, this can lead to a saturation of the probabilities, that is the probabilities of one (observed to be the best) route go to 1 and the probability of the other routes goes to 0. As a result, new routes never get discovered. In a dynamic situation with the possibility of breakage of links and mobility of nodes, this means that the algorithm is unable to adapt to changes in the network.

To make the algorithm fully adaptive to mobility and topology changes, we introduce another set of ants, called uniform ants. These are similar to the agents proposed in [11]. X % of the time, instead of creating regular ants, each node sends out uniform ants. These are created in the same manner as regular ants, however they are routed differently. Instead of using the routing tables at each node, they choose the next hop with uniform probability. If the current node has n neighbors, then the probability of taking a neighbor as the next hop is $1/n$. This is in contrast to regular ants which prefer taking next hops with higher probabilities more often.

Uniform ants explore and quickly reinforce newly discovered paths in the network. Further, they ensure that previously discovered paths do not get saturated.

4.1.7 Backward Ants

When a regular or uniform ant reaches its destination, it generates a 'Backward Ant'. The backward ant inherits the stack contained in the forward ant. The forward ant is de-allocated. The backward ant is sent out on the high priority queues. This ensures that backward ants are propagated in the network quickly, so that they can update the information regarding the state of the network without delay.

The purpose of the backward ant, as already mentioned, is to propagate information regarding the state of the network gathered by the forward ants. The backward ant retraces the path of the forward ant by popping the stack, making

modifications in the routing tables and statistic tables at each intermediate node according to the following learning rules:

1.

$$P_{fd} \leftarrow P_{fd} + r(1 - P_{fd}) \tag{4.2}$$

$$P_{nd} \leftarrow P_{nd} - rP_{nd} \tag{4.3}$$

Probability P_{fd} is the probability of choosing neighbor f when the destination is d and decrementing by normalization the other probabilities P_{nd} , which is the probability of choosing neighbor n when the destination is d.

r is the reinforcement parameter.

2.

$$P_{fd} \leftarrow \frac{(P_{fd} + r)}{(1 + r} \tag{4.4}$$

$$P_{nd} \leftarrow \frac{P_{nd}}{(1 + r)} \tag{4.5}$$

Once again, probability P_{fd} is the probability of choosing neighbor f when the destination is d and decrementing by normalization the other probabilities P_{nd} , which is the probability of choosing neighbor n when the destination is d.

In both the above cases, the reinforcement parameter, r can defined as a function of some metric or a combination of metrics, e.g. delay or the number of hops.

$$r = \frac{k}{f(c)}, k > 0 \; and \; f(c) \; is \; monotone \; function \; of \; the \; metric \tag{4.6}$$

The backward ant also updates the existing estimates of the forward trip time at the source node as well as intermediate nodes. The trip time of this backward ant is used to update the statistics.

The mean and the variance, $(\mu_{kd}, \sigma^2_{kd})$ are updated using the following update rules:

$$\mu_{kd} \leftarrow \mu_{kd} + \eta(o_{k \rightarrow d} - \mu_{kd}) \tag{4.7}$$

Where μ_{kd} is the mean of the ant trip times at the current node, k, to the destination node, d. η is a constant, $o_{k \rightarrow d}$ is the trip time of the ant from the current node k to the destination node d.

and

$$\sigma^2_{kd} \leftarrow \sigma^2_{kd} + \eta((o_{k \rightarrow d} - \mu_{kd})^2 - \sigma^2_{kd}) \tag{4.8}$$

Where σ^2_{kd} is the variance of the ant trip times at the current node, k, to the destination node, d. η, $o_{k \rightarrow d}$ and μ_{kd} are the same as above.

4.1.8 Changes in routing tables due to node mobility

When a node or nodes enter into the transmission range of a node, this creates the possibility of there being new available routes to a destination that was either reachable by a longer route or previously unreachable. The detection of a newly moved node is through the beaconing mechanism. The Hello messages broadcast by each node give information regarding the availability of a node as a next hop. Suppose a node A moves into the neighborhood of a node B, then the IP address of A is added to the list of neighbors of B and vice versa. Node B then adjusts its routing table to include A with a small probability. So, if node B has existing routes to nodes $d_1, d_2, ..., d_m$, it adds A as a next hop with a small probability for each of $d_1, d_2, ..., d_m$. Thus, the probability of a forward ant taking A is,

$$p_A = \delta, \delta \ll 1 \tag{4.9}$$

and the probability of the other nodes, n, $n \neq A$ becomes,

$$p_i = p_i - \frac{\delta}{N} \tag{4.10}$$

where N was the number of B's neighbors before A entered its transmission range. The sum of the probabilities remains 1. i.e. $\Sigma_i p_i = 1$.

This allows the exploration of new routes through node A from node B by regular ants. If new and efficient routes are found with node A as an intermediate node, they are quickly reinforced and can be utilized for routing data packets.

When a node A leaves the transmission range of a node B, A is removed from B's routing table. That is, for all destinations for which B has routing probabilities, A is removed from the routing table, The probability of taking A as the next hop is made 0 for all destinations from node B.

$$p_i = p_i + \frac{p_A}{N} \tag{4.11}$$

where N is the number of neighbors of B after A leaves its transmission range, and,

$$p_A = 0 \tag{4.12}$$

The probability distribution is then normalized for all the other nodes, so that the sum of the probabilities is 1, i.e. $\sum_i p_i = 1$.

4.1.9 Routing Data packets

Data packets are routed based on the maximum probability at each intermediate node from the source node to the destination node. As such, local information (next hop probability at an intermediate node) is used in such a way that global information (a complete route between the source and the destination) emerges from it.

4.2 Algorithm parameters and other issues

The unicast routing algorithm proposed here has three key parameters:

- *The rate at which forward regular and uniform ants are sent.* The frequency with which forward ants are sent determines how quickly the algorithm discovers new paths, reinforces existing paths, destroys unavailable paths and adapts to new network topologies. While sending out too few ants does not allow the algorithm to adapt to network changes, sending out a large number of ants creates substantial overhead, a particularly difficult issue in the MANET environment, where there might be severe constraints on bandwidth resources.

- *The percentage of forward and uniform ants.* Forward ants mainly reinforce the existing, previously discovered routes. Uniform ants, on the other hand, discover new previously unavailable routes. Regular ants 'reinforce', while uniform ants 'explore'. Hence, the percentage of uniform ants is an important parameter. If no uniform ants are sent, the probabilities converge and the routes are saturated, leaving no scope for exploration and adaptation.

  On the other hand, if too many uniform ants are sent, the probability values oscillate to a large extent, so that the routing tables do not reflect the network status accurately.

- *The update function and reinforcement* that are used by the backward ants to

reinforce the routing probabilities at each node. These parameters together determine how quickly the algorithm adapts to changes in the network topology and connectivity.

4.2.1 Broadcast vs Unicast in the MANET environment

Wireless data transmission offers several challenges and opportunities for routing. One of the advantages it offers is the inherent broadcast capability, that is, all the neighbors of a node receive (at the MAC layer) each data packet transmitted by the node. However, the algorithm proposed above does not take advantage of this inherent capability of the wireless environment, and though it is well suited for the wired environment, we note that this algorithm leads to high overhead inefficient route discovery in the wireless environment. As the number of nodes increases, the number of ants required to find a path to the destination also increases rapidly, leading to very high overhead, high delays as well as high packet losses.

Figure 4.2 shows a comparison of the goodput required for AODV and the unicast algorithm described above. Even for a low mobility speed of 1 m/s, the overhead required is very high for the unicast algorithm.

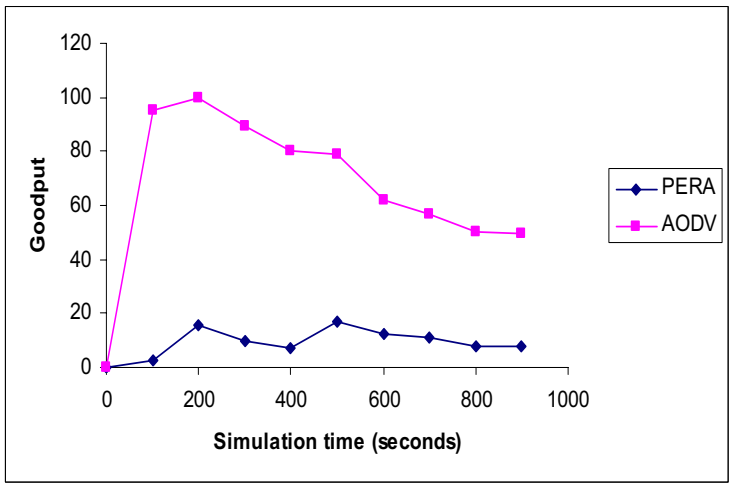

Figure 4.2: Goodput comparison of AODV and the unicast swarm based algorithm for 20 nodes in an area of 500m X 500m for speed of 1 m/s.

Chapter 5

The Probabilistic Emergent Routing Algorithm

5.1 The Probabilistic Emergent Routing Algorithm

This chapter presents an algorithm for routing in MANETs inspired by the *Swarm Intelligence* (SI) paradigm. A routing scheme is proposed that exhibits emergent behavior. This behavior is based on a type of learning algorithm, similar to one described in AntNet, that provides deterministic forwarding of message packets by creating multiple routes at source nodes for destination nodes. This provides a natural resistance to the effects of corruption of routing table entries. As such, the algorithm exhibits robustness.

Route discovery in the algorithm is done by two kinds of agents or ants - forward and backward. These agents create and adjust a probability distribution at each node for the node's neighbors. The agent packets, or *Ants* are of a relatively small (variable) size. The probability associated with a neighbor reflects the relative likelihood of that neighbor forwarding and eventually delivering the packet. Further, multiple routes between the source and the destination are created.

Our algorithm assumes bidirectional links in the network and that all the nodes in the network fully cooperate in the operation of the algorithm. The algorithm assumes the existence of the IP layer and is independent of the MAC layer.

5.1.1 Initialization and beaconing

Initialization is carried out at each node by establishing the identities of the neighboring nodes. Each node sends out single-hop 'Hello' (beacon) broadcast messages with its ID. All the receiving nodes that get this message add the sending node to their neighbor list. This neighbor list is then used to build routing tables and to update routes when a change in the topology occurs. All links are assumed to be bidirectional. Therefore, if a node A has a node B in its list of neighbors, then node B also has A in its list of neighbors.

Beaconing is thus used to signify the presence of neighboring nodes and to indicate their *spatial, temporal, connection,* and *signal stability.* Further, it is used to make updates to routing tables and maintain their accuracy.

Hello messages are sent out periodically with an interval of $HELLO\_INTERVAL$ seconds. A good value of $HELLO\_INTERVAL$, derived through extensive simulations has been found to be 1 second, [2] and [27].

Some MAC protocols like MACAW and 802.11 [26] provide explicit link layer acknowledgements. Several refined versions of protocols such as AODV and DSR use this MAC layer feedback for maintaining routing tables accurately. This pro-

duces efficiencies in terms of the number of control packets sent per data packet. Our algorithm, however, does not use explicit MAC layer RTS/CTS signalling mechanism currently.

5.1.2 Bootstrapping the routing tables

Routing tables at each node are of the form ($Destination$, $Nexthop$, $Probability$). Thus, at a node n, the probability of taking node f as the next hop to route a data packet to destination d is stored as P^n_{fd}. The routing table at a node for a particular destination is established when a backward ant is routed to the node from the destination. Hence, the node at which the routing table is being created is either the source of the forward ant (and the destination of the backward ant) or an intermediate node along the path used by a forward ant to reach the destination node from the source node.

Hence, the routing table is initialized when it is known by the current node that a path to the destination d exists by taking a certain next hop f and that this path has been previously discovered by a forward ant. The initialization of the routing table is done by incorporating all the neighbors of node n in the routing table. Each node is assigned an initial probability $1/N$, where N is the number of neighbors of node n. The routing tables are then modified to give a higher probability to the node that the backward ant just came from, as discussed in section 6.1. This learning rule provides the node from which the backward ant just

arrived a higher probability than the other neighbors, establishing a path toward the destination.

Further, each node also maintains the structure $(\mu_{nd}, \sigma^2_{nd})$ for each destination, that is, the mean μ_{nd} and the variance σ^2_{nd} of the delays of the routes from node n to destination d as perceived by the ants that have traversed a route to the destination. On the receipt of the first backward ant, the value of the time taken by the ant to travel to the destination from the current node, $T_{n \to d}$ is assigned to the mean, μ_{nd} and the variance, σ^2_{nd} is assigned a value of zero. Modifications to $(\mu_{nd}, \sigma^2_{nd})$ are made upon the arrival of later backward ants based on the learning rule as discussed in section 6.1.

The routing table and the table of local statistics at each node can be visualized as in figure 5.1.

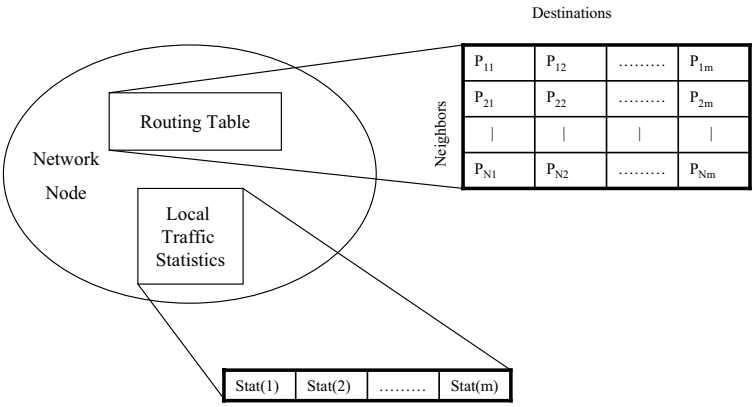

Figure 5.1: The routing table and local statistics table maintained at each node.

5.1.3 Forward Ants

To carry out the process of Route Discovery, Forward Ants or agents are used. The Forward Ants are somewhat similar to the Route Request packets used by AODV [18] and DSR [14] routing protocols, but have some subtle differences.

A forward ant is sent from a source node to a destination node to find a set of routes between the two nodes. Further, forward ants collect information about these routes. The first forward ant is created at a node s when the node has a data packet to send to another node d for which it has no record of an existing route. Forward ants are then sent to the destination periodically to maintain and optimize available routes, as well as discover new routes. As mentioned in section 5.1.2, the node s would not have a record of a route to destination d until a backward ant has passed through it or if it was the destination of a backward ant generated at node d.

Each forward ant contains the IP address of its source node, the IP address of the destination node, a sequence number, a hop count field and a dynamically growing stack. The structure of the forward ant packet is as shown in 5.2.

The stack contains information about the nodes that the forward ant traverses and the times at which these nodes have been traversed, ie. ($NODE\_ID$, $NODE\_TRAVERSAL\_TIME$). Thus, each element of the stack contains the IP address of a node and the time at which the forward ant is received at the node. The hop count field is set to the network diameter (defined as the minimum

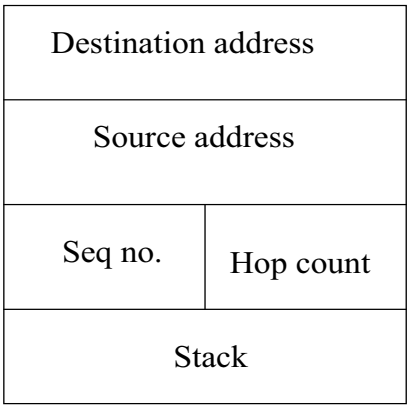

Figure 5.2: The structure of forward and backward ant packets

number of hops required to move from any node in the network to any other node) and it is decremented each time the forward ant is received at an intermediate node.

When a node does not have a record of a route to a destination to which it has to send a packet, it creates a forward ant and broadcasts it to all its neighbors. Before broadcasting the forward ant, the node pushes its own IP address on to the stack of the forward ant as well as the time at which the ant is created. Henceforth, the node keeps sending forward ants periodically to the destination for as long as a route is required.

When a node receives a forward ant, it checks in the destination IP address field if the address corresponds to its own IP address. If the forward ant is not directed to the current node, the node pushes its own IP address and the time

at which the ant was received at the node. Also, the hop count field of the forward ant is decremented by 1. Each forward ant is uniquely identified by the values of its source node IP address and the sequence number, i.e. the record $(Source\ IP\ address, Sequence\ Number)$. The sequence number for each ant is assigned at the source node and is unique for that source and forward ant. Thus, each node stores the pair $(Source\ IP\ address, Max\ Sequence\ number)$, where the $Max\ Sequence\ number$ is the highest value of the sequence number of an ant received from that source node. The node drops forward ants with a sequence number less than or equal to the $Max\ Sequence\ number$ that it receives from the same previous hop. This avoids the duplication of forward ants in the network. If the $Max\ Sequence\ number$ value is greater than that previously recorded by the node, the node updates this value.

An ant which reaches a node that it has already traversed is destroyed in similar fashion. It has taken a circuitous route and is therefore not allowed to contribute to the store of information regarding the status of the network.

It is important to note that the forward ants travel on the same queues as data packets. In our experiments, these queues are modeled as FIFO queues. Hence, the forward ants experience the same delay and congestion as the data packets when the metric being used is delay. This allows us to reinforce certain routes more than other routes depending on the current network status as perceived by the forward ants.

When a forward ant reaches the node that is its intended destination, the node

extracts all the relevant information from the forward ant. That is, the source address, the hop count and the stack. The forward ant is then 'killed', i.e. its memory is deallocated. The information obtained by the forward ant is then used by the node to create a backward ant.

It is important to note that since the the forward ant is broadcast at the source and intermediate nodes, each forward ant will cause the broadcast of multiple forward ants, several of which may find different paths to the destination, generating multiple backward ants *with the same source sequence number*.

Since forward ants are re-broadcast at every intermediate node, creating multiple forward ants, it can be seen that a forward ant broadcast from the source node may find more than one route to the destination, if more than one routes exist. In the case when the network is closely connected and the network diameter (defined as the minimum number of hops between any two nodes) is small, a single broadcast forward ant successfully finds several feasible paths to the destination node from the source node.

Further, the forward ant also collects information about each of these paths, that is, the number of hops on the path and the delay on the intermediate subroutes as well as on the entire route. It should be noted here that the Route Discovery phase is similar to that of existing MANET algorithms like AODV and DSR, in the sense that a flooding-based approach is used which uses the inherently broadcast medium of the wireless environment to its advantage. However, an important difference is that our algorithm discovers a set of routes (unlike AODV,

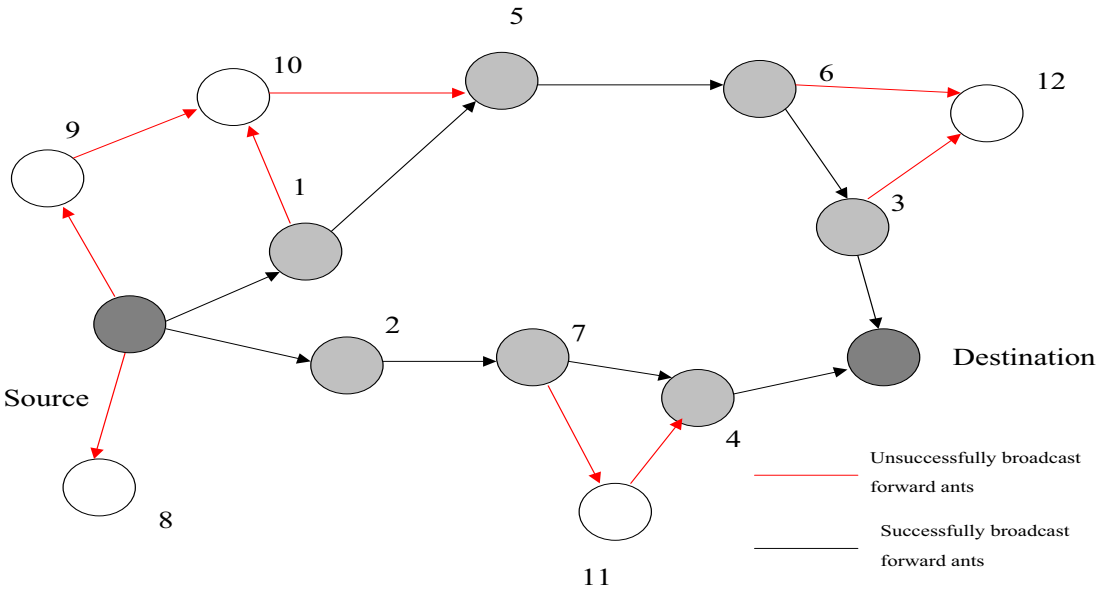

Figure 5.3: Each broadcast forward ant generates multiple forward ants and may find multiple paths to the destination.

for instance, which uses the first route that the Route Request packet establishes and discards others). Further, we obtain information about these paths and use this information as feedback to the system.

Figure 5.3 shows an example of forward ants propagating through the network and finding a route to a destination node. A forward ant $F^s_{s\to d}$ is created and broadcast from the source node s to find a route(s) to the destination d. At each neighbor of s, node k, the forward ant is rebroadcast as $F^k_{s\to d}$. In the figure, lightly shaded lines indicate transmitted forward ants that are dropped by the receiving node and the dark lines indicate forward ants that are accepted by the receiving node and help build routes. For instance, node 5 rejects the forward ant it receives

from node 10 as it has already received an ant with the same destination and sequence number from node 1.

Further, two alternate paths are found, $s \rightarrow 1 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow d$ and $s \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow d$. These paths are then graded by the corresponding backward ants, assigning a higher probability values to nodes on the path $s \rightarrow 2 \rightarrow 7 \rightarrow 4 \rightarrow d$, if the metric in consideration is hop count. The figure also shows forward ants that are received but have to be discarded as the receiving node has already received an ant with the same sequence number. For instance, node 5 drops the forward ant it receives from node 10 as it has already received a forward ant from node 1 with the same sequence number.

The forward ant that eventually reaches destination d via the route $s \rightarrow 1 \rightarrow 5 \rightarrow 6 \rightarrow 3 \rightarrow d$ then contains, as part of its stack, the IP addresses of nodes s, 1, 5, 6 and 3. The stack of the forward ant also contains the time at which the forward ant was created at node s as well as the times at which it traversed nodes 1, 5, 6 and 3. Further, this forward ant travels on the same queue as data packets between these nodes and therefore experiences the same set of delays. Hence, it has accurate information about the status of the network.

In summary, a node sends forward ants to a destination if it has a data packet to send to this destination and it has no record of an available route. If this node is the source of the data packet, it sends forward ants periodically for as long as the route is required. On the other hand, if this node is not the source of the data packet, it sends a forward ant exactly once.

5.1.4 Backward Ants

When a forward ant reaches the destination node that it is intended for, the destination node creates a new agent, a Backward Ant. The purpose of the backward ant is to retrace the path of the corresponding forward ant that triggered its creation. It uses the information contained in the forward ant on the reverse path to change the probability distribution at each node and update the routing tables to reflect the current status of the network more accurately.

When a node receives a forward ant that is intended for it, the node creates a new agent, a backward ant. The IP address of the source node of this agent is the destination address of the backward ant and the current node is the source of the backward ant. The backward ant is similar to the forward ant, it contains the following fields:

- Destination IP address : The IP address of the source of the forward ant,
- Source IP address : The IP address of the current node, i.e. the node creating the backward ant,
- Hop count,
- The stack of the forward ant,
- The sequence number of the forward ant - this is not unique anymore for the set of backward ants.

The backward ant travels in *unicast* fashion back to the source node. It is forwarded on high priority queues, which are separate from the queues used for forward ants and data packets. This is important since the backward ants are required to spread information about the current state of the network as quickly as possible. The stack of the forward ant is used to route it. Using the address at the top of the stack, the node forwards the backward ant to the correct next hop.

Suppose that a forward ant from source node s is received at node d. Node d generates a backward ant. When the backward ant is received at the next hop(also the penultimate hop of the corresponding forward ant), node f, the stack of the backward ant is popped once. The resulting information is the following:

- The IP address of the current node f,
- The $NODE\_TIME$, the time at which the corresponding forward ant was received at node f.
- The time at which the backward ant was created at its source node d, ANT_TIME. Then, the time taken to reach the destination of the forward ant from the current node is the difference $ANT\_TIME - NODE\_TIME$,
- The number of hops from the current node f to the destination d are calculated by subtracting the value in the hop count field from the network diameter.

These values are used to update the routing and local statistics tables at the

intermediate nodes f.

If routing table entries for destination d do not exist at node f, new ones are created with the neighbor list of the node f. All the neighboring nodes are given a probability of $1/N$, where N is the number of neighbors of the node f. The routing tables are then readjusted according to the probability rules discussed underneath.

If routing table entries for d already exist at node f, they are updated so as to increase the probability (goodness, preference) of taking as the next hop, the node from which the backward ant has just been received, node f to reach the destination d.

The following update rules may be used to adjust the probabilities at the intermediate nodes:

1.

$$P_{fd} \leftarrow P_{fd} + r(1 - P_{fd}) \tag{5.1}$$

$$P_{nd} \leftarrow P_{nd} - rP_{nd} \tag{5.2}$$

Probability P_{fd} is the probability of choosing neighbor f when the destination is d and decrementing by normalization the other probabilities P_{nd} , which is the probability of choosing neighbor n when the destination is d.

r is the reinforcement parameter.

2.

$$P_{fd} \leftarrow \frac{(P_{fd} + r)}{(1 + r)} \tag{5.3}$$

$$P_{nd} \leftarrow \frac{P_{nd}}{(1 + r)} \tag{5.4}$$

Once again, probability P_{fd} is the probability of choosing neighbor f when the destination is d and decrementing by normalization the other probabilities P_{nd} , which is the probability of choosing neighbor n when the destination is d.

In both cases above, the reinforcement parameter r can defined as a function of some metric or a combination of metrics, e.g. delay or the number of hops.

$$r = \frac{k}{f(c)},\ k > 0 \textit{ and } f(c) \textit{ is a monotone function of the metric.} \tag{5.5}$$

The backward ant also updates the existing estimates of the forward trip time at the source node as well as intermediate nodes. It is important to note that in this scheme, which utilizes broadcast flooding of forward ants, each forward ant potentially generates many backward ants. Suppose a forward ant $F^{i}_{s \to d}$, where i is the sequence number of the forward ant, generates a set of l backward ants, $B^{i,1}_{d \to s},\ B^{i,2}_{d \to s},\ .....,\ B^{i,l}_{d \to s}$. Then only the statistics of the backward ant, $B^{i,best}_{d \to s}$, such that

$$cost(B_{d\rightarrow s}^{i,best}) = min[cost(B_{d\rightarrow s}^{i,1}), cost(B_{d\rightarrow s}^{i,2}), ..., cost(B_{d\rightarrow s}^{i,n})] \tag{5.6}$$

are used to calculate the new mean and variance at the node k for the destination d. In the case where the optimization metric is delay, the first backward ant to reach its destination is the required ant. The trip time of this backward ant is used to update the statistics.

The mean and the variance, $(\mu_{kd}, \sigma_{kd}^2)$ are updated using the following update rules:

$$\mu_{kd} \leftarrow \mu_{kd} + \eta(o_{k\rightarrow d} - \mu_{kd}) \tag{5.7}$$

where μ_{kd} is the mean of the ant trip times at the current node, k, to the destination node, d. η is a constant, $o_{k\rightarrow d}$ is the trip time of the ant from the current node k to the destination ndoe d, and

$$\sigma_{kd}^2 \leftarrow \sigma_{kd}^2 + \eta((o_{k\rightarrow d} - \mu_{kd})^2 - \sigma_{kd}^2) \tag{5.8}$$

where σ_{kd}^2 is the variance of the ant trip times at the current node, k, to the destination node, d. η, $o_{k\rightarrow d}$ and μ_{kd} are the same as above.

The learning rules are an important factor in the efficient functioning of the algorithm and are dealt with in further detail in chapter 6.

To further illustrate the functioning of the algorithm for individual ants as well as individual nodes, figure 5.4 depicts the algorithm flow for each ant, while figure 5.5 depicts the algorithm flow ateach node.

Figure 5.6 presents the abstract pseudo code for the algorithm.

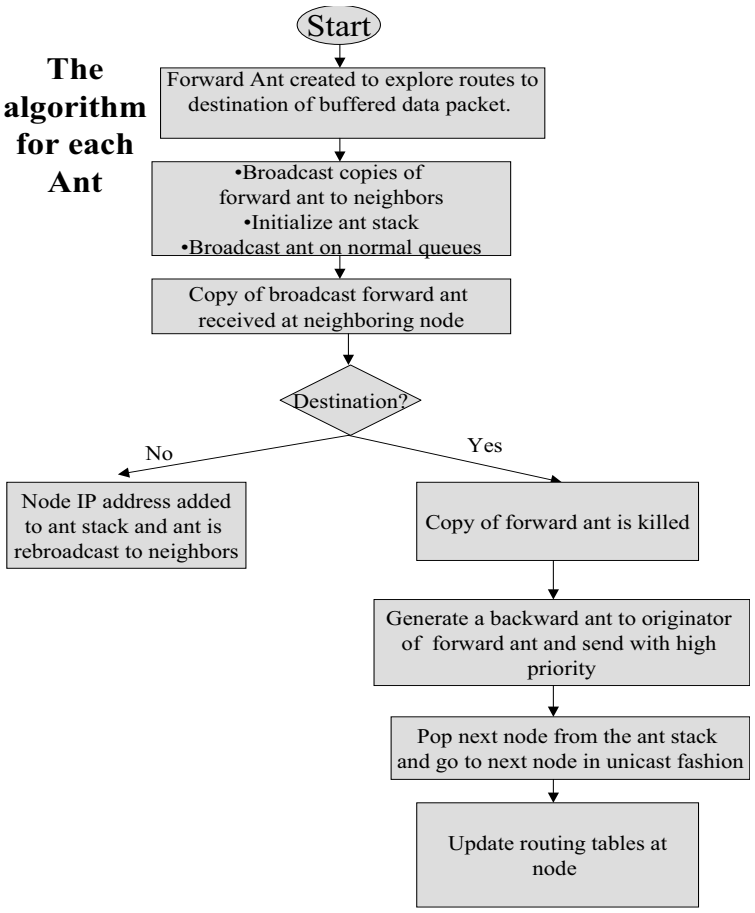

Figure 5.4: Algorithm for each ant.

5.1.5 Changes in routing tables triggered by node mobility

In a mobile, ad hoc network, the possibility of rapid movement of nodes creates high dynamicity and requires the algorithm to be highly adaptive to these changes. It is of crucial importance that the algorithm adapt quickly to changes in the network toppology, discovering new, efficient paths frequently and efficiently as

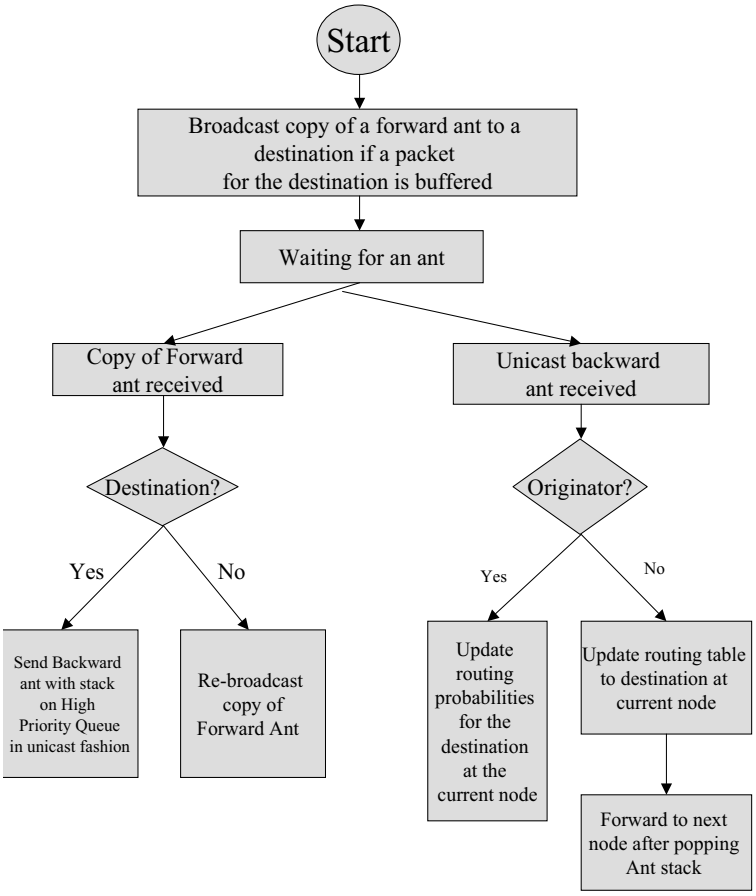

Figure 5.5: Algorithm at each node.

well as reacting to changes in the network topology when existing paths are no longer available.

More specifically, due to the movement of nodes, it is possible that new, more efficient paths might become available to route packets between the source and destination nodes. The algorithm should quickly adapt itself and adjust the routing tables accordingly so as to take advantage of such paths. On the other hand, existing routes may be frequently disrupted and/or become temporarily unavailable

due to node movement, loss of connectivity as well as factors such as terrain and other conditions. In such a situation, the algorithm should allow the nodes to quickly gather information regarding the loss of specific links and create new routes between source and destination nodes.

When a node or nodes enter into the transmission range of a node, this creates the possibility of there being available new routes to a destination that was either reachable by a longer route or previously unreachable. The detection of a newly moved node is through the beaconing mechanism. The Hello mesages broadcast by each node give information regarding the availability of a node as a next hop. Suppose a node A moves into the neighborhood of a node B, then the IP address of A is added to the list of neighbors of B and vice versa. Node B then adjusts its routing table to include A with a small probability. So, if node B has existing routes to nodes $d_1, d_2, ..., d_m$, it adds A as a next hop with a small probability for each of $d_1, d_2, ..., d_m$. Thus, the probability of taking node A as the next hop is,

$$p_A = \delta, \delta \ll 1 \tag{5.9}$$

and the probability of the other nodes, n, $n \neq A$ becomes,

$$p_i = p_i - \frac{\delta}{N} \tag{5.10}$$

where N was the number of B's neighbors before A entered its transmission range. The sum of the probabilities remains 1. i.e. $\Sigma_i p_i = 1$.

This allows the exploration of new routes through node A from node B. If

new and efficient routes are found with node A as an intermediate node, they are quickly reinforced and can be utilized for routing data packets.

When a node A leaves the transmission range of a node B, A is removed from B's routing table. That is, for all destinations for which B has routing probabilities, A is removed from the routing table. The probability of taking A as the next hop is made 0 for all destinations from node B. The probability distribution is then normalized for all the other nodes, so that the sum of the probabilities is 1, i.e. $\sum_i p_i = 1.$.

$$p_i = p_i + \frac{p_A}{N} \tag{5.11}$$

where N is the number of neighbors of B after A leaves its transmission range, and,

$$p_A = 0 \tag{5.12}$$

In the case when A was the next hop with the highest probability for a certain destination d, this means that the best possible route to d from B is no longer available. In this case, the neighbor with the next highest probability is chosen to be the next hop. If this neighbor does not have a route available to the destination d, it sends a single forward ant for route discovery and waits for a backward ant before forwarding the data packet. On the other hand, if this node has a previously established route available, it simply forwards the data packet on this

route. It is important to note here that looping for data packets will not occur since probabilities for next hops are created only when a backward ant arrives at a node, and forward ants are inherently loop-free.

Figure 5.7 illustrates how a new route is found by the adjustment of probabilities at the routing tables. Initially, the routing probabilities at node 2 are such that the data packet is routed to node 3 as the next hop. However, as node 3 moves out of transmission range of node 2, the routing table of 2 changes so that 4, which is the neighbor with the next-highest probability for destination d, is the next hop. The new route is then found through node 4.

5.1.6 Routing Data packets

The data packets can now be routed via a number of possible schemes:

1. The data packets can be routed on the basis of the highest probability for the next hop at a node for the data packet's eventual destination. This creates a complete global route by using local information.

2. The data packets can also be routed probabilistically. Previous results [6] for swarm intelligence algorithms show excellent results for this method in the case of static networks with relatively small topologies. However, this might not be the suitable method in the case of the mobile ad hoc environment with rapid topology changes.

5.1.7 Modularity in PERA

Most routing protocols consist of the following three phases of operation:

- Route Discovery,
- Route Maintenance,
- Route Table Maintenance
- Local Connectivity Management

In the route discovery process, packets are sent between nodes for the discovery of routes between-source destination pairs. Proactive protocols do route discovery continuously and typically for all the nodes in the network. Reactive protocols typically go into the route discovery phase only when a route is required for sending a data packet. The routing table entries are changed suitably to reflect established paths.

In the route maintenance process, existing routes are updated and possible new routes are established between source-destination pairs. Once again, proactive attempt to do this constantly whereas reactive protocols go into this phase only when a previously available route for data packets is found to be no longer available.

Nodes use a Local Connectivity Management scheme to keep a record of their immediate, single-hop neighborhood. This allows them to keep a track of local network status and change their routing table accordingly.

Figure 5.8 shows a comparison between AODV and PERA in terms of the different phases, route discovery and route maintenance.

```
t:= Current time;
t_end:= Time length of the simulation;
Δt:= Time interval between ants generation;
foreach (Node)  /* Concurrent activity over the network  */
      M = Local traffic model;  /* Contains μ_d and σ_d for some destinations */
      τ  = Node routing table;
      while  (t <= t_end)
        in_parallel  /* Concurrent activity on each node  */
        if (t mod Δt = 0)
          LaunchForwardAnt(destination node, source node)
        end if
  foreach (ActiveForwardAnt[source node, current node, destination node])
            while (current node ≠ destination node)
              next_hop_node := BroadcastAnt(current_node, destination_ node);
              PushOnTheStack(next_hop, node_elapsed_time);
              current_node:=next_hop_node;
            end while
            LaunchBackwardAnt(destination_node, source_node, stack_data);
            Die();
        end foreach
foreach (ActiveBackwardAnt[source_node, current_node, destination_node])
            while (current_node ≠  destination node)
              next_hop_node := PopTheStack();
              WaitOnHighPriorityLinkQueue(current_node, next_hop_node);
              CrossTheLink(current_node, next_hop_node);
              UpdateLocalTrafficModel(M, current_node, source_node, stack_data);
              UpdateLocalRoutingTable(τ, current_node, source_ node,
      reinforcement)
            end while
        end foreach
      end in_parallel
  end while
end foreach
```

Figure 5.6: Pseudo code for the algorithm.

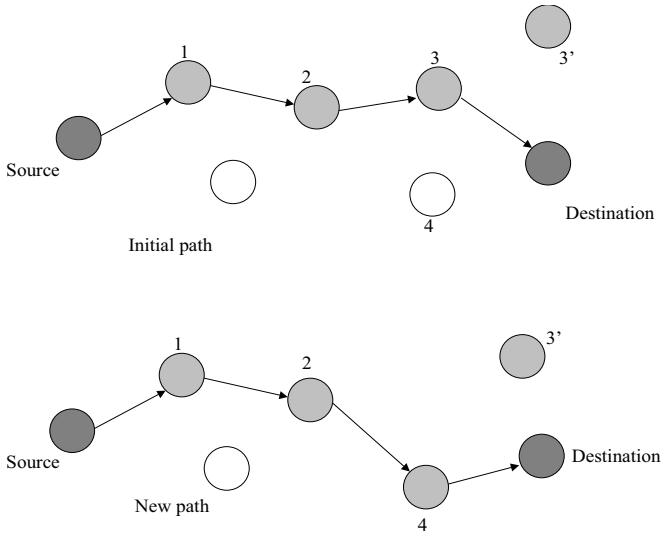

Figure 5.7: A new path is found by using probability adjustments in the case of link breakage due to node movement.

	PERA-Reactive	AODV
Route Discovery	•First ant sent to a destination if no route available, then forward ants are sent periodically •Forward ants are broadcast to neighboring nodes •Forward Ants, Backward Ants are the routing packet types •Frequency of sending ants can be varied according to the variances of routes	•RREQ sent to a destination if no route available •RREQs are broadcast to neighboring nodes •Route Requests (RREQs), Route Replies (RREPs) and Route Errors (RERRs) are routing packet types
Route Table	•Per destination next hop probabilities •Per destination routes, possibly multiple in number •Route statistics – route means and variances	•Per destination next hop entries at each node •Modifications proposed to allow multiple routes
Routing Data Packets	•Data packets buffered if no route found •Route discovery phase •If a route is found, packets are routed accordingly to next hop •If no route is found after a number of attempts, packets are routed probabilistically Similar process at intermediate nodes	•Data packets are routed on a per destination next hop basis.

Figure 5.8: Forward ants flooded to find multiple routes to a node.

Chapter 6

Algorithm Parameters

The routing problem in Mobile Ad Hoc Networks is a stochastic distributed multi-objective problem. Information propagation delays and the difficulty to completely characterize the network dynamics under arbitrary traffic pattern and node mobility make the routing problem in MANETs intrinsically distributed. Routing decisions can be made only on the basis of local and approximate information about the current and future state of the network.

These features make the problem well adapted to be solved by a multi-agent approach like swarm intelligence algorithms. The important parameters that control the performance of the algorithm described in chapter 5 are:

- The *rate at which Forward Ants* are sent from a node to explore feasible paths to destination nodes in the network. To keep the routing tables updated at frequent intervals, and with the latest information regarding the status of the network, forward ants are required to be sent at relatively frequent intervals. On the other hand, though each forward ant that is broadcast may generate a set of routes (since it generates possibly multiple backward ants), it also

generates significant overhead. In the MANET environment, the routing overhead is required to be low due to the low power and limited bandwidth available at individual nodes. Therefore, schemes which allow the algorithm to control the sending rate of ants are key to the successful functioning of the algorithm.

- *The learning rule and the reinforcement* that the backward ants use to update routing tables. This determines the speed with which the algorithm adapts to changes in the network topology. This also determines which routes are given high reinforcement relative to others. Further, the reinforcement parameters determine how quickly newly found routes are reinforced and actually used for routing data packets. Good routes should be rewarded substantially as long as they are available and newly available routes that are more efficient than pre-existing routes should be quickly 'ramped-up' to high probability (goodness) values.

6.1 Routing Table Updates

The algorithm discussed in chapter 5 provides a method of multi-objective optimization in the mobile, ad hoc environment. An important consideration, then, is the metric or combination of metrics to be used for obtaining good solutions to the routing problem. The metric of choice for most popular routing algorithms has traditionally been the number of hops. AODV [18] and DSR [14], two of the most

popular algorithms use this metric. Other routing algorithms such as ABR [13] use a 'connectivity relationship' metric. This combines the connectivity of a node to its neighbors with the available battery power at the node to obtain a metric for finding good routes in the MANET.

In our experiments using swarm intelligence inspired algorithms, we use delay and the number of hops as two possible metrics. However, this algorithm has further possible applications with other metrics, such as the battery power available at the node. In short, this algorithm provides a generalized method of optimization of the required metric.

The time measure (delay) can be used as the reinforcement signal to provide structural and temporal credit assignment. The observed times are absolute measures and we cannot associate with them a precise error measure. We cannot score the trip times according to an absolute scale because "optimal" times depend on the traffic, and the network mobility, and they need to be considered from a network wide point of view.

If the network is in a congested state, all the trip times of the ants will score poorly with respect to times observed in low load situations, but we should nevertheless be able to understand that the network is under congestion and therefore we should give adequate reward to a trip time high with respect to absolute values but good relative to the trip times observed in the congested situation.

The update of the routing tables happens using the available feedback signal, that is, the trip time experienced by the forward ant. This gives a clear indication

about the goodness of the followed route because it is proportional to its length from a physical point of view (number of hops, transmission capacity of the used links, processing speed of the traversed nodes) and from a traffic congestion point of view. This aspect is extremely important as forward ants share the same queues as data packets, whereas backward ants have priority over data to propagate faster the accumulated information regarding the state of the network. So if the forward ants cross a congested region of the network with high delays, they will take a longer time. This has two effects:

1. The trip grows and the back-propagated increments to the routing tables are small.

2. Also, these increments will be assigned with a larger delay (i.e. the interval between two consecutive increments will be large). Hence, links and paths that are congested will be given a small, delayed reward.

In a mobile environment, the number of hops is a good metric as it allows the feedback to be rapidly disseminated in the network. Unlike in the case of using delay as a metric, when using the number of hops as a metric, the forward ants are not required to move on normal queues. As such, they do not face possibly long delays and the feedback is largely independent of the network status and congestion. Both forward and backward ants move on high priority queues. This is similar to other reactive MANET protocols such as AODV and DSR. This allows quick adaptation and optimization to find the paths with the fewest hops. However,

this does not give accurate information about the network in terms of the traffic flows between nodes, the delays and congestion at nodes. As a result, in the case of high traffic flows, this may lead to inefficiencies in the network.

There are several methods or learning rules that can be used for updating and adjusting the probability distributions at each node. These update rules aim to increase the probability for the next hop for nodes from which backward ants have just been received. Hence, feedback regarding the status of the network from the backward ants is used as reinforcement to adjust the routing tables.

In the following sections, we discuss some learning rules and the results obtained in experiments using these updating rules.

6.1.1 Learning Rule 1

We discuss here a learning rule proposed by Dorigo et al. in their work on AntNet, an ant routing algorithm for communication networks [6].

As noted in section 5.1.4, each forward ant broadcast from the source node may result in the reception of a number of backward ants at the source of the forward ant. All the backward ants are used to update the routing probabilities at intermediate nodes as well as the source node. This allows the building of alternative routes (rather than just the best route as per the perception of the forward ants of the status of the network). If only the best performing ant is used to update the routing table probabilities, then only one route is reinforced.

The remaining routes, which would be useful in the case of node mobility, are not reinforced at all. This leads to the building up of a single, temporarily good route. On the other hand, by using all the backward ants to adjust the probability tables, we ensure that routes other than the best-performing are also reinforced. With mobility and the breakage of links, the second best route can then be used to route data packets if the best route is no longer available.

The statistics $(\mu_{sd}, \sigma^2_{sd})$ are updated only for the first backward ant that is received by the source node s of the corresponding forward ant to destination d. This ensures that only the statistics of the best known route are available to the node. Since the statistics at a node for a destination may be used to control the sending rate of ants and for grading routes at the node for the destination, only the mean and the variance of the best available route should be used. Using cumulative statistics from all the backward ants will give an inaccurate representation of the network status at the node for a given destination.

Consider a forward ant $F^l_{s\to d}$, the kth forward ant from source node s to destination node d. This ant generates a set of backward ants, $B^{l1}_{d\to s}, B^{l2}_{d\to s}, ..., B^{ln}_{d\to s}$. The first backward ant to arrive at node s is then chosen to update the statistics (μ_{sd}, σ_{sd}) for destination d. The following discusses a procedure to update the probability distribution at the routing table of each node that the backward ant visits. This is similar to the learning method used by DiCaro and Dorigo in [6].

If T is the trip time of the backward ant received at a node i (which is either the source node of the forward ant or an intermediate node on the path between

source node s and destination node d) and μ_{id} is the mean value of previous good round trip times as stored in the routing table as part of the vector $(\mu_{id}, \sigma_{id}^2)$, then we can compute a raw quantity r' measuring the goodness of T, with small values of r' corresponding to satisfactory trip times,

$$r' = \frac{T}{c\mu},\ c \geq 1\ if\ \frac{T}{c\mu_{id}} < 1 \tag{6.1}$$

$$= 1,\ otherwise. \tag{6.2}$$

r' is a problem-independent measure that scores how good the trip time is with respect to the average of the observe trip times. c is a scaling factor and it is assigned a reasonable value (currently chosen to be 2). Out-of-scale values are given a value of 1.

A correction strategy is applied to the goodness measure r' taking into account how reliable the currently observed trip time is with respect to the variance in the the so far observed values. This takes into account the stability of the trip time. Observations are considered stable if

$$\frac{\sigma_{id}}{\mu_{id}} < \epsilon, \epsilon \ll 1 \tag{6.3}$$

In this case, a good trip time, (that is $r' < t$ where t is arbitrarily set to 0.5) is decreased by subtracting a value

$$S(\sigma_{id}, \mu_{id}; a) = e^{-a\frac{\sigma_{id}}{\mu_{id}}} \tag{6.4}$$

to the value of r', while a poor trip time is increased by adding the same quantity.

On the other hand, if the mean is not stable, the raw values r' cannot be completely considered reliable and, in this case, the quantity

$$U(\sigma_{id}, \mu_{id}; a) = 1 - e^{-a' \frac{\sigma_{id}}{\mu_{id}}}, \tag{6.5}$$

with $a' \leq a$, is added to a good r' value and subtracted from a poor one. In this case, the attempt is to avoid traffic fluctuations, with the risk of amplifying them.

Then, the above correction strategy can be summarized as:

$$r' \leftarrow r' + sign(t - r')sign(\frac{\sigma_{id}}{\mu_{id}} - \epsilon)f(\sigma_{id}, \mu_{id}), \tag{6.6}$$

with f being$S$ or U according to the case. The final value of r' is bounded in the interval [0,1].

The obtained value r' is used by the current node i to define a positive reinforcement, r_+, for the node f that the backward ant comes from, and a negative one, r_-, for the other neighboring nodes n:

$$r_+ = (1 - r')(1 - P_{fd}) \tag{6.7}$$

$$r_- = -(1 - r')P_{nd}, n\epsilon N_k, n \neq f, \tag{6.8}$$

where P_{fd} and P_{nd} are the last probability values assigned to neighbors of node k for destination d. N_K is the set of k's neighbors. In this way, the reinforcements

are proportional to the obtained goodness measure r' and to the previous values of the goodness probabilities.

These probabilities are then increased or decreased by the reinforcement values as follows:

$$P_{fd} \leftarrow P_{fd} + r_+, \; P_{nd} \leftarrow P_{nd} + r_- \tag{6.9}$$

Thus, the rescaling of the r' value is equivalent to the definition of the learning rate, the scale compression factor and its degree of non-linearity determine the final size of the probability changes. The constants $(c, a, a', \epsilon, h, t)$ are not problem-dependent and simply provide a suitable scaling.

6.1.1.1 Actor-Critic Systems

The principle behind the algorithm is similar to Reinforcement Learning [12] and can be represented as an actor-critic system. The raw information as represented by the round trip times in this case is processed by the critic and then used to train the actor to improve the system performance. The quantity r' is used to process the raw information. Figure 6.1 is a representation of actor-critic systems.

6.1.2 Learning Rule 2

The following learning rule was proposed by Schoonderwoerd et al. [10] in their algorithm, Ant Based Control.

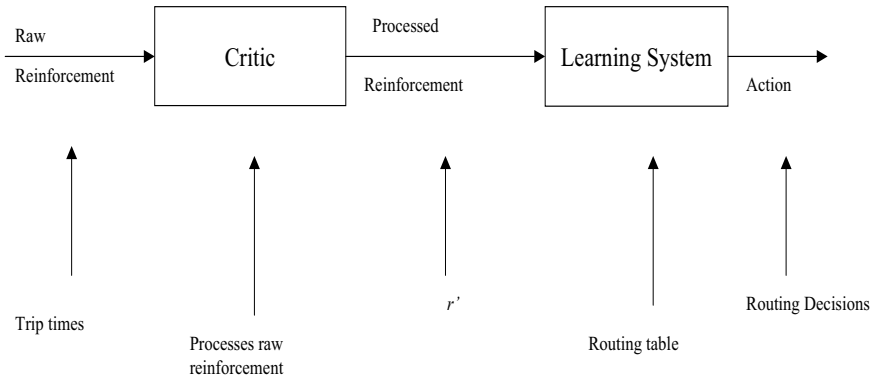

Figure 6.1: The Actor-Critic system.

In the algorithm for call routing in telecommunication networks proposed in [10] and mentioned in chapter 2, ants go from their source node to the destination node by moving from node to node. The next node an ant will move to is selected probabilistically according to the routing table of its current node. When the ant reaches its destination node it is deleted. Each visited node's routing table is updated by the deposition of some pheromone on the routing table entry corresponding to the neighbor node it just came from and to the ant source node, which is viewed as a destination node. Formally, routing tables are updated as follows. Let s be the source node of an ant, f be the node it just came from, and i its current node at time t. The entry $P_{fd}(t+1)$ is reinforced while other entries $P_{nd}(t+1), n \neq f$, in the same column decay by probability normalization.

$$P_{fd}(t+1) = \frac{P_{fd}(t)+\delta r}{1+\delta r}, \tag{6.10}$$

$$P_{nd}(t+1) = \frac{P_{nd}(t)}{1+\delta r}, n \neq f, \tag{6.11}$$

where δr is the reinforcement parameter that depends on the ant's characteristics such as the age. This updating procedure also conserves the normalization of the probability values and keeps the sum as 1 if these values are initially normalized. The probability P_{fd} is more reinforced when it is small (eg. when the node f was not already on an established path between s and d) and reinforced to a smaller extent when it is large (ie. the intermediate node f is already on the path between s and d). This allows new routes to be discovered quickly when established routes have become congested or unstable. There is, however, an exploration-exploitation trade-off, since a large amount of instability in choosing routes does not allow localization of searches.

The amount of reinforcement δr depends on how well the ant is performing. The age (time taken to traverse the nodes) of the ant can be used to determine δr. In principle, the age of the ant should be proportional to the cost metric (eg. delay along a path). If an ant takes a long time on a certain path, then δr should be reduced. Schooenderwoerd et al. [10] choose to use the absolute time (or age) T, measured in time units spent by the ant in the network and propose

$$\delta r = \frac{a}{T} + b, \tag{6.12}$$

where a and b are chosen parameters.

6.1.3 Using a Cost Function to update probabilities

A non-linear learning rule for the probabilities was proposed by Chen et al. [11]. This rule uses the cost function, $f(c)$ of the links traversed by each ant to update the probabilities.

$$P_{fd} = \frac{P_{fd} + \Delta p}{1 + \Delta p}, \tag{6.13}$$

$$P_{nd} = \frac{P_{nd}}{1 + \Delta p}, 1 \leq j \leq n, i \neq j \tag{6.14}$$

where $\Delta p = \frac{k}{f(c)}, k > 0$ is a constant and $f(c)$ is a non-decreasing function of c, the cost. P_{fd} is the probability of taking neighbor f (from where the backward ant has been received) as the next hop to reach destination d. P_{nd}'s are the probabilities of taking the other neighbors, $n, n \neq f$ as the next hop to reach destination d.

The constant k is the *learning rate* of the algorithm. It should be set high enough so that it has a substantial effect on P_{fd} for each ant and it should be low enough so that the routing tables converge with high probability. As for earlier learning rules, k can also be generated as a function of the absolute cost of the route when compared to the mean of the cost observed so far.

A significant advantage of this scheme and of the ant routing paradigm in general is the fact that the learning rules can be used to optimize the routing table with respect to one of a number of metrics, or with respect to a number of metrics.

- Delay
- Number of hops
- A trust metric
- Battery power available at a node or at a set of nodes
- Mobility

6.2 The Sending Rate of Ants

The sending rate of forward ants is the most important criterion for routing as an optimization problem. One of the key aspects of swarm intelligence techniques applied to the MANET environment is the tradeoff between sending ants frequently and the routing overhead generated by these ants. Sending out ants at a high frequency allows high adaptivity to changes in the network. It also allows for more optimized routes in terms of the metric. However, this creates high overhead and inefficient operation.

We attempt to control the sending rate of Forward Ants by two mechanisms:

1. By sending ants only when a node receives a data packet for a destination for which it has no route. The forward ants then establish routes in the network. Data packets are buffered until a route is established. Once a route has been established, the data packets are dequeued and routed as per the available routing probabilities.

2. We use available information about existing routes to establish a method to control the sending of Forward Ants. For each route (as built by the probabilities at each intermediate node), along with the next hop, we also maintain the mean and the variance $(\mu_{fd}, \sigma^2_{fd})$ of the route (between node f and destination d) as perceived by the ants.

   The variance of the routes give us information about the stability of the route. Then, we can base the rate of sending forward ants on the variance of the route.

Chapter 7

Simulation Results

7.1 Simulation Environment

Network Simulator 2 [28] discrete event simulator was used to simulate our algorithm. The simulation modeled a network of nodes randomly placed within an area of varying dimensions. The network density, i.e., the number of nodes per unit area of the network were varied as simulation parameters. At the physical layer, radio propagation distance for each node was set to $250m$ and the channel capacity was $2Mbps$. Our model does not support radio capture [27] so, in the case of packet collisions all packets are dropped. The IEEE 802.11 Distributed Coordination Function (DCF) [26] as implemented in NS2 was used as the Medium Access Control (MAC) protocol. The communication medium is broadcast and nodes have bi-directional connectivity. Each simulation was run for 900 seconds. Multiple runs with different seed values were conducted for each scenario and the collected data were averaged over those runs. The algorithm was developed as a separate NS2 routing layer protocol.

7.1.1 Node Placement

Nodes in the network were randomly placed. Hence, network partitions can exist, irrespective of the density of the network. For a given experiment, all routing protocols compared are configured with the same seed value. Thus all schemes will encounter the same network scenario. Hence the performance of the routing protocols can be directly compared.

7.1.2 Mobility Model

All nodes in the network are mobile and move according to the "Random Waypoint" model [17]. In this model nodes move at the set speed for a specified period of time towards a random destination and after reaching the destination nodes pause for a set amount of time (a simulation parameter). After this period a new random destination is chosen and the node moves towards this new destination with the same speed. If a node reaches the boundary of the fixed area, the node will rebound. Thus, the nodes are free to move to any region in the network area. The continuous movement of the nodes ensures continuous change in the topology. This highly dynamic network is ideal to test the reliability of our algorithms.

7.1.3 Other simulation parameters

In all our simulations we use a set of eight end-to-end connections, that is, eight source nodes (two or more of which might be the same) attempt to send constant

bit rate traffic to eight destination nodes (again, two or more of which might be the same). The pause time is maintained at 50 seconds throughout and the mobility speeds range from 1 m/s to 20 m/s.

7.1.4 Application Traffic

The NS2 source generator model was used to generate Constant Bit Rate (CBR) traffic. The size of each data packet payload was 128 bytes. Source nodes start generating data at random instants of time. The source then sends data packets at constant intervals of time. This interval is determined by the traffic load.In fact, the interval is the inverse of the load in *packets/sec*. The value of *packets/sec* throughout the simulations was 1.

7.2 Simulation Methodology

7.2.1 Algorithms for Evaluation

Since a large number of routing algorithms exist for Mobile Ad Hoc networks, it is important to compare the performance of our routing algorithm against other algorithms. We use the Ad Hoc On Demand (AODV) routing algorithm for comparisons. This gives us a comparative analysis of swarm algorithms for MANETs when compared to both Link State and Distance Vector algorithms.

We use the throughput, the goodput and the average end-to-end packet trans-

mission delay for comparisons. All the simulations were carried out with the same seed for the given simulation scenario and hence the results can be directly compared for the routing algorithms.

7.2.2 Simulation Attributes

The above schemes were evaluated as a function of the following attributes.

7.2.2.1 Protocol specific parameters

- The update rule used. As noted in 6.1, a number of possible methods of updating the routing tables can be used.
- The value of the reinforcement used to calculate the applied reinforcement as discussed in 6.1

7.2.2.2 Network parameters

- Mobility speed: The speed at which the network nodes move is varied. Variation against mobility determines the robustness of the protocols.
- Pause time: The pause time in the random waypoint model is varied. The performance variation gives another indication of the robustness of the protocols.

7.2.3 Performance Metrics

The collated data from the different runs were used to generate the following metrics.

7.2.3.1 Goodput

The goodput is defined as the ratio of data packets received at the nodes to the number of routing packets received at the nodes. Thus, it is a measure of the efficiency of the routing protocol in terms of the number of routing packets used to route each data packet.

$$Goodput = \frac{Data\ packets\ received\ at\ routers * 100}{Routing\ packets\ received\ at\ routers} \tag{7.1}$$

7.2.3.2 Throughput

The throughput is defined as the ratio of the number of data packets received to the number of data packets sent.

$$Throughput = \frac{Data\ packets\ received * 100}{Data\ packets\ sent} \tag{7.2}$$

7.2.3.3 Average End-to-End Delay

The end-to-end delay is the interval between the instant a source generates a packet and the time at which the destination receives the packet. The end-to-end delay is aggregated for each packet for each source-destination pair. The average per

packet end-to-end delay is then calculated as the number of source-destination pairs and the number of packets received is known.

7.3 Simulation Results and Trade-off Analysis

In this section we present the simulation results for all the schemes as a function of the simulation attributes described in 7.2.2. The basic simulation model for all the experiments is as described in 7.1. Also we analyze the trade-off characteristics of these schemes.

7.3.1 Hop count based optimization

In this experiment, we evaluate the performance of the routing algorithm based on the hop count metric.

The network consisted of 20 nodes, randomly placed in an area $500m$ x 500. 4 source and destination pairs were randomly chosen from these 20 nodes. Each source transmitted 1 $packet/sec$. Nodes in the simulation were mobile.

7.3.2 Mobility Speed

In this experiment,the mobility speed was varied between 0 to 20 m/s (0,5,10,20,15,20). The model consisted of 20 nodes randomly placed in the network. In each run of the simulation, all the schemes were tested on the same scenario, this was ensured by setting the same seed for all the scenarios. The seed was changed

for different runs. 4 source-destination pairs were randomly chosen from these 20 nodes. The sources generated 1 packet/sec.

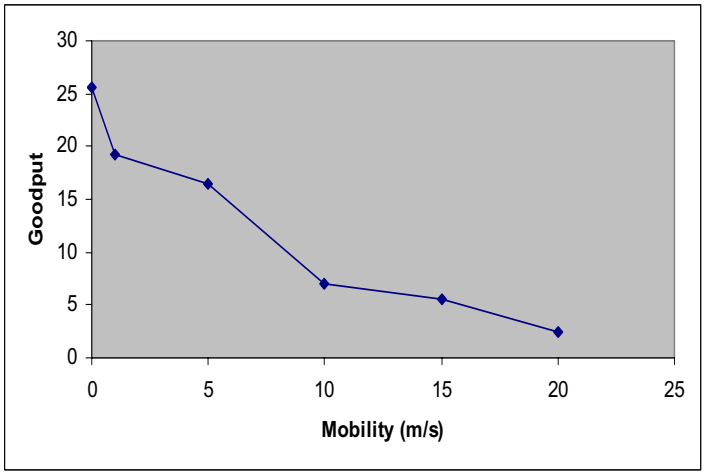

Figure 7.1: Variation in the goodput with mobility in a scenario with 20 nodes in 500m X 500m.

Figure 7.1 shows the goodput as a function of the node mobility speed. It is seen that the goodput decreases with increase in mobility. This is to be expected since with an increase in mobility, a larger number of forward ants are required to be sent to discover new routes and modify and update existing routes which are no longer available for packet transmission. Hence, at higher speeds, the number of routing packets required per data packet correctly transmitted is high, leading to lower goodput.

Figure 7.2 shows the percentage packet loss as a function of the mobility. With 0 and low mobility (1 m/s), the packet loss is 0. With speeds of 5 m/s, the packet

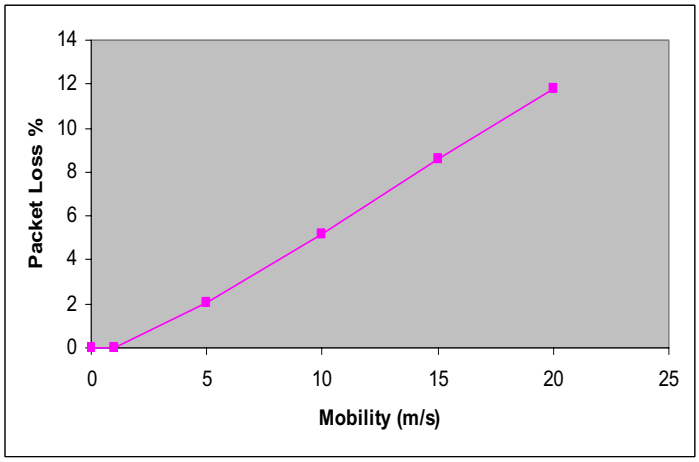

Figure 7.2: The percentage packet loss for varying mobility with 20 nodes in 500m X 500m.

loss in under 2%. However, with increasing mobility, the packet loss increases linearly. Thus, even the increased rate of sending ants (as evidenced by the decreased goodput) does not serve to maintain a low percentage of packet loss. To keep the packet losses low, the rate of sending ants has to be increased non-linearly. That is, the rate of increase in sending of ants has to increase faster than the mobility.

7.3.3 Rate of sending Forward Ants

In this experiment the rate of sending forward ants is varied for different mobility speeds and the behavior of the algorithm is studied.

Table 7.3.3 shows the variation in goodput and percentage packet loss as a function of the $ANT\_INTERVAL$ (the time period between the transmission of

ANT_INTERVAL	Goodput	% Packet Loss
15	7.92	4.39
25	11.44	10.37
50	7.92	11.60
100	19.24	15.59

Table 7.1: Goodput and % Packet Loss with ANT_INTERVAL with mobility of 10 m/s.

two forward ants) for 20 nodes in an area of $500m\ X500m$ with speeds of 10 m/s and a pause time of 50 *secs*. For a high value of $ANT\_INTERVAL$, the packet loss is high. This is explained by the fact that information regarding the current state of the network is not updated rapidly. The algorithm fails to adapt in many cases resulting in high packet loss. However, as the period between the sending of two consecutive forward ants is decreased, the packet loss reduces significantly. This shows that the algorithm adapts to the changes in the network quickly as the number of forward ants being sent increases. With a value of 15 seconds for the $ANT\_INTERVAL$, the packet loss is 4.39%.

Table 7.3.3 shows the variation in the goodput and percentage packet loss as a function of the $ANT\_INTERVAL$ for 20 nodes in an area of $500m\ X\ 500m$ with speeds of 5 m/s and a pause time of 50 *secs*. For a low value of $ANT\_INTERVAL$, the packet loss is lower than for a higher value. Further, it is important to note that the packet loss for values of $ANT\_INTERVAL$ 100 and 150 are the same.

ANT_INTERVAL	Goodput	% Packet Loss
50	12.12	3.67
100	9.29	5.45
150	8.37	5.45

Table 7.2: Goodput and % Packet Loss as functions of ANT_INTERVAL for mobility of 5 m/s.

ANT_INTERVAL	Goodput	% Packet Loss
300	17.23	0
900	19.18	0

Table 7.3: Goodput and % Packet Loss as functions of mobility with 1 m/s.

This is because the increase in the number of forward ants that are sent is not sufficient to cause an increase in performance in terms of goodput and packet loss. The goodput therefore goes down since the packet loss remains constant.

Table 7.3.3 shows the variation in the goodput and percentage packet loss as a function of the $ANT\_INTERVAL$ for 20 nodes in an area of $500m\ X\ 500m$ with speeds of 1 m/s and pause time of 50 *secs*. Since the mobility is very low, the adaptivity required of the algorithm is relatively low. Even by sending ants at a higher rate, there is no change in the packet loss, since a single forward ant sent at the start of the simulation obtains enough data (with some redundancy due to the availability of multiple routes) for all data packets to be successfully routed.

7.3.4 Reinforcement

In this experiment the value of the reinforcement used based on the learning rule described in section 6.1.1 to update the routing tables at the nodes is varied between 0.1 and 0.5 (0.1, 0.15, 0.20, 0.30, 0.40, 0.50). The network consists of 20 nodes randomly placed in an area of $500m$ X $500m$ with speeds of 5 m/s and 10 m/s and pause time 50 $secs$. This experiment allows us to study the impact of the reinforcement value on algorithm performance.

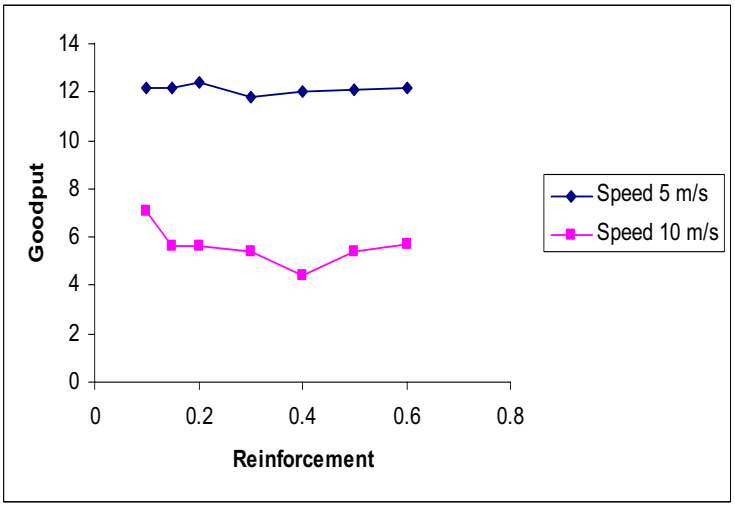

Figure 7.3: Variation in the goodput with the reinforcement parameter.

Figure 7.3 shows the variation of the goodput as a function of the value of the reinforcement. The speeds of the nodes are 5 m/s and 10 m/s. The goodput is higher for speed 5 m/s, as expected.

Figure 7.4 shows the variation of the percentage packet loss as a function of the value of the reinforcement for 20 nodes in an area of $500m$ X $500m$ with the nodes

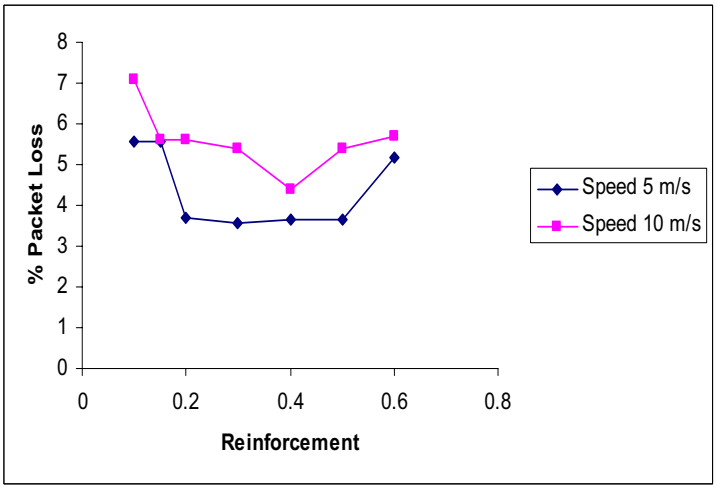

Figure 7.4: The percentage packet loss with varying reinforcement parameter for 20 nodes in 500m X 500m.

moving at speeds of 5 m/s and 10 m/s and a pause time of 100 $secs$. As expected, the packet loss is lower for lower speed of movement. However, it is important to note that with an increasing value of the reinforcement applied, the packet loss first increases and then decreases. This is because a weak (small) applied reinforcement implies that routes do not get positively reinforced to a sufficiently high degree. In the situation where mobility exists in the network, this reduces the adaptivity of the algortihm, leading to stale routes being used for the transmission of data packets. If the reinforcement applied is increased beyond a certain value, it causes the routes to be reinforced too fast. This leads to routes that may not actually be the best routes being used for the transmission of data packets.

7.4 Comparison with AODV

As mentioned earlier, we compare the algorithm proposed in chapter 5 with AODV in terms of the throughput, delay and the goodput.

7.4.1 Goodput comparison

Figure 7.5 shows a comparison of the goodput for AODV and PERA for a scenario with 20 nodes in an area of $500m\ X\ 500m$ with the nodes moving with speeds of 1 m/s and a pause time of 100 *secs*. Since the mobility is low, the overall goodput for both algorithms is also low. However, while the goodput for AODV is nearly the same throughout, that of PERA shows marked variations. This depicts the learning phases of the algorithm - in the initial phase, forward ants are sent to discover new paths in the network. Once these paths have been established, the algorithm has learnt the current state of the network.

Figure 7.6 shows a comparison of PERA and AODV for the same scenario as above, but with a mobility speed of 10 m/s. The goodput is observed to be lower than that of AODV. This is because forward ants are sent more frequently to allow quick adaptation to the network conditions.

7.4.2 Throughput

Figures 7.7 and 7.8 show the throughput comparisons for AODV and PERA for mobility speeds of 1 m/s and 10 m/s and pause time 100 *secs*. At the lower speed,

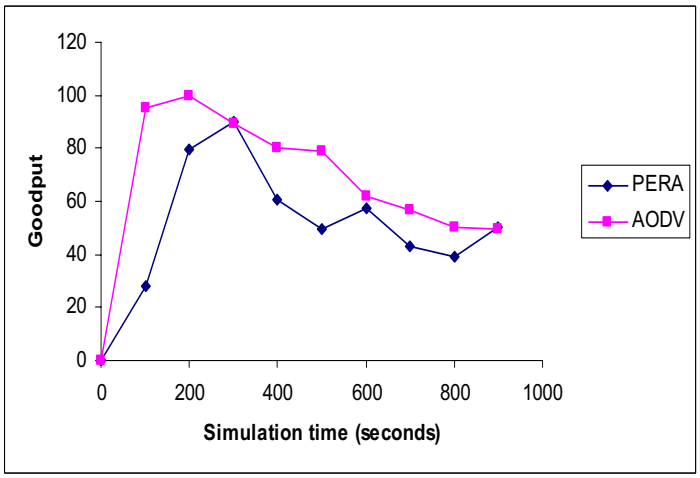

Figure 7.5: A goodput comparison of PERA and AODV at 1 m/s.

the throughput is the same for both AODV and PERA, however, at the higher speed, the throughput is slightly less for PERA in some cases. This is because with mobility, PERA adjusts gradually to the changes in topology.

7.4.3 Delay

Figures 7.9 and 7.10 show the comparison of delay for AODV and PERA. Both algorithms show a large initial delay, which is required for routes to be set up. Subsequently, AODV shows large delays again in situations with high mobility. PERA on the other hand, shows low delays in all cases, as instead of buffering data packets until a new route id found, PERA delivers the data packet through an alternate route.

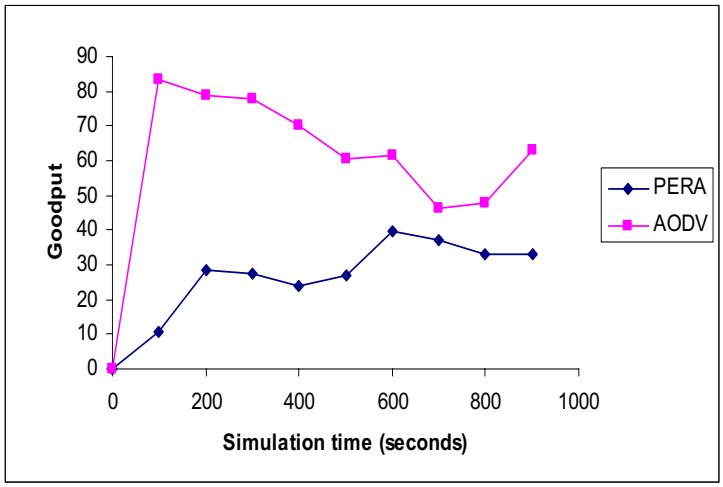

Figure 7.6: A goodput comparison of AODV and PERA at 10 m/s.

7.5 Some Comments

It should be noted that scenarios with network partitions and persistent mobility, though realistic, represent worst case scenarios. The protocols evaluated should perform better in networks with improved connectivity and predetermined or meaningful mobility patterns. Hence we attempt to establish the performance of our routing protocol in scenarios that are fairly well-connected.

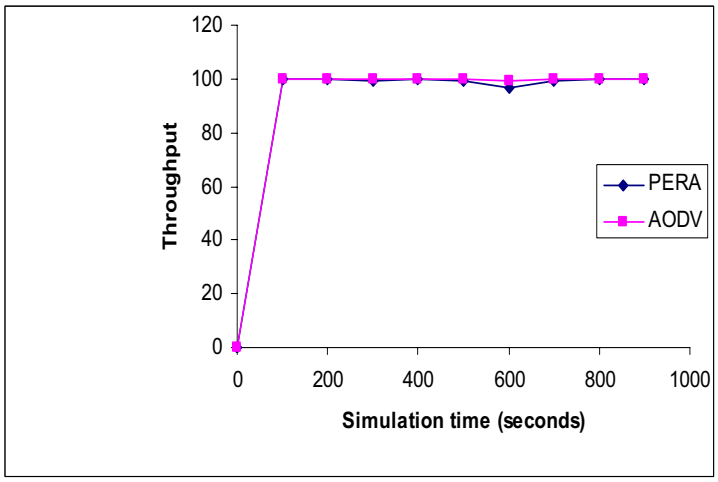

Figure 7.7: A throughput comparison of AODV and PERA at 1 m/s.

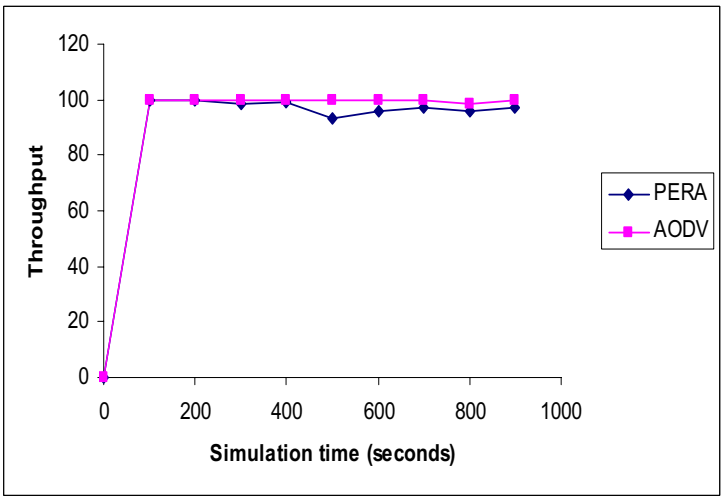

Figure 7.8: A throughput comparison of AODV and PERA at 10 m/s.

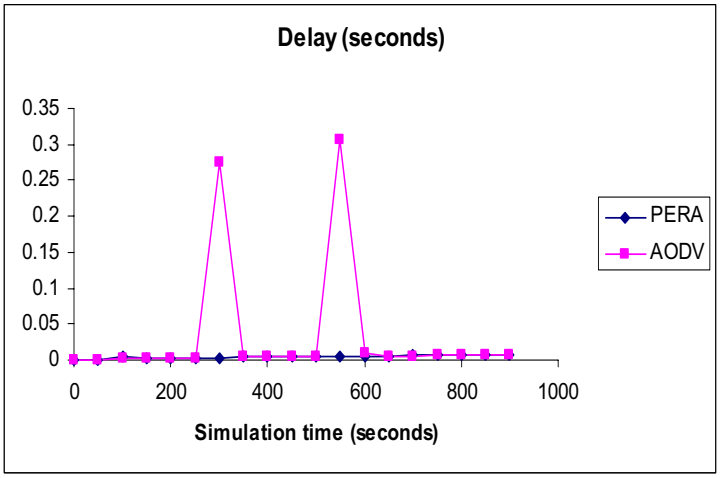

Figure 7.9: A delay comparison of AODV and PERA at 1 m/s.

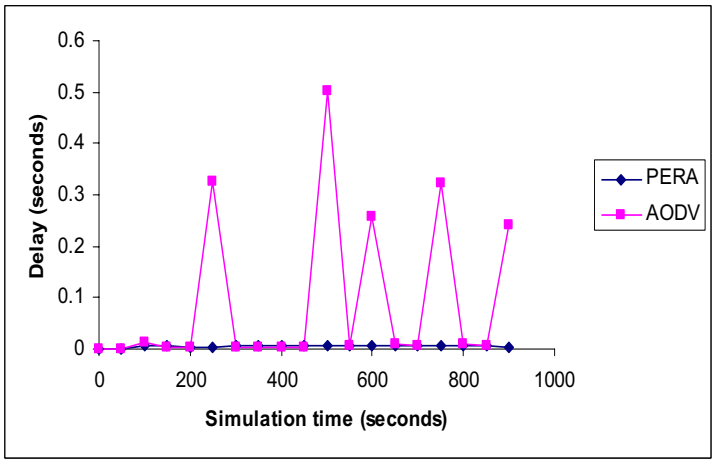

Figure 7.10: A goodput comparison of AODV and PERA at 10 m/s.

Chapter 8

Conclusions and Future Work

The inherent constraints of a mobile wireless ad hoc networks viz mobility, bandwidth and energy limitations, pose difficult challenges in designing routing algorithms. Thus, a routing algorithm for a MANET, should be robust against topology changes and achieve efficient data distribution. In this thesis we presented a routing algorithm for MANETs inspired by the swarm intelligence meta-heuristic. Swarm intelligence methods have been widely studied and successfully applied to difficult optimization problems. Further, these algorithms have been found to be robust and resistant to router state corruption. The algorithms uses two kinds of agents - forward and backward agents, referred to as forward and backward ants, to collect and disseminate information regarding the status of the network. These agents adjust a probability distribution at each node that reflects the likelihood of a reaching a given destination using a neighbor as the next hop. We found that the algorithm is robust against mobility and topology changes, and performs with minimum delay in high mobility conditions due to the presence of multiple graded paths. The algorithm is found to function particularly well in closely-

connected scenarios, but with increasing mobility, the routing packets required per data packet (as indicated by the goodput) goes up.

Some of the important aspects of these algorithms understood in the course of this work are the advantages of establishing and grading multiple paths probabilistically. This allows several routes to be found between source-destination pairs, without a large amount of bandwidth and storage resources (memory) being consumed. Further, a unicast approach to route discovery in MANETs incurs high overhead and low efficiency.

Another important aspect of this algorithm is that some of its important elements can be incorporated into many existing routing algorithms without incurring a large penalty in terms of overhead. We identify the following areas of future research:

- Introducing routing efficiencies by using more information from the MAC layer.
- A thorough study of the robustness and resistance to router state corruption of swarm intelligence-based algorithms under various extreme conditions.
- A study and analysis of incorporating ideas such as multiple paths and probabilistic grading of these paths into existing algorithms such as AODV and DSR.
- A study of operation of the algorithm under various metrics such as available energy at each node.

BIBLIOGRAPHY

[1] Corson, S. and Macker, J. Mobile Ad Hoc Networking(MANET): Routing Protocol Performance Issues and Evaluation Considerations.IETF MANET Working Group, January 1999. http://www.ietf.org/html.charters/manet-charter.html

[2] Broch, J. Maltz, D. Johnson, D. Hu, Y. Jetcheva. A Performance Comparison of MultiHop Wireless Adhoc Network Routing Protocols, Carnegie Mellon MONARCH Project. October 1998 http://www.monarch.cs.cmnu.edu/

[3] Perkins, C.and Royer, E. Ad Hoc On-Demand Distance Vector (AODV) Routing IETF MANET Working Group, January 1999. http://www.ietf.org/html.charters/manetcharter. html

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz. "Swarm Intelligence: From Natural to Artificial Systems". SFI Studies in the Sciences of Complexity. Oxford University Press, July 1999.

[5] E. Bonabeau, F. Henaux, S. Guerin, D. Snyers, P. Kuntz and G. Theraulauz. "Routing in Telecommunications Networks with Smart Ant-Like Agents",

in *Proceedings Intelligent Agents for Telecommunications Applications, IATA '98*, (Berlin), 1998.

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

[7] M. Dorigo and L.M. Gambardella. "Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem", *Biosystems* 43 (1997):73-81.

[8] E. L. Lawler and A. H. Rinnooy-Kan. "The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization", Wiley-Interscience Series in Discrete Mathematics, 2000.

[9] Z. Haas and M. Periman. "The Zone Routing Protocol (ZRP) for ad hoc networks", Internet draft, Mobile Ad-Hoc Network (MANET) Working Group, IETF, 1998.

[10] Schooenderwoerd R., O. Holland, J. Bruten, and L.Rothkrantz."Ant Based Load Balancing in Telecommunications Networks"", Adaptive Behavior 5, 1996: pp. 169-207.

[11] D. Subramaniam, P. Druschel and J. Chen."Ants and Reinforcement Learning : A Case Study in Routing in Dynamic Networks ", in*Proceedings of IEEE MILCOM*,(Atlantic City, NJ), 1997.

[12] I. Kassabalidis, M.A. El-Sharkawi, R.J. Marks II, P. Arabshahi, and A.A. Gray, "Swarm intelligence for routing in communication networks," in *Proceedings of IEEE Globecom 2001*, San Antonio, Texas, 2001.

[13] C-K Toh."Wireless ATM and Ad hoc Networks : Protocols and Architectures", Kluwer Academic Publishers.

[14] D.B. Johnson and D.A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks". In *Mobile Computing*, T. Imielinski and H. Korth, eds., Kluwer Academic Publishers, Norwell, Mass., 11996, 153-181.

[15] V.Park and M.S.Corson,"A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks",*Proceedings of IEEE INFOCOM*, (Kobe,Japan), 1997.

[16] C. E. PERKINS and P. BHAGWAT, "Higly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers", *Proc. of the SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications*, Aug 1994, pp 234-244.

[17] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks", In *Mobile Computing, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.*

[18] C.E.Perkins and E.M.Royer. "Ad-Hoc On Demand Distance Vector Routing", Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA), February 1999.

[19] S.Y.Ni, Y.-C.Tseng, Y.S.Chen, and J.-P.Sheu. "The Broadcast Problem in a Mobile Ad Hoc Network" Proceedings of MOBICOM, August 1999.

[20] M.S.Corson and J.Macker. "Mobile Ad hoc Networking (MANET):Routing Protocol Performance Issues and Evaluation Considerations", Request For Comments 2501, Internet Engineering Task Force, January 1999.

[21] B.M.Leiner, D.L.Neison, F.A.Tobagi. "Issues in Packet Radio Network Design", Proceedings of IEEE, special issue of Packet Radio Networks,1987.

[22] J.Jublin and J.D.Tornow. "The DARPA Packet Radio Network Protocols", Proceedings of the IEEE, vol 75, January 1987.

[23] G.S.Malkin and M.E.Steenstrup. "Distance-Vector Routing", in Routing in Communications Networks, edited by M.E.Steenstrup, Prentice Hall, 1995.

[24] J.Moy. "Link State Routing", in Routing in Communications Networks, edited by M.E.Steenstrup, Prentice Hall, 1995.

[25] S.Murthy and J.J.Garcia-Luna-Aceves. "An Efficient Routing Protocol for Wireless Networks", ACM/Baltzer Mobile Networks and Applications, special issue on Routing in Mobile Communications Networks, October 1996.

[26] IEEE Computer Society LAN MAN Standards Committee, Wireless LAN Medium Access Protocol (MAC) and Physical Layer (PHY) Specification, IEEE Std 802.11-1997. The Insititute of Electrical and Electronics Engineers, 1997.

[27] C.E.Perkins. "AD HOC NETWORKING", Addison Wesley, 2001.

[28] NS2 Manual and Documentation, http://www.isi.edu/nsnam/ns/ns-documentation.html