POSITION BASED ANT COLONY OPTIMIZATION ROUTING IN MOBILE AD HOC NETWORKS

Shahab Kamali

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

August 2007

Your file Votre référence
ISBN: 978-0-494-34752-2
Our file Notre référence
ISBN: 978-0-494-34752-2

Abstract

Position based ant colony optimization routing in mobile ad hoc networks

Shahab Kamali

Availability of cheap positioning instruments makes it possible for routing algorithms to use the advantage of knowing the positions of nodes in a mobile ad hoc network. Position based routing algorithms may fail to find a route from a source to a destination or the path that they find may be longer than the shortest path if the network contains nodes with irregular transmission ranges. On the other hand, routing algorithms which are based on ant colony optimization (ACO) find routing paths that are close in length to the shortest paths. The drawback of these algorithms is the large number of messages that needs to be sent or the long delay before the routes are established. In this thesis we propose two position based ACO routing algorithms for mobile ad hoc networks, POSANT and HYBNET. POSANT combines the idea of ant colony optimization with information about the position of nodes. HYBNET is a hybrid routing algorithm for mobile ad hoc networks which adapts itself to different network topologies. Our simulations show in most cases, POSANT and HYBNET perform better than the other routing algorithms.

Acknowledgments

I would like to express my sincere gratitude to my supervisor Jaroslav Opatrny who helped, guided, and motivated me since the first day that I arrived to Concordia.

I owe everything to my Mom, my Dad, and my brother, Shahin. Without their endless love and support I wouldn't have been here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mobile ad hoc networks (MANETs) consist of wireless mobile hosts that communicate with each other in the absence of a fixed infrastructure. Routes between two hosts in a mobile ad hoc network may consist of hops through other hosts in the network. Hosts are free to move and organize themselves arbitrarily which cause frequent unpredictable topology changes. Therefore, *routing*, the task of finding and maintaining routes between a source to a destination node, in MANETs is nontrivial. In order to reach a destination, a message must pass through some intermediary nodes; therefore these networks are usually called *multi-hop* networks. A routing algorithm is called *proactive* if it calculates routes before they are needed and tries to keep routing-information to all nodes every time up-to-date. A *reactive routing* algorithm calculates a route only when it is needed and does not try to keep routing information to all nodes always up-to-date.

The availability of cheap instruments like GPS receivers for estimating the position of nodes in a network motivated some researchers to propose *position based* routing algorithms. In position based routing algorithms it is assumed that a node is aware of its position, the position of its neighbors, and the position of the destination. We can mention *GPSR*[20] and *DIR* (also referred to as *Compass routing*)[22] as examples of reactive position based routing algorithms. The drawback of these algorithms is that they may fail to find a route or they may find a non-optimum route in some situations. Algorithms based on *face routing* [3, 5, 20] guarantee to find a route to the destination if it is possible to extract locally a planar subnetwork of a given network. In the networks where the transmission radius of nodes varies a lot, extracting a planar subnetwork is not always possible [3]. Hence face routing algorithms do not guarantee data delivery and they often find a route that is much longer than the shortest path [20]. *Partial flooding* methods are a group of position based routing algorithms which use information about the position of nodes to reduce the number of generated control messages. DREAM [4], V-GEDIR [25] and LAR [21] are examples of partial flooding algorithms. However these algorithms limit the number of generated control messages, in many cases the amount of generated control traffic is still high. This overhead makes these routing algorithms unsuitable for some applications.

Another family of routing algorithms for mobile ad-hoc networks is based on *ant colony optimization* (ACO). Information about the position of nodes has not been

used in the algorithms of this family. In general, the ant colony optimization meta-heuristic tries to solve a combinatorial problem using the collaboration of a group of simple agents called artificial ants. ACO routing algorithms establish optimum paths to the destination using a number of artificial ants that communicate indirectly with each other by *stigmergy*. Stigmergy is a way of indirect communication between individuals which, in an ad-hoc network case, is done through the modification of some parameters in the nodes of the network.

ANTNET[8] and *ANTHOCNET*[10] are two well known ant colony based routing algorithms. ANTNET is a proactive and ANTHOCNET is a reactive routing algorithm. They have a very high delivery rate and find routes whose lengths are very close to the length of the shortest path [8], [14]. The drawback of ANTHOCNET is the number of routing messages that needs to be sent in the network for establishing routes to the destination and the disadvantage of ANTNET is the time needed before a system of paths between the nodes of the network is established. This is referred to as the *convergence time*. Regarding the dynamic nature of mobile ad-hoc networks, a long convergence time is a significant drawback.

As we mentioned, the topology of a wireless ad hoc network may change frequently. For example a network may have more nodes in daytime than at night, thus a network graph which is usually dense in daytime may become sparse at night.

Position based routing algorithms perform very well in some networks with special

topologies but their performance is significantly reduced in some other topologies. For example, when the average degree of nodes (i.e. average number of neighbors) is high, DIR (this algorithm is defined in Chapter 3) performs very well, while in a sparse graph it may fail to find a route or the found routes may be longer than a shortest path. Algorithms which are based on *face routing* (refer to Chapter 3) may fail in graphs which contain nodes with different transmission ranges. Also when the average degree of nodes is low, the found route may be longer than the optimum[20, 13].

As with position based routing algorithms, the performance of ACO routing algorithms depends on the network topology. For large networks (i.e. networks with a large number of nodes), the number of ants generated by ANTHOCNET is very large. Furthermore, when the source and destination nodes are far from each other, long time is needed before a system of paths between the nodes of the network is established. This becomes a severe drawback of ANTNET.

To the best of our knowledge, all the position based and ACO routing algorithms proposed so far perform poorly in some network topologies.

The main contribution of this thesis is the proposal of two new position based ant colony optimization routing algorithms for mobile ad hoc networks.

First, we propose a new reactive routing algorithm which is based on ant colony optimization and uses information about the location of nodes in order to reduce the route establishment time while keeping the number of generated ants small in

comparison to other ant colony based routing algorithms. Since our algorithm is a position based ant colony routing algorithm, we call it POSANT (POSition based ANT colony routing for mobile ad hoc networks). The applications of POSANT may include cases where a huge amount of data is transmitted after route establishment and thus it is important to find optimum routes (e.g., streaming data over a network - video and audio streaming) and also where the irregularity of transmission ranges or of the node density limits the use of other position based routing algorithms.

Second, we present a hybrid reactive routing algorithm, called HYBNET, which combines the ideas of greedy position based routing algorithms with *POSANT* routing algorithm to adapt itself to different network topologies. This algorithm performs well in many different types of mobile ad hoc network topologies. The applications of this algorithm includes cases where the network graph includes dense, sparse or empty regions and the cases where the transmission range of nodes is highly irregular. In addition HYBNET has the same advantages as POSANT, which makes it suitable for the cases where a large amount of data is transmitted after route establishment and thus it is important to find optimum routes (i.e. an example is video and audio streaming).

In the next chapter we give a definition of the routing problem and the network model. We then explain the typical position based and ant colony based routing algorithms in more details in Chapters 3 and 4. Chapter 5 defines POSANT routing

algorithm and contains the simulation results of POSANT and a comparison with ANTNET, ANTHOCNET and GPSR routing algorithms. In Chapter 6, HYBNET routing algorithm is defined and its performance is investigated under different topological conditions. Also the effect of various parameters of the network topology on the performance of different routing algorithms is discussed. Chapter 7 contains conclusions and future works.

Chapter 2

Network model and problem definition

A *mobile ad-hoc network* is a group of mobile hosts which are able to transmit messages directly to the other hosts located in their transmission ranges. Each node is self configurable and can freely join the network, leave it or move inside it. Communication links are wireless and a connection may fail when a node moves or leaves the network (i.e. new connections may appear in a similar way).

We represent an ad-hoc network as a graph with an edge between each couple of nodes that can communicate directly. Each node may have a different transmission reach in different directions, i.e. as in reality where the existence of an obstacle or noise makes the transmission radius of a node irregular. Furthermore due to difference of power, different nodes may have different transmission ranges in a mobile ad hoc network. We assume all communications are bi-directional, i.e. a node maintains a

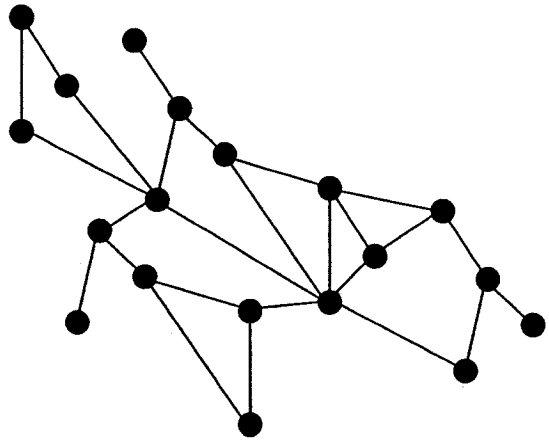

Figure 1: An example of the network model.

link to another node if they are able to exchange messages directly. An example of such network is shown in Fig.1.

Our model deals with a three dimensional distribution of nodes as follows. If nodes of a three dimensional network graph have similar heights (i.e. from the ground), we consider the projection of the nodes on the ground and connect two projected nodes if their corresponding nodes are connected. Since the heights of the nodes are similar, for a found route on the projected graph there is an equivalent route on the original graph.

In contrast to our model in which we assume different nodes may have different transmission ranges, in *unit dist graph* (UDG) model, it is assumed that the transmission range of all nodes is the same in all directions. This model is defined in the following way. Two nodes A and B in the network are neighbors (and thus joined by an edge) if the Euclidean distance between their coordinates in the network is at most R, where R is the transmission radius which is equal for all nodes in the

network (see Fig 2). Although the unit graph model is not realistic (e.g., it doesn't consider difference between the power of nodes, existence of noise or obstacles and so on. Details will be discussed in the next chapters.), most of the position based routing algorithms (i.e. defined in Chapter 3) are proposed for this model.

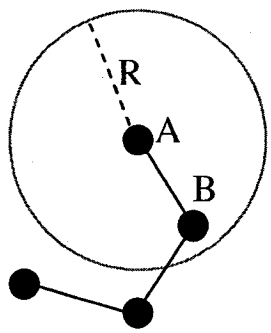

Figure 2: Unit graph representation of a mobile ad hoc network.

Each node is assumed to know the information about its position, the position of its neighbors and the position of the destination node.

The *routing problem* is to find one or more routes from a given source node to a given destination node in a wireless ad hoc network when a need for such a route arises.

We suppose the time required for a packet to move from one node to a neighbor node is the same for all nodes (i.e. we do not consider distance between two neighboring nodes, buffering, congestion and other causes of packet delay in this paper). It means the delay of a packet (i.e. the elapsed time since the moment that the packet is launched from the source node until the moment that the packet is received by the destination node) is a linear function of the number of hops in the way to the destination and hop count can be used as an evaluation metric for packet delays. Our

aim is to make the massage delivery delay as small as possible. Thus the goal is to establish one or more routes to the destination which are close in the number of nodes to the shortest path.

We suppose the route finding algorithm starts as soon as a data packet needs to be sent from a source to a given destination (reactive routing). To reduce the delay of data delivery we must find the route in the shortest possible time. Also the algorithm must guarantee finding a path to the destination when such a path exists.

Chapter 3

Position based routing algorithms

The possibility of estimating the position of nodes in a mobile ad hoc network motivated some researchers to propose position based routing algorithms [13]. The distance between neighboring nodes can be estimated on the basis of incoming signal strengths. Relative coordinates of neighboring nodes can be obtained by exchanging such information between neighbors. Alternatively, the location of nodes may be available directly by communicating with a satellite, using GPS (Global Positioning System), if nodes are equipped with a small low power GPS receiver. In position-based routing algorithms it is assumed that a node is aware of its position, the position of its neighbors, and the position of the destination. To obtain the position of the destination node, position based routing protocols assume a *location service* is available that provides location information on the nodes in the network. Many location services have been proposed so far. *Greed Location Service*(GLS)[23, 6], *Simple Location Service*(SLS)[6] and *Reactive Location Service*(RLS)[6] are examples of location

services. In GLS, each mobile node periodically updates a set of location servers with its current location. The set of location servers chosen is determined by a predefined geographic grid and a predefined ordering of mobile node identifiers in the ad hoc network. A node using the Simple Location Service (SLS) transmits a *location packet*(LP) to its neighbors at a given rate. It also periodically receives a location packet from one of its neighbors. The node will then update its location table based on the received table entries, such that each location information with the most recent time (between the nodes own table and the received table) is maintained.

In position based routing algorithms the routing decision is usually made locally in each node. To evaluate most position based routing algorithms, the unit graph model (i.e. defined in Chapter 2) which is a widely accepted graph theoretical model for mobile ad hoc networks has been considered.

The parameters for evaluating the performance of a routing algorithm typically include *loop-freedom*, *demand-based operation*, *guaranteed delivery*, *memorization*, *hop count* and *scalability*.

- loop-freedom: A proposed position based routing algorithm may be classified as having or not having the loop freedom property. A loop free routing algorithm avoids cumbersome exit strategies like memorizing passed traffic or defining timeout for packets.

- demand-based operation: Instead of assuming a uniform traffic distribution in the network and maintaining routing information between all nodes at all times,

a demand based routing algorithm adapts to the traffic pattern on a demand or need basis.

- guaranteed delivery: If there is a route from a source node to a destination node, guaranteeing packet delivery is important. If the algorithm doesn't guarantee packet delivery, the source node should somehow determine whether the packet is reached the destination or not and try to resend it in the case of a failure.

- memorization: If a routing algorithm requires nodes to memorize route or past traffic, it will be sensitive to node queue size and node mobility while routing is ongoing. It is better to avoid memorizing past traffic at any node, if possible.

- hop count: There may be more than one route from a source node to a destination node. These routes may have different costs for a packet which traverses them. Hop count is the number of hops on a route from a source node to a destination node. Usually the routes with small number of hops are less costly and routing algorithms try to find routes with the same (or almost the same) number of hops as a shortest path. Hop count is an important parameter for evaluating a routing algorithm.

- scalability: A routing algorithms should perform well for wireless networks with arbitrary number of nodes. Some sensor networks may have hundreds or thousands of nodes. The overhead of scalable strategies is a linear function (i.e. instead of exponential for non-scalable strategies) of the number of nodes in the network.

The metrics that are used in simulations normally reflect the goal of the designed algorithm, and are naturally decisive in the route selection. Most routing schemes use hop count as the metrics. This choice of metric agrees with the assumption that nodes cannot adjust their transmission radii in order to reach desired neighbor with minimal power. It also assumes that the impact of congestion is not significant so delay is proportional to hop count. However, if nodes can adjust their transmission power (knowing the location of their neighbors) then the constant metric can be replaced by a power metric that depends on distance between nodes. As mentioned, in position based routing algorithms, the routing decision is made locally at each node and no routing information needs to be exchanged between nodes. Also these algorithms do not need to memorize past traffic. As a result, position based routing algorithms are very simple to implement and they usually scale quite well.

In the remainder of this chapter we explain some position based routing algorithms.

3.1 Direction and distance based methods

As a result of the availability of information about the position of points in a network, a forwarding node can make a locally optimal, greedy choice in choosing a packet's next hop. There are several routing protocols based on the greedy heuristic. These protocols differ in the way that they define local optimum to choose the next hop.

They have many advantages that make them very common in ad hoc wireless networks. In the following we review this family of routing protocols.

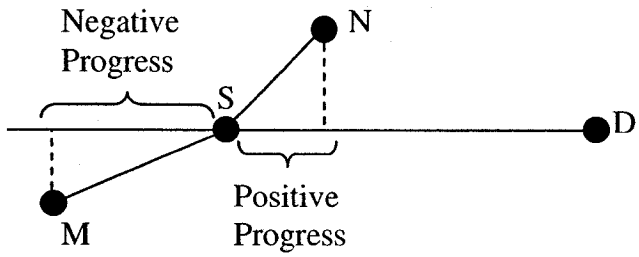

Figure 3: Node N has a positive progress and node M has a negative progress.

Progress is an important concept in many position based routing algorithms. For each node like S that receives a message for destination D, the progress of a neighbor N is the projection of line segment SN onto line SD. The progress can be positive or negative. See Fig. 3.

Local minimum is a common problem between most routing algorithms of this family. It happens when an algorithm follows a greedy strategy to select the next hop at each point and reaches a point where none of the neighbors could be selected as the next hop based on this strategy. For example if the routing strategy is to send a packet to a neighbor which is closer to the destination than the current node, a local minimum problem happens if a point is reached which doesn't have a neighbor that is closer to the destination than itself (see Fig. 4).

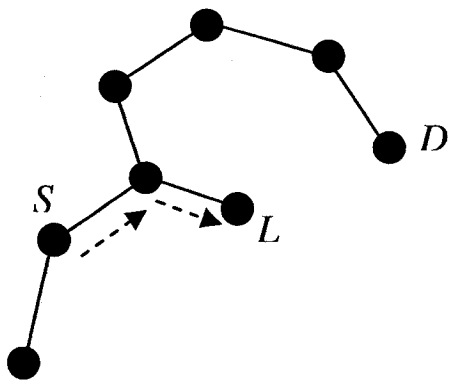

Figure 4: S is the source node and D is the destination node. Local minimum happens at node L.

3.1.1 Compass routing

In *Compass routing* (i.e. also known as DIR) [22] a node S that receives a message for destination D, calculates for every neighboring node N the angle between the line segments from S to D and S to N. The message is forwarded to a neighbor of S for which this angle is the smallest (See Fig. 5). Compass routing scales very well for big networks. This algorithm is not loop-free and doesn't use memory. It finds a single path from a source node to a destination node (i.e. in contrast to algorithms which find multiple paths). Compass routing doesn't guarantee packet delivery even if there is a path from the source node to the destination node (i.e. because of local minimum problem) [13].

3.1.2 Greedy routing

In this method [26, 13], a greedy principle is adopted to choose the successor node: Among the neighbors of the current node, choose the node that is closest to the

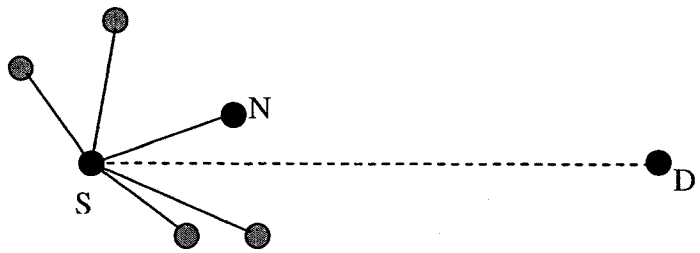

Figure 5: Node N will be selected as the next hop in Compass routing.

destination. When none of neighboring nodes is closer to the destination than the current node N, the algorithm differs from GEDIR studied later. One of the proposed solutions to this problem is searching all n-hop neighbors by limited flooding until a node closer to the destination than N is found [12], where n is a network dependent parameter. The algorithm has nontrivial details and does not guarantee delivery nor optimize flooding rate. It is proved that this algorithm has no loops since it always forces a message to take a step closer to the destination. In 2-hop greedy method node N selects the best candidate node C among its 1-hop and 2-hop neighbors according to the corresponding criterion. Then N forwards m to its best 1-hop neighbor in the set of neighbors of N and C. This basic idea is applicable also to most other methods. In alternate greedy method, the i'th received copy of m is forwarded to i'th best neighbor, according to the selected criterion (it fails if number of copies exceeds number of neighbors). In the disjoint greedy method, each intermediate node, upon receiving m, will forward it to its best neighbor among those who never received the message (it fails if no such neighbor exists). These methods reduce failure rate compared to greedy method, by memorizing past traffic. *MFR* is another variation of greedy routing which is explained in the next section.

MFR

MFR [27] was first proposed for Packet Radio Terminals. Given a packet and its final destination, a terminal transmits to the terminal most forward (with the greatest progress) in the direction of the final destination(see Fig. 8). If no terminals are in the forward direction, it transmits to the least backward terminal, if any. (A terminal cannot transmit to itself.) In case there are no terminals in the circle of radius R at all, it does not transmit in that slot. MFR is proved to be a loop-free algorithm.

Adjusting the transmission power to the distance between two nodes modifies the method. In this scheme, packet is sent to the nearest neighboring node with forward progress.

3.1.3 GEDIR

GEDIR [26, 13] is a variant of greedy routing algorithm. According to this method, a node forwards a message to the neighbor that is closest to the destination among the other neighbors. A message is dropped if the best choice for a current node is to return the message to the node the message came from. GEDIR is a loop free algorithm. This algorithm doesn't guarantee packet delivery. GEDIR scales well (i.e. however for large networks its delivery rate may decrease) and is very simple to implement.

3.1.4 Face routing

Face routing [5, 7, 2] (also called perimeter algorithm), constructs a planar and connected subgraph of the network graph and then applies routing along the faces of the subgraph that intersect the line between the source and the destination using a graph traversal method called *right-hand rule*. Right-hand rule states that when arriving at node x from node y, the next edge traversed is the next one sequentially clock-wise about x from edge (x,y)(see Fig. 6). A face is a polygon region and the right-hand rule traverses the interior of a face.

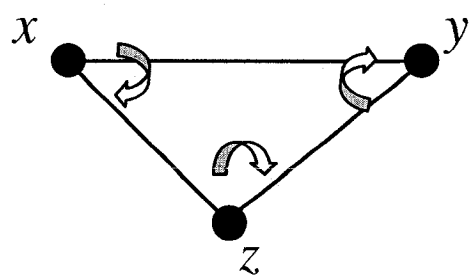

Figure 6: The right-hand rule. x receives a message from y and forwards it to the first neighbor counterclock-wise about itself, z.

Face routing is a scalable single path routing algorithm. In cases where it is possible to construct a planar and connected subgraph of the network graph (i.e. which is always possible for unit dist graphs), it guarantees packet delivery. This algorithm uses a constant-size memory and is loop-free.

3.1.5 GPSR

GPSR [20] is a combination of face routing and greedy routing algorithms. In this method, a packet can have two different modes, greedy-mode and perimeter mode.

Upon receiving a greedy-mode packet, a node searches its neighbors to see if there is a neighbor with positive progress. If such a neighbor exists, it will forward the packet to the neighbor which is closest to the destination, otherwise the packet will be marked into perimeter-mode. GPSR uses the right-hand rule (see Fig. 7) to forward perimeter-mode packets. When a packet enters perimeter mode at node S for destination D, it will be forwarded on progressively closer faces of the planar graph which are crossed by line SD. On each face, the traversal uses the right-hand rule to reach an edge that crosses line SD. At that edge, the traversal moves to the adjacent face crossed by SD. When a perimeter-mode packet enters a node which is closer to D than S, it will be marked into greedy-mode again.

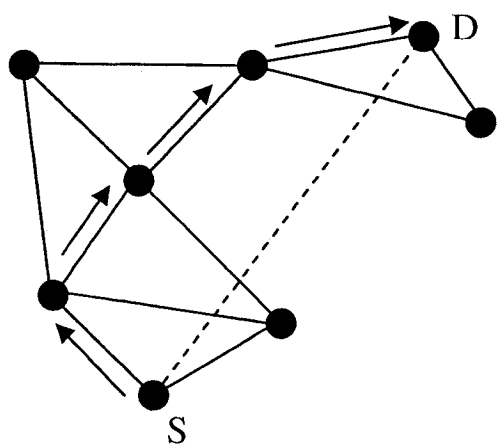

Figure 7: Graph traversal in GPSR using right hand rule.

GPSR scales very well for big networks. This algorithm uses constant-size memory and is loop-free. Like face routing, in cases where it is possible to construct a connected and planar subgraph of the network graph, it guarantees packet delivery. GPSR is a single path routing algorithm.

3.1.6 Random progress method

Random policies [24, 15] have been proposed for packet radio networks. In one variation, packets destined toward destination node D, are routed with equal probability towards one intermediate neighboring node that has positive progress. These methods are not loop-free and don't guarantee packet delivery. They scale well for big networks.

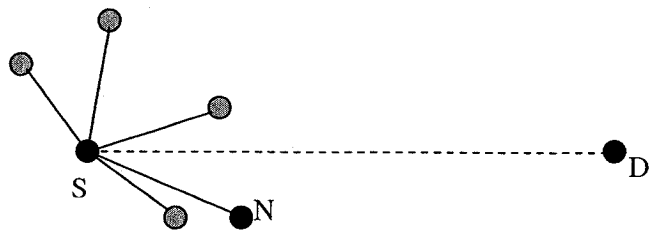

Figure 8: Node N will be selected as the next hop in MFR algorithm.

3.1.7 NFP and NC

As we mentioned, in MFR routing algorithm, a message is sent to the neighbor with maximum progress. However, it also consumes maximum power to cover maximum distance, which in turn increases collisions with other nodes. Hence, instead of forwarding packets to the farthest neighboring node, the NFP (Nearest with Forward Progress)[16] scheme sends the packet to the node closest to the sender. NC (Nearest Closer)is a variant of NFP method in which the current node selects one of its closest neighbors that are closer to the destination than itself. The transmission can thus be accomplished with minimum power and the interference with the other nodes is minimized, while the probability of a successful transmission is maximized.

3.2 Partial flooding methods

In this family of position based routing methods, the information about the position of nodes is used to reduce the number of produced control messages by limiting the flooding to some nodes (i.e which are for example closer to the destination).

Partial flooding routing algorithms require nodes to memorize past traffic to avoid forwarding the same message multiple times. DREAM [4], V-GEDIR [25], LAR [21], CH-MFR [25] belong to this family of routing algorithms.

3.2.1 DREAM

In DREAM, messages are forwarded to the neighbors whose directions belong to a specific range. This range is determined by some factors including the maximum possible movement of the destination node and the position of the current and destination nodes.

3.2.2 LAR

In LAR algorithm (i.e. abbreviation of Location Aided Routing) [21], a request zone is defined and nodes, which are not in the request zone, do not forward a route request to their neighbors. This request zone is consisted of a circle centered at the destination node and the area between the tangents to that circle from the current node. The

maximum possible movement of the destination node determines the radius of this circle. The source or an intermediate node N will forward a message to all nodes that are closer to the destination than N.

Chapter 4

Ant colony optimization based routing algorithms

Construction algorithms build solutions to a problem in an incremental way starting with an empty initial solution and iteratively adding solution components until a complete solution is obtained. Often best results are obtained if a heuristic estimate of solution components is taken into account. *Ant colony optimization* (ACO) is a stochastic approach for solving combinatorial optimization problems like routing in computer networks. The idea of this optimization is based on the observation of how ants optimize food gathering in nature. In many species of ants, individuals deposit a particular chemical substance called *pheromone* on the ground while walking. By depositing pheromone and making a trail, they mark a path from the nest to a food source. Foragers can sense pheromone trails and follow the paths which have been discovered by other ants to reach the food source. Most importantly, they can exploit

pheromone trails to choose the shortest path among the available paths from the nest to the food source.

Ant colony optimization routing algorithms [8, 10, 14, 17, 31, 9, 28] use artificial ants to iteratively construct a solution for the routing problem as an optimization problem. We can explain an ant colony optimization algorithm in a graph G informally as follows. A pheromone trail and a heuristic pheromone value is associated with each edge of G. A folk of ants move on the adjacent nodes of G concurrently and asynchronously to find an optimum solution. Each ant selects the next hop by making a stochastic decision using the existing pheromone trails and heuristic information. The solution is built incrementally as the ants move on the edges of the graph. While moving on the edge between two nodes and building a solution, an ant evaluates this solution and deposits pheromone on that edge. This pheromone trail will be used by future ants to make a routing decision. Pheromone evaporation is a process that causes the amount of pheromone which is deposited on the links of the graph to be decreased over time. Regarding a set of termination conditions, T, an ant stops when one of the termination conditions in T is satisfied. In some cases an ant retraces its path backward and modifies the pheromone values after it finds a solution.

In the remainder of this chapter we will explain some ACO routing algorithms and then discuss the performance of these algorithms.

4.1 AntNet

AntNet [8] is a proactive ACO routing algorithm for packet switch networks. In this algorithm, a forward ant is launched from the source node at regular intervals. A forward ant at each intermediate node selects the next hop using the information stored in the routing table of that node. The next node is selected with a probability proportional to the goodness of that node which is measured by the amount of pheromone deposited on the link to that node. The probability is calculated using Formula 1.

$$p_i = \frac{\phi_i}{\sum_{j=1}^{k} \phi_j} \tag{1}$$

In the above formula, ϕ_i represents the amount of pheromone trail deposited on the edge to the ith neighbor and p_i is the probability that the ith neighbor is selected as the next hop.

For each forward ant, AntNet pushes the identifier of every visited node and the time elapsed since its launching time to arrive at this node onto a memory stack carried by the ant. If a cycle is detected, that is, if an ant is forced to return to an already visited neighbor node, the cycle's nodes are popped from the ant's stack and all the memory about them is destroyed.

When a forward ant reaches the destination, it generates a backward ant which takes the same path as the corresponding forward ant but in opposite direction. The backward ant updates pheromone values as it moves in its way to the source node. Arriving in a node k coming from a neighbor node f, the backward ant increases

the amount of pheromone deposited on edge $\vec{kf}$ by an amount which is a decreasing function of the traveled distance from the source node (i.e. this is the reason for popping nodes of a cycle from an ant's stack).

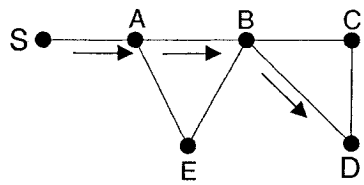

Figure 9: A forward ant is sent from the source node(S) to the destination node (D).

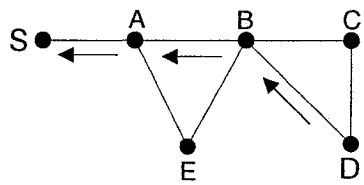

Figure 10: A backward ant is sent from the destination node(D) to the source node (S).

AntNet is a multi-path routing algorithm which means each node maintains several paths to a certain destination. In AntNet, the routing table and the statistic information of a node are local and they are not transmitted to any other node.

4.2 ARA

ARA [14] is a reactive ACO routing algorithm for mobile ad hoc networks. ARA has two phases: route discovery, and route maintenance. In route discovery phase, the sender broadcasts a forward ant. The ant is relayed by each intermediate node until it reaches the destination. After receiving a forward ant at the destination, the

ant is destroyed and a backward ant is sent back to the sender. The backward ant increases the pheromone value corresponding to the destination in each intermediary node until reaches the sender. When the sender receives a backward ant, the route maintenance phase starts by sending data packets. Since the pheromone track is already established by the forward and backward ants, subsequent data packets will perform the route maintenance by adjusting the pheromone values.

In ARA, each forward and backward ant has a unique sequence number so these packets do not generate loops. ARA supports sleep period operation, nodes are able to sleep when their amount of pheromone reaches a threshold. Other nodes will then not consider these nodes. ARA is a multi-path routing algorithm.

4.3 ANTHOCNET

ANTHOCNET [10] is a hybrid routing algorithm that combines reactive route establishment with proactive route maintenance. Where there is a demand for a route, the algorithm first checks the routing table of the source node to see if there is any information about the destination. If there is no routing information for the destination, it will broadcast a forward ant. Due to this broadcasting, each neighbor receives a copy of the ant and checks its routing table. Again, the ant will be broadcasted if there is no routing information for the destination, otherwise it will be sent to the next hop using a stochastic decision. Each forward ant keeps a list of the visited nodes

and when it reaches the destination, uses this list to take the same path back to the source and update the pheromone values deposited in the links. To avoid generating a huge number of ants of the same generation, a limit, a, is defined and when the length of the traveled path exceeds this limit the ant will be discarded. Whenever an intermediate node receives two ants of the same generation, checks the length of the traveled path of both ants and discards the ant with longer traveled path if the difference is more than a certain value, α_1. This algorithm uses a stochastic scheme similar to that used in ANTNET for routing data packets. While a data session is running proactive ants will be sent to perform route maintenance.

4.4 The performance of ACO routing algorithms

In general, ACO routing algorithms guarantee message delivery [8, 14, 29]. They converge to a route which is very close to the optimum route. The time it takes for an ACO routing algorithm to converge to a route is called *convergence time*. In most cases AntNet has a long convergence time. As mentioned above, AntNet launches an ant at regular intervals. The more the number of ants needed to converge to a route, the more the time elapsed since the start of the algorithm. ANTHOCNET and ARA broadcast an ant in the route discovery phase and have relatively short convergence times. Although an ant is a small control packet, broadcasting implies an overhead on the network and when the size of the network is too large this overhead becomes a sever drawback. In ANTHOCNET, since the ant contains a list of the visited

nodes, its size will grow as it goes far from the source and the routing overhead will be increased. As a result ARA and ANTHOCNET are not scalable. However the amount of the generated control traffic in AntNet is small (i.e. because it launches one ant at each time), the long convergence time for large networks is a main drawback which makes this algorithm non-scalable.

From the above mentioned algorithms, AntNet and ANTHOCNET are simulated and compared to the proposed algorithms in this thesis.

Chapter 5

POSANT routing algorithm

In this section we define POSANT (Position Based Ant Colony Routing Algorithm) [19], an ant colony routing algorithm which uses location information to improve its efficiency. POSANT is able to find optimum routes when a given network contains nodes of different transmission ranges. As stated before, the algorithm is reactive, thus a route is searched for only when there is a collection of data packets that are to be sent from a source node S to a destination node D. Sending the data packets will start after a route from S to D is established. Before that, only forward and backward ants are being exchanged. In order to minimize the time that POSANT needs to find a route while keeping the number of generated ants small, information about the position of nodes is used as a heuristic value. In the next section we introduce the concept of zones which play an important role in our algorithm.

5.1 Zones

Consider a destination node D and a network graph G. For each node S (i.e. S is not necessarily the source node) we partition its neighbors into 3 zones called $zone_1$, $zone_2$ and $zone_3$. Consider a line segment between S to D. For a neighbor H of S angle θ_H is defined as the angle between line segments SH and SD. Node H belongs to $zone_1$ if $\theta_H \leq \pi/4$,$zone_2$ if $\pi/4 < \theta_H < 3\pi/4$, and $zone_3$ if $3\pi/4 \leq \theta_H \leq \pi$, see Figure 11.

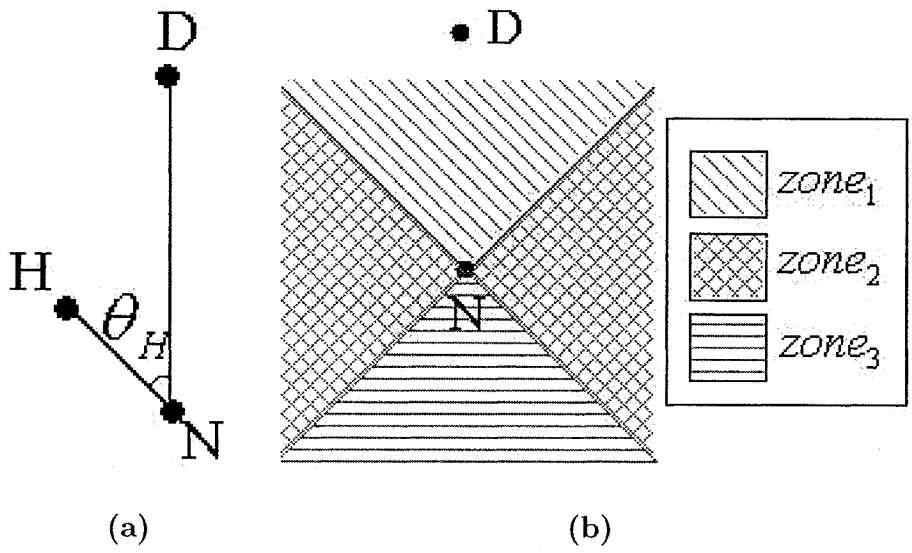

Figure 11: (a) θ is the angle between SH and SD. (b) Different zones of N for destination node D.

5.2 Pheromone initialization

Before starting the route establishment step, the amount of pheromone deposited in the edges of the graph must be initialized. For each node like A in the network, a

pheromone trail is assigned to each of the outgoing links. Suppose B is one of the neighbors of A. To initialize the value of the deposited pheromone on $\vec{AB}$ we follow a greedy policy, the assigned values decrease with the zone number. More formally, having three values ν_1, ν_2 and ν_3 such that $\nu_1 \geq \nu_2 \geq \nu_3$, an amount of pheromone equal to ν_i will be assigned to $\vec{AB}$ if B belongs to $zone_i$. The motivation is that in most cases shortest paths pass through the neighbors whose directions are closer to the direction of the destination. As a result, this greedy initialization causes a faster convergence to a shortest path most of the time. Our experiments show that the value of ν_1, ν_2 and ν_3 has a significant effect on the performance of POSANT.

POSANT assumes that each node maintains a table of the values of pheromone trails assigned to its outgoing links for different destinations. Whenever a node receives a packet for a specific destination, it will check its table to see if there is at least one pheromone trail for that destination. If such pheromone trail exists, it will be used for making a stochastic decision (which will be explained in details later) to choose the next hop. If it doesn't exist, the pheromone initialization process begins and assigns pheromone trails to all the outgoing links. As just explained, the amount of the deposited pheromone on each link depends on the zone of the corresponding neighbor. The pheromone trails for a specific destination will be deleted from the pheromone trail table of a node if the node doesn't receive any packet pointing to that destination for more than a specific time which is defined to be in the order of seconds. The entries in the BR table which are related to that destination will also be deleted after that time. BR table is explained in the next section.

Algorithm 1 POSANT routing algorithm

```
{S is the source node, D is the destination node and C is the current node}
for each clock time do
  if C = S then
    if one of the conditions in Section 5.4 is true then
      send one datapacket from each of the three zones of C
    else
      send one forward ant from each of the three zones of C
    end if
  end if
  for each message m in C's buffer do
    if (m→type = ForwardAnt) or
      (m→type = DataPacket) then
      NextHop = SelectNextHop(n)
      send m to NextHop
      if NextHop = D then
        m →type = BackwardAnt
      end if
    else if m→type = BackwardAnt then
      find NextHop in C's BackRouting table
      send m to NextHop
      IncreasePheromone(NextHop,m)
      if NextHop = S then
        update averages of packet delays for the corresponding zone
        drop m
      end if
    end if
  end for
  Evaporate()
end for
```

Algorithm 2 SelectNextHop

Input: node N

for i = 1 to the number of N's neighbors **do**

$p_i = \frac{\phi_{gr_i}}{\sum_{j=1}^{k} \phi_{gr_j}}$

return neighbor i with probability p_i

end for

5.3 Route establishment

Consider a destination node D and a source node S with its three zones. To establish a route, S launches n forward ants with unique sequence numbers from each zone at regular time intervals ($3n$ ants each time) (i.e. see Algorithm 1). In our experiments we set n to be 1. Assigning very large values to n increases the overhead of the algorithm without resulting to a significant improvement. Similar to other ACO routing algorithms, at each node a forward ant makes a stochastic decision which is based on the values of pheromone trails to select the next hop. As mentioned before, the values of pheromone trails are stored in a table at each node. Suppose that a forward ant is currently residing in node N and this node has k neighbors $H_1, H_2, ..., H_k$, and ϕ_i is the amount of pheromone assigned to $N\vec{H}_i$. The forward ant will select H_i as the next node with a probability p_i which is calculated using the following equation (i.e. see Algorithm 2).

$$p_i = \frac{\phi_i}{\sum_{j=1}^{k} \phi_j} \tag{2}$$

In addition to the pheromone trial table discussed before, each node maintains another table which we call Back Routing (BR) table. Whenever a forward ant enters a node from one of its neighbors, an entry in the BR table will be created that stores the identifier of the neighbor which the forward ant is coming from, the sequence number of the ant and the identifier of the destination. Repeated forward ants will be destroyed. When a forward ant reaches the destination, it is destroyed and a backward ant is sent back to the source. This backward ant has the same sequence number as the corresponding forward ant and traverses the same path to the source using the information stored in BR tables. Moving from node B to node A, the backward ant increases the amount of pheromone stored in $\vec{AB}$ using the following formula.

$$\phi_{\vec{AB}} = \phi_{\vec{AB}} + \omega(\vec{AB}) \cdot g(d) \tag{3}$$

In the above formula d is the length of the traveled path from the destination to node B by the backward ant and $g(d)$ is a decreasing function of d. Function ω is a *weight function* and its value is dependent on the zone of B as given in Formula 9.

$$\omega(\vec{AB}) = \begin{cases} w_1 \geq 1 & \text{if B is in } zone_1 \text{ of } A \\ w_2 = 1 & \text{if B is in } zone_2 \text{ of } A \\ w_3 \leq 1 & \text{if B is in } zone_3 \text{ of } A \end{cases} \tag{4}$$

Using the above weight function yields a faster convergence in most cases because the shortest path usually passes through the nodes which are closer in direction to the destination. The motivation is similar to the one in the greedy initialization.

An evaporation process causes the amount of pheromone deposited in each link to decrease as the time passes on. It is done by multiplying the current pheromone value by a number $\mu < 1$ at regular intervals.

$$\phi(\vec{BA}) = \mu \cdot \phi(\vec{BA}) \tag{5}$$

Sometimes especially at the beginning of route establishment process, some ants take non-optimum routes to reach the destination. At their way back to the source they will update pheromone trails. Pheromone evaporation, reduces the effect of these updates as the time passes on.

The above stochastic strategy establishes multiple paths between the source and destination. As a result, in contrast to regular position based routing algorithms which usually find a single route to the destination, POSANT is a multipath routing algorithm (i.e. like other ACO routing algorithms). A big advantage of multipath routing algorithms is that they reduce the chance of congestion in the network.

5.4 Sending data packets

As stated before, to establish routes to the destination, the sender launches n forward ants to each of its three zones at regular intervals. The modification of pheromone values by backward ants and the evaporation of pheromone cause a higher amount of pheromone to be deposited in some links than the others. It means that the algorithm converges to some routes. After that sending data packets should be started. It is important to start sending data packets instead of control packets at an appropriate

time. If we start sending them too early, they may be lost or follow long routes because optimum routes are not established yet. On the other hand, sending them too late increases data delivery delay. To estimate the best time to start sending data packets we do as follows.

For each of the three zones of the source node for destination D, the average and standard deviation of the delays reported by backward ants is calculated. As said before, the number of nodes on a path taken by an ant to reach the destination is the delay of that ant. Each backward ant carries the length of the path passed from the destination to its current hop. Whenever a backward ant is received by the source node, we update the average and standard deviation of packet delays for the corresponding zone using the delay reported by this ant. To reduce the effect of old backward ants, we define a fixed size *window* for each zone that contains recently received backward ants from that zone. The average and standard deviation of delays will be calculated only for the backward ants in the window. When a new backward ant is received we put it in the window of the corresponding zone and discard the oldest ant when the widow size has been reached. Selecting an appropriate window size is important. If the window size is too small, the average delay calculated from the window information would be too far from the real average. If the window size is very big, existence of very old ants would affect the result for a long time. Suppose α_i and σ_i are the average and standard deviation of the delays reported by the backward ants received from $zone_i$ and residing in the window.

- If σ_i is less than a threshold say t, we stop sending forward ants to $zone_i$ and

start sending data packets instead.

- If $\sigma_i, \sigma_j < t$ and $\alpha_j > \alpha_i + c$, we stop sending forward ants or data packets to $zone_j$. In this formula c is a constant value which determines how different the length of the established routes can be.

If σ_i is small enough we can assume that almost all the ants launched from $zone_i$ are following routes with almost the same length to the destination. It means that the algorithm has converged to a route or a group of routes with similar lengths. Thus it is a reasonable time to start sending data packets from $zone_i$.

If the second condition is true it means that the algorithm will not converge to a route (or a group of routes with almost the same length) passing through $zone_j$ which length is shorter than or equal to the average length reported by the ants launched from $zone_i$ plus a constant value (c). In this case the algorithm does not use routes passing through $zone_j$ anymore and stops sending data packets or ants from this zone. It is done by removing all the pheromone trials assigned to the outgoing links in $zone_j$ for destination D from the pheromone trail table of the source node. Assigning big values to c causes the algorithm to establish multiple paths to the destination but choosing very big values may allow non-optimum routes to be used.

In [8], [11] and [29] it is shown that the ant colony policy eventually converges to a route or group of routes with the same length, so the first condition becomes true. In the next section we show that sending data packets is started quite soon in practice.

Data packets follow the same process as the forward ants.

A precise selection of the parameters of POSANT is discussed in Section 5.6

5.5 Failure recovery

If, due to the mobility of nodes or any other reason, a link between two nodes say A and B breaks while a connection is running between a source S and a destination D, POSANT does as follows. For each node, POSANT defines a mode that can have two values, *regular mode* or *broken mode*. Initially each node is in regular mode. Whenever a link to one of the neighbors breaks, the mode of this node changes to broken mode. Suppose A realizes that the link to B is broken and there is a pheromone trial corresponding to link $\vec{AB}$ for D in the pheromone table of A. In this case the stochastic data routing will continue but if there is no pheromone trail for D corresponding to any of the other outgoing links of A, A sends a message to its neighbors to inform them that there is no route to D from A. Upon receiving this message, these neighbors do the same as if the link to A is broken. To avoid a loop, if A receives a packet for the second time while it is in broken mode, sends a message to its neighbors to inform them there is no route to the destination from A. If the source node has only one outgoing link that contains a pheromone trail for D, and this link breaks or a message from this link is received that states there is no route to D, a new route establishment process will begin and sending data packets will be suspended until a new route is found. The mode of A will be changed to regular mode after a specific time (which is defined to be in the order of milliseconds) is passed from the

moment that the link failure is detected.

5.6 Performance evaluation

In this section we investigate the behavior of POSANT with different parameter settings and compare the performance of POSANT, ANTNET, ANTHOCNET and GPSR. The comparison is made based on the simulation results of these algorithms. The simulation program was written in VC++ environment. In our experiments we used network graphs with 80 nodes randomly and uniformly spread over a square of length 500 units. The transmission range of each node is randomly selected for different directions and its value is between 50 and 70 units. Two nodes are connected if each one is in the transmission range of the other one. In our experiments we tried the routing algorithms only on connected graphs which are generated as mentioned. If the network graph is not fully connected, the source and destination may reside in different components which means there is no route to the destination. To make a graph connected, we find disconnected nodes and move them to a random point and check again the connectivity of the graph. To estimate the performance, we ran the algorithms on 200 networks and averaged the results. For each network, one source-destination pair is selected randomly. The algorithms are evaluated in terms of *delivery rate*, *packet delay* and *overhead* (i.e. the number of generated control packets).

Table 1 lists the parameters of POSANT and their corresponding values used in

Table 1: Parameters defined in POSANT and their values.

Parameter	Value	Description
t	1	standard deviation threshold
(ν_1, ν_2, ν_3)	(20,1,1)	used in pheromone initialization
μ	0.95	used in pheromone evaporation
window size	50	
$g(d)$	$\frac{1000}{d^2}$	used in equation 5

Table 2: Parameters defined in ANTHOCNET and their values.

Parameter	Value	Description
μ	0.95	used in pheromone evaporation
α_1	2	
a	20	used in pheromone evaporation

our experiments. We found that POSANT performs better with this setting after trying different values assigned to the above parameters. The parameters of ANTHOCNET and their corresponding values are listed in Table 2.

For ANTNET algorithm, the evaporation rate was set to 0.95, the same as for POSANT.

The next subsections are about the results of the simulation program.

5.6.1 Selection of the parameters

In this section, we study the influence of different parameters of POSANT on its performance. We tried POSANT with different parameter settings on a set of randomly generated graphs. Among the parameters listed in Table 1, the effect of μ, t and ν_is

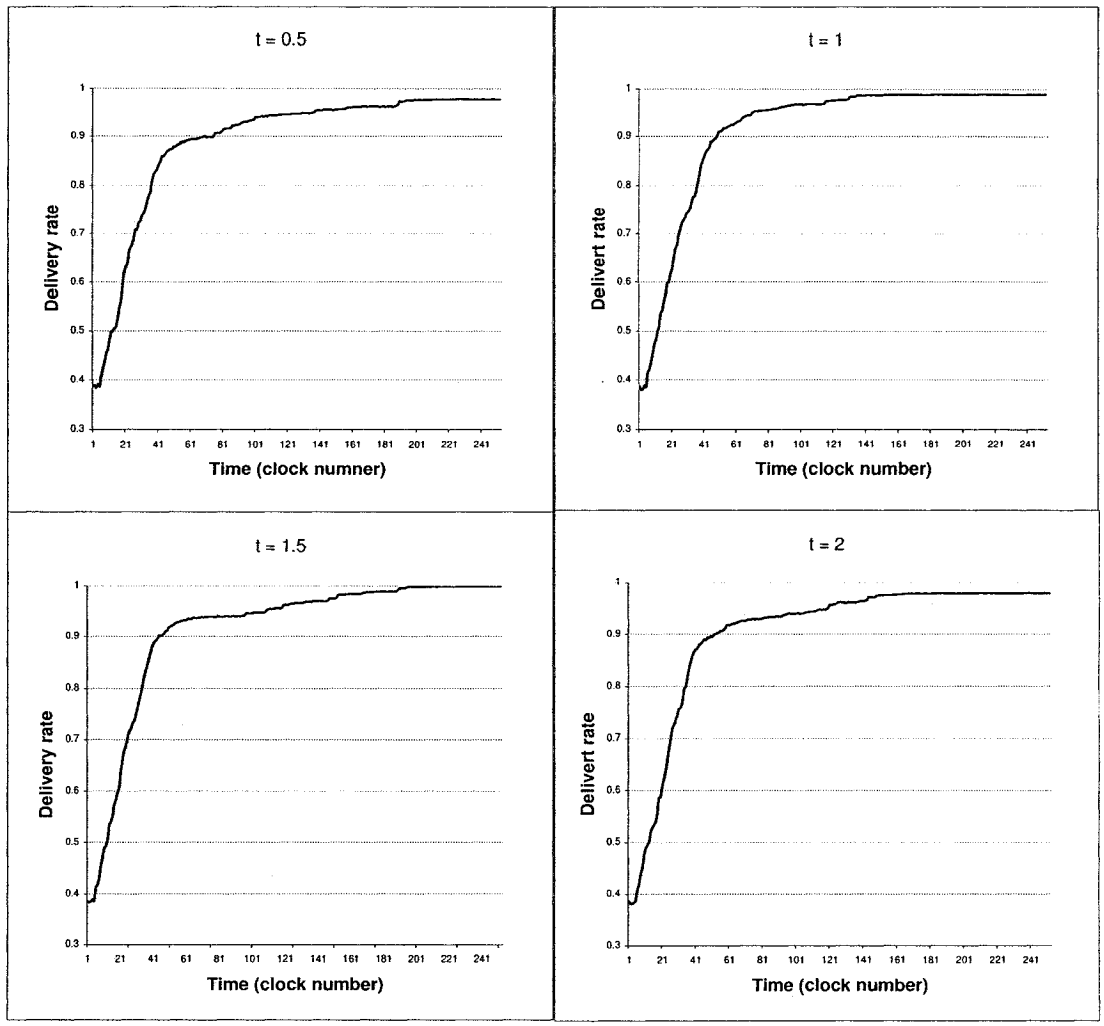

Figure 12: Average delivery rate of POSANT with different values assigned to parameter t.

on the performance of POSANT is investigated. In the following comparisons, except the parameters whose effect is studied, the other parameters have the values listed in Table 4.

In Figures 12, 13, and 14 the effect of t, the threshold of the standard deviation of packet delays, on the performance of POSANT is studied. As these figures show, for $t = 1$ and $t = 1.5$ the performance of POSANT is the best. For $t = 0.5$ and $t = 2$,

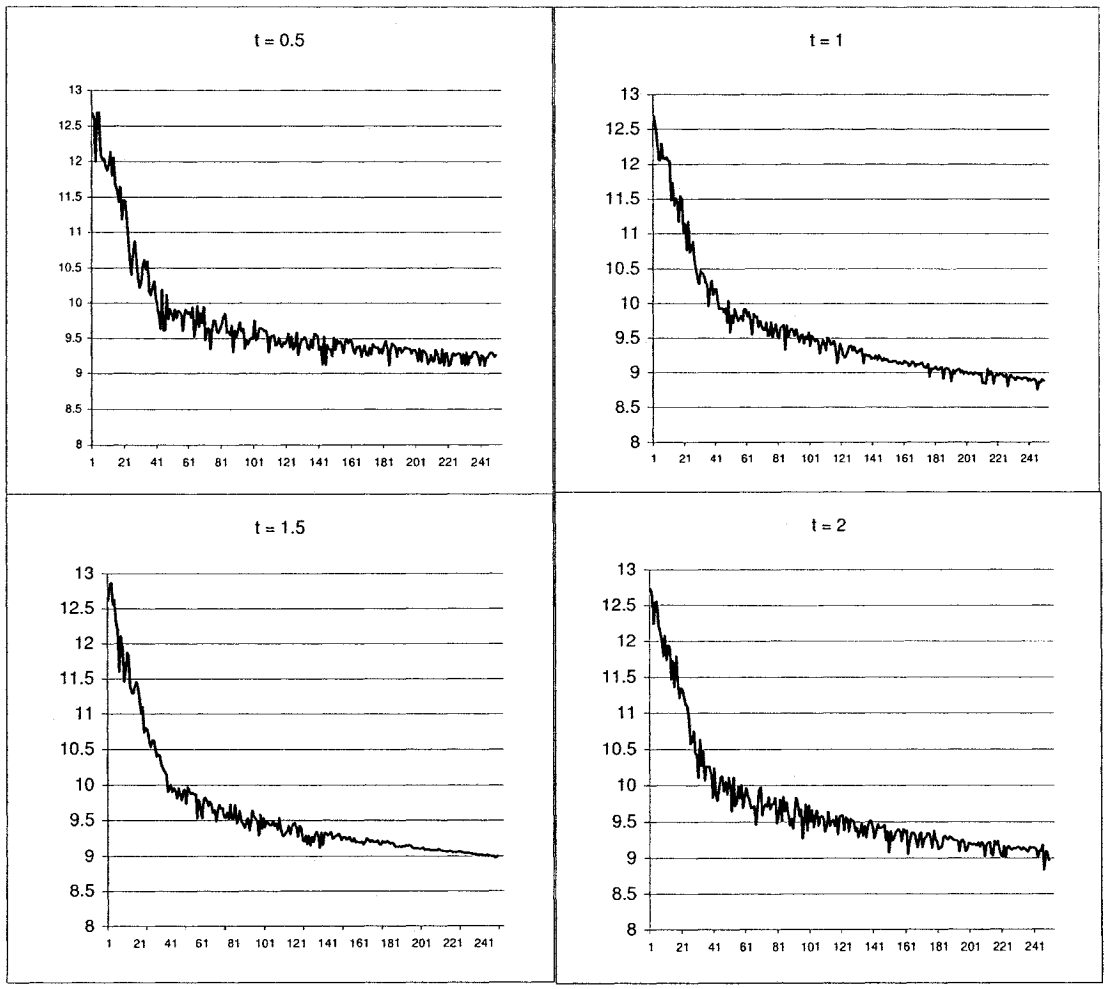

Figure 13: Average packet delay in POSANT with different values assigned to parameter t.

the graphs have relatively high fluctuations.

Figures 15, 16, and 17 show the effect of μ on the performance of POSANT. The performance of POSANT with $\mu = 95\%$ is the best.

The values of ν_1, ν_2 and ν_3 have a significant effect on the performance of POSANT. In the next subsections we ran POSANT with different values assigned to these parameters to compare the results. POSANT(a,b,c) denotes a POSANT algorithm with

ν_1, ν_2, ν_3 set to a,b,c.

5.6.2 Delivery rate

If a routing algorithm fails to deliver a packet to the destination, the sender must somehow detect this failure and try to resend the packet hoping that this time the algorithm delivers it. This increases the traffic in the network and also the delay experienced by the receiver. Thus it is very important to guarantee high delivery rate. Delivery rate is the ratio of the number of packets received by the destination node to the number of packets sent by the source node. In Figure 18 the average delivery rate of ANTNET, ANTHOCNET and POSANT is compared. This figure shows how the average delivery rate varies with time. As the time progresses, the delivery rate increases and eventually becomes almost 100% for all algorithms except GPSR. As the graph shows, ANTHOCNET reaches 100% delivery rate faster than the others. This is the result of broadcasting ants in this algorithm. GPSR always has a relatively low delivery rate. It is because this algorithm fails in some cases as a result of varying transmission ranges of the nodes. POSANT(20,1,1) has the highest delivery rate among the others. ANTNET and POSANT(20,20,20) reach to 100% delivery rate slower than the other ant based algorithms.

5.6.3 Convergence time

In this part the average packet delay of POSANT is compared with that of ANTNET, ANTHOCNET and GPSR. The results of this comparison is presented in Figure 19.

As mentioned before, hop count is used as the metric to measure packet delays. Figure 19 shows how the average packet delays for the different algorithms vary by time. In the beginning, the average packet delay of ANTHOCNET is smaller than the others. GPSR has a relatively small average delay at the beginning but it is still longer than that of ANTHOCNET. It is because GPSR doesn't find the shortest path in many cases. The average delay in POSANT(20,1,1) reaches its minimum faster than the others. As the time passes on, all the algorithms except GPSR converge to the paths with almost the same lengths and the packets experience almost the same delay.

POSANT decides that a convergence to a route has been achieved when the standard deviation of packet delays is less than a threshold value. Since POSANT is reactive, the convergence time is important. Even for a proactive routing algorithm like ANTNET the convergence time should be small because the topology of the network is changing very often and a new route to the destination should be found as fast as possible. In Figure 20 the average standard deviation of the packet delays is shown. The average standard deviation of POSANT(20,1,1) and ANTHOCNET is smaller than the others in the beginning. A small standard deviation means that the packets are traveling paths with almost the same length.

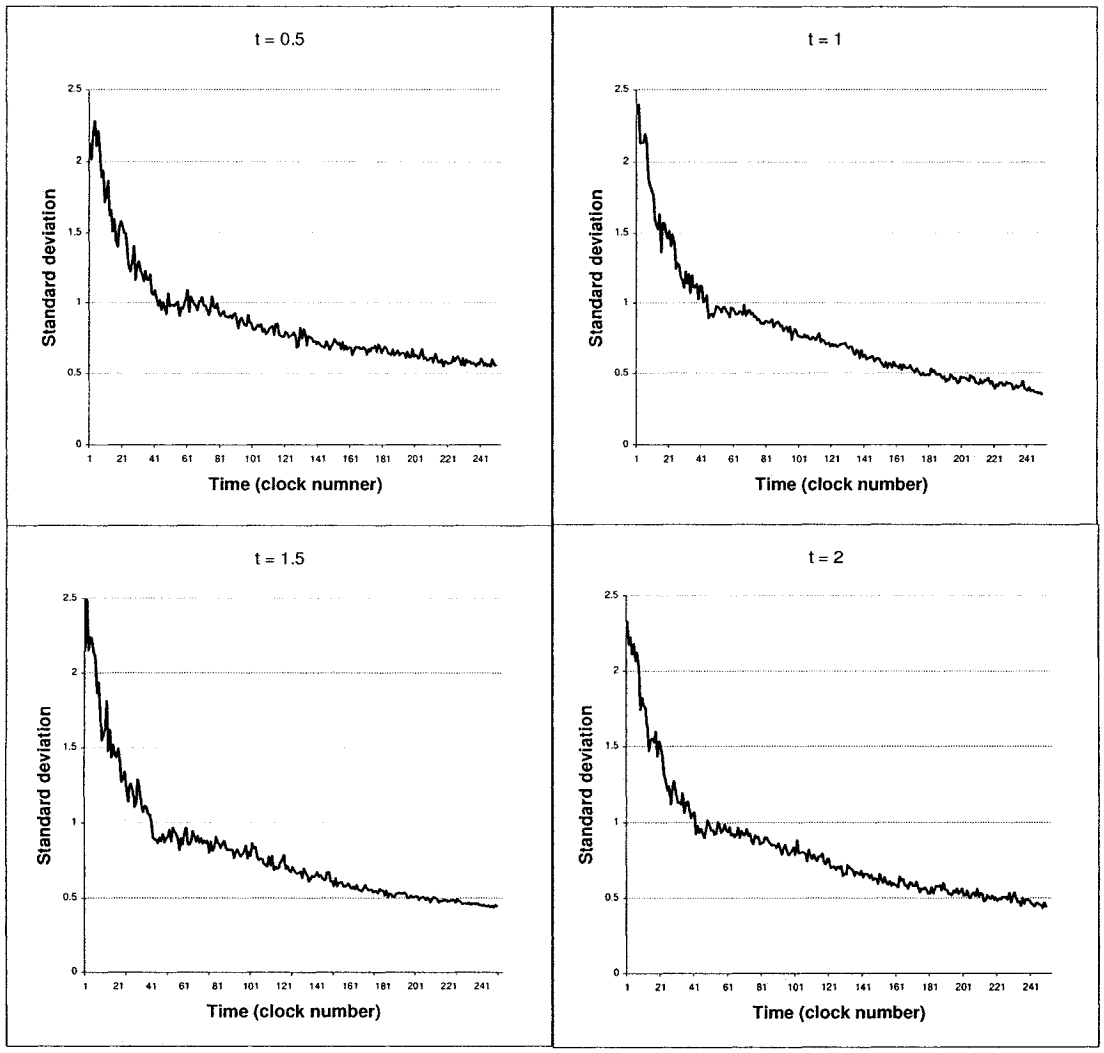

Figure 14: Standard deviation of packet delays in POSANT with different values assigned to parameter t.

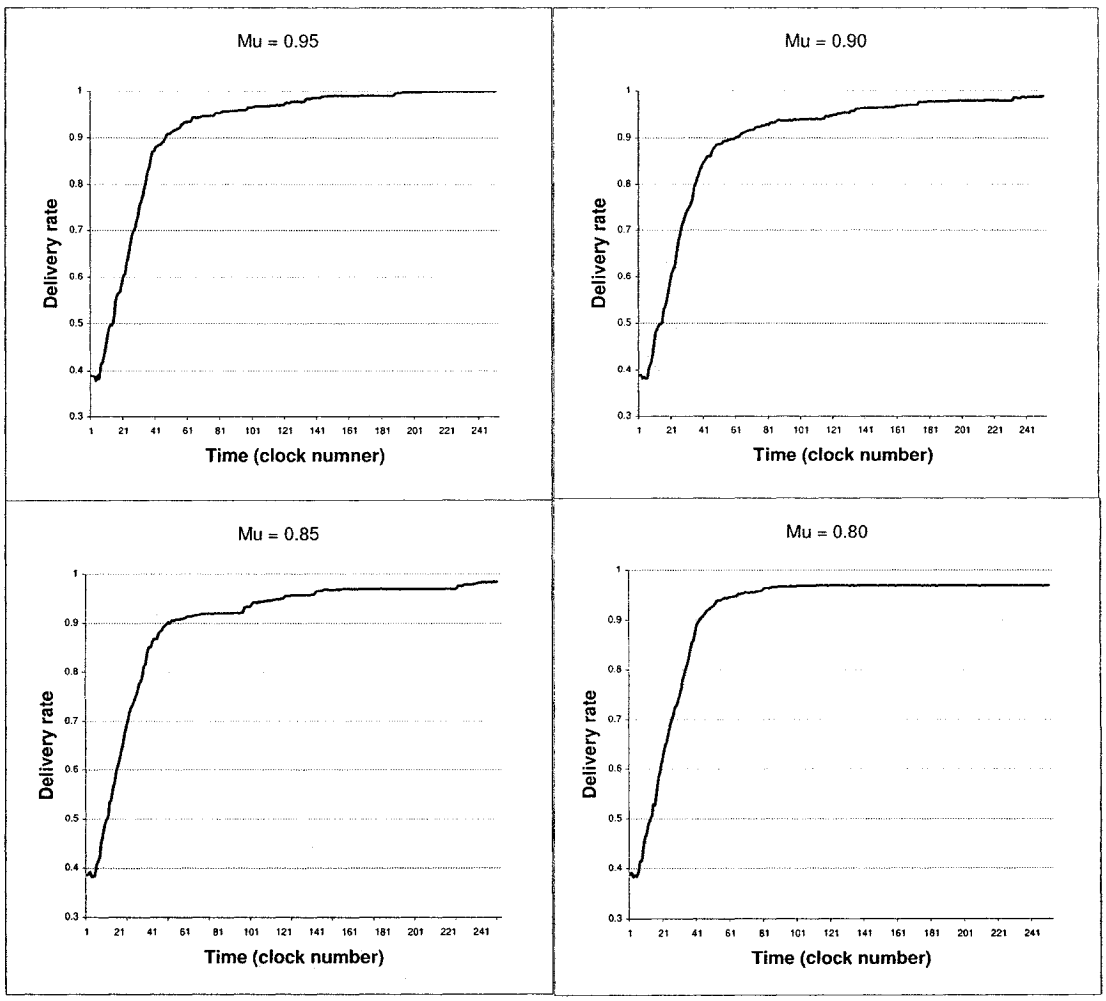

Figure 15: Average delivery rate of POSANT with different values assigned to parameter μ.

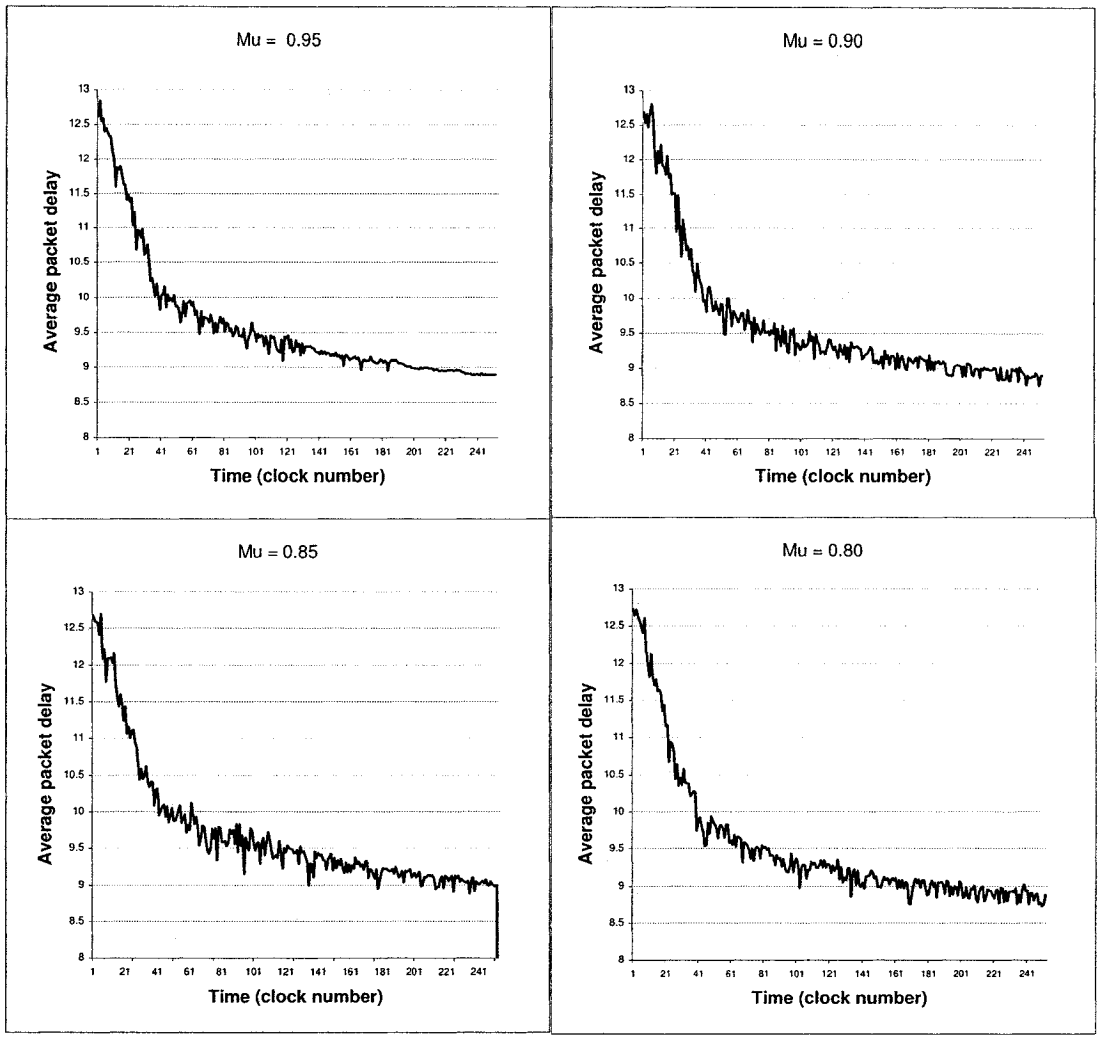

Figure 16: Average packet delay in POSANT with different values assigned to parameter μ.

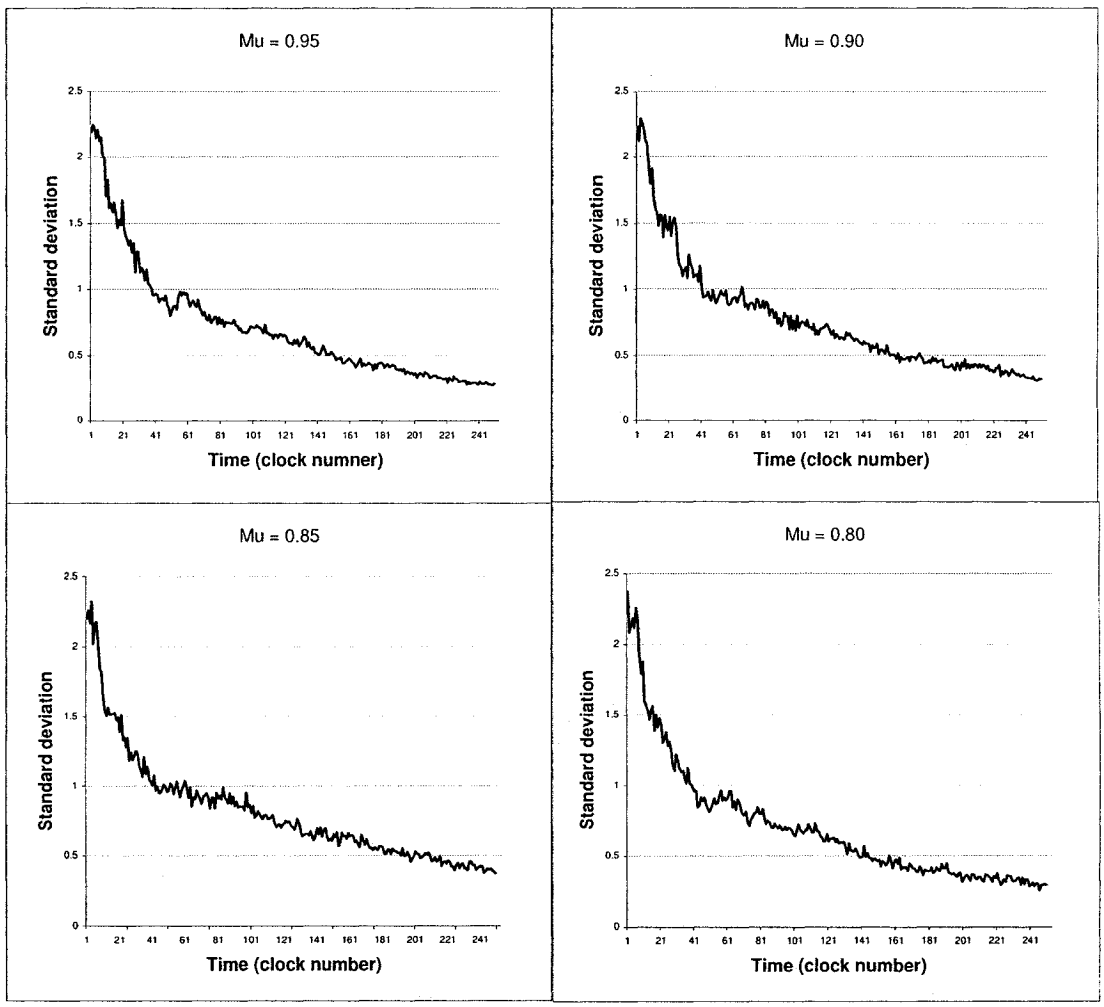

Figure 17: Standard deviation of packet delays in POSANT with different values assigned to parameter μ.

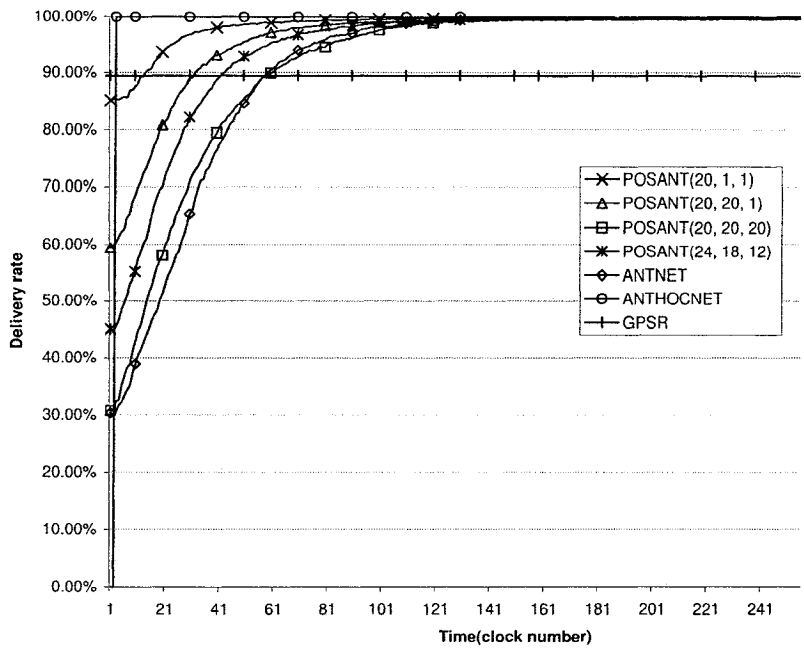

Figure 18: Average delivery rate of ANTNET, POSANT, ANTHOCNET and GPSR.

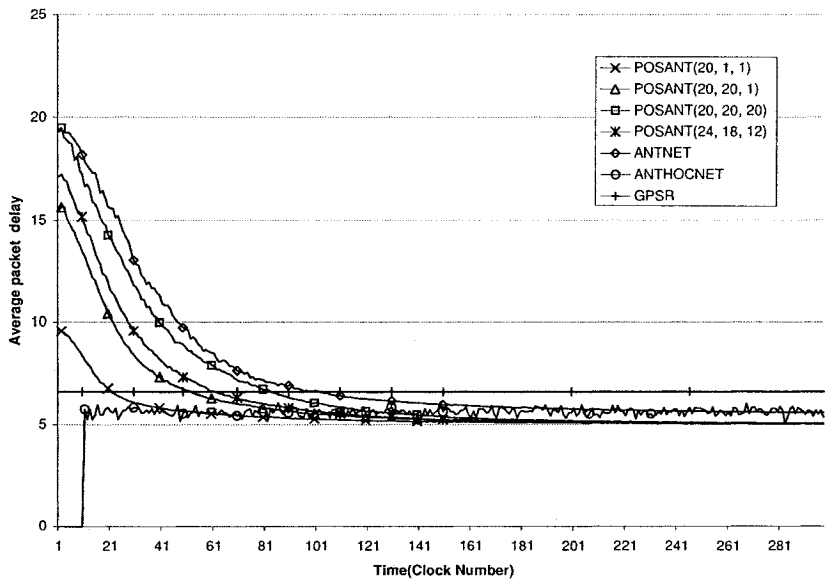

Figure 19: Average packet delay of ANTNET, POSANT, ANTHOCNET and GPSR.

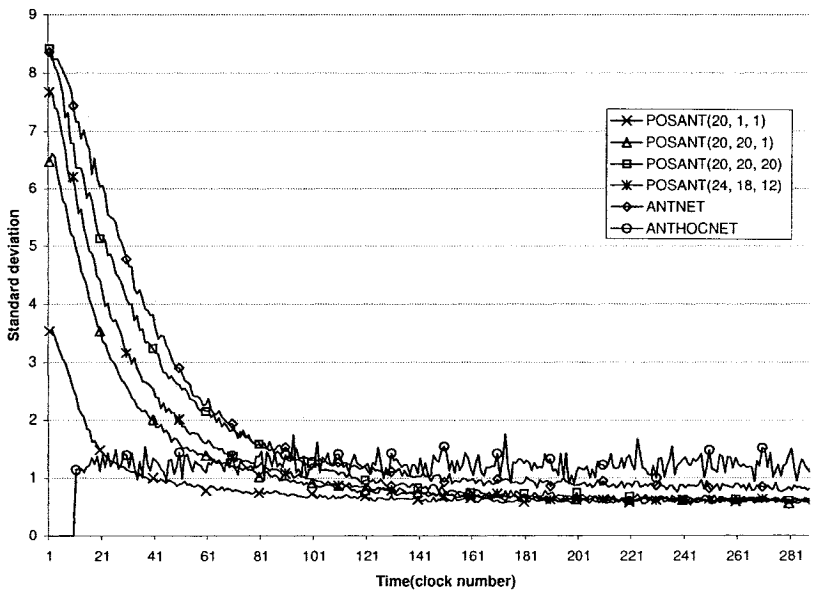

Figure 20: The standard deviation of the packet delays in ANTNET, POSANT, ANTHOCNET and GPSR.

Chapter 6

HYBNET routing algorithm

Our second routing algorithm, HYBNET [18], is able to adapt itself to non-uniform and dynamically changing network topologies. The advantage of HYBNET is that its efficiency remains relatively high for such network topologies and in comparison to other routing algorithms, the unevenness of topology has less effect on its efficiency. HYBNET is a hybrid algorithm which combines the idea of different variations of POSANT (i.e. each variation has different parameter settings) and greedy position based routing algorithms to establish routes between a source and a destination. Like POSANT, HYBNET has three main phases, route establishment phase, data transmission phase and failed/new link handling phase. We suppose there is a number of data packets in the source node S waiting for a route to be established between S and destination D. First, only forward and backward ants are transmitted and after the algorithm decides that the routes are established, data packets are transmitted.

In HYBNET, zones have the same definition as in POSANT algorithm. Consider

a destination node D and a network graph G. For each node N in G, we partition its neighbors into 3 zones called $zone_1, zone_2$ and $zone_3$. If H is one of the neighbors of N, θ_H is defined as the angle between line segments NH and ND. Node H belongs to $zone_1$ if $\theta_H \leq \pi/4$, $zone_2$ if $\pi/4 < \theta_H < 3\pi/4$, and $zone_3$ if $3\pi/4 \leq \theta_H \leq \pi$.

6.1 Pheromone initialization

In HYBNET, two pheromone trails may be assigned to each of the outgoing links of a node of the network for a specific destination. For each node we assign a *greedy trail* and a *regular trail* to each of its outgoing links if it is reached by at least one forward ant heading to that destination. Suppose A is a node in the network that receives an ant for the first time heading to destination D, and B is one of the neighbors of A. To initialize the value of the greedy trail on $\vec{AB}$, we use a greedy policy similar to the one used in POSANT, the assigned values decrease with the zone number. Having three values ν_{gr_1}, ν_{gr_2} and ν_{gr_3} such that $\nu_{gr_1} \geq \nu_{gr_2} \geq \nu_{gr_3}$, an amount of pheromone equal to ν_{gr_i} will be assigned to the greedy trail of $\vec{AB}$ if B belongs to $zone_i$. To the regular trail, also dependent on the zone, one of the three values $\nu_{reg_1}, \nu_{reg_2}, \nu_{reg_3}$ is assigned. In our experiments we assigned equal values to all ν_{reg}s.

Each node maintains a table which contains the values of the pheromone trails assigned to its outgoing links for different destinations. Whenever a forward ant for a specific destination arrives at a node, this table will be searched for pheromone trails assigned to that destination. If no pheromone trail exists, the pheromone initialization

process assigns pheromone trails for that destination to all the outgoing links of that node. If a node doesn't receive any packet concerning a destination for a specific period of time, the corresponding pheromone trails of that destination will be deleted from the pheromone trail table of that node. This time period is defined to be in the order of seconds. Also the entries in the BR table (i.e. to be defined later in this section) for that destination will be deleted after that time.

6.2 Route establishment

For establishing routes between a given source S and destination D, HYBNET launches n forward ants from S heading to D at regular time intervals. In our experiments we set n to be 1. Assigning big values to n may slightly decrease the route establishment time but it also increases the overhead of the algorithm.

In addition to the pheromone trial table, each node maintains another table which we call *Back Routing (BR)* table. When a forward ant enters a node from one of its neighbors, an entry in the BR table will be created that stores the identifier of the neighbor the forward ant is coming from, the sequence number of the ant, and the identifier of the destination. Using information stored in this table repeated forward ants will be destroyed. At each node a forward ant makes a stochastic decision in two steps to select the next hop. In the first step it decides which of the greedy or regular pheromone trails to use for making a stochastic decision in the second step. In the second step it selects the next hop stochastically using the values of the

selected pheromone trails (i.e. see Algorithm 4). The destination node upon receiving a forward ant destroys it and sends a backward ant back to the source node. Using information in the BR tables, the backward ant, which includes the sequence number of the corresponding forward ant, takes the same route but in reverse to reach the source node.

The decision of a forward ant to use either greedy pheromone trails or regular trails is marked in the BR table. When a backward ant reaches a node, the BR table will be searched for the corresponding forward ant. Based on the type of the pheromone trail (i.e. greedy or regular) used by that forward ant, we call this backward ant greedy or regular backward ant (notice that a regular backward ant in a node may become a greedy backward ant in another node). At each node, a backward ant reports the length of the traveled path from the destination to that node (i.e. number of hops). Each node keeps the average and standard deviation of the reported distances to the destination for the greedy and regular backward ants for a given destination. For simplicity, we call them greedy average, greedy standard deviation, regular average and regular standard deviation. After a node receives a greedy or regular backward ant, greedy or regular standard deviations and averages will be updated using the reported distance to the destination.

To reduce the effect of old backward ants, we define two fixed size *windows* in each node that contain recently received backward ants. One window is assigned to greedy backward ants and the other one is assigned to regular backward ants. The average and standard deviation of reported distances to the destination will be

Algorithm 3 HYBNET routing algorithm

```
{S is the source node, D is the destination node and C is the current node}
if C = S then
  put one compass packet in C's buffer
end if
for each clock time do
  if C = S then
    if one of the conditions in Section 6.3 is true then
      put one data packet in C's buffer
    else
      put one forward ant in C's buffer
    end if
  end if
  for each message m in C's buffer do
    if (m→type = ForwardAnt) or (m→type = DataPacket) then
      NextHop = SelectNextHop(n)
      send m to NextHop
      if NextHop = D then
        m →type = BackwardAnt
      end if
    else if m→type = BackwardAnt then
      find NextHop in C's BackRouting table
      send m to NextHop
      IncreasePheromone(NextHop,m)
      update regular and greedy averages
      if NextHop = S then
        drop m
      end if
    else if m→type = Compass then
      find NextHop using compass routing algorithm
      send m to NextHop
      if NextHop = D then
        m →type = BackwardCompass
      end if
    else if m→type = BackwardCompass then
      find NextHop in C's BackRouting table
      send m to NextHop
      if NextHop = S then
        drop m and enter greedy mode
      end if
    end if
  end for
  Evaporate()
end for
```

calculated only for the backward ants residing in the window. When a new backward ant is received we put it in the corresponding window and discard the oldest ant if the maximum widow size has been reached. Selecting an appropriate maximum window size is important. If the maximum window size is too small, the average delay calculated from the window information would be too far from the real average. If the maximum window size is very big, existence of very old ants would affect the result for a long time.

Suppose α_{gr}, σ_{gr} and α_{reg}, σ_{reg} are the average and standard deviation of the delays reported by greedy and regular backward ants residing in the corresponding windows. These averages are used by the future forward ants to decide on the use of greedy or regular trails for selecting the next hop. Suppose P_{gr} is the probability that greedy trails are used and P_{reg} denotes the probability that regular trails are used by a forward ant. P_{gr} and P_{reg} are calculated using Equation 6.

$$P_{gr} = \begin{cases} 1 & \text{if } \sigma_{gr} < t \text{ and } \alpha_{gr} < \alpha_{reg} \\ 0 & \text{if } \sigma_{reg} < t \text{ and } \alpha_{reg} < \alpha_{gr} \\ C_{gr} & \text{otherwise} \end{cases} \tag{6}$$

$$P_{reg} = 1 - P_{gr}$$

In the above equation t is a threshold value and when the standard deviation of the reported distances to the destination by a group of ants is less than this value, we conclude they have taken paths with almost the same lengths. C_{gr} is a constant value that determines which type of pheromone trails has higher chance to be used at the beginning of the route establishment.

In the second step, the ant should be forwarded to one of the neighbors using the values of the pheromone trails of the selected type. This step is done using a stochastic decision similar to other ACO routing algorithms. Suppose the algorithm decides to use greedy trails to select the next hop. Consider a forward ant is currently residing in node N with k neighbors $H_1, H_2, ..., H_k$. Suppose ϕ_{gr_i} is the value of greedy pheromone trail assigned to $\vec{NH_i}$. The ant will select H_i as the next hop with a probability p_i which is calculated using the following equation.

$$p_i = \frac{\phi_{gr_i}}{\sum_{j=1}^{k} \phi_{gr_j}} \tag{7}$$

A similar equation is used if the algorithm decides to use the regular trails. As mentioned, when a forward ant reaches the destination, it is destroyed and a backward ant is sent back to the source. Moving from node B to node A, a backward ant increases the amount of the corresponding pheromone trails stored in $\vec{AB}$. It uses a formula similar to the one used in POSANT to update the greedy pheromone trail on $\vec{AB}$.

$$\phi_{gr_{\vec{AB}}} = \phi_{gr_{\vec{AB}}} + \omega(\vec{AB}) \cdot g(d) \tag{8}$$

In the above formula d is the number of traveled nodes from the destination to node B by the backward ant and $g(d)$ is a decreasing function of d. ω is a *weight function* and its value is dependent on the zone of A which B is residing in, as given

in Formula 9.

$$\omega(\vec{AB}) = \begin{cases} w_1 \geq 1 & \text{if B is in } zone_1 \text{ of } A \\ w_2 = 1 & \text{if B is in } zone_2 \text{ of } A \\ w_3 \leq 1 & \text{if B is in } zone_3 \text{ of } A \end{cases} \tag{9}$$

Decreasing the convergence time in cases where the shortest path passes through nodes which are closer in direction to the direction of destination (i.e. most of the times this is the case), motivates the use of this weight function.

For updating regular trails, we use equation 10. In this equation $g(d)$ has the same value as $g(d)$ in Equation 8.

$$\phi_{reg_{\vec{AB}}} = \phi_{reg_{\vec{AB}}} + g(d) \tag{10}$$

An evaporation process modifies the value of pheromone trails in regular time intervals. It is done by multiplying the value of each pheromone trail by a number $\mu < 1$ at regular intervals.

$$\phi_{\vec{AB}} = \mu \cdot \phi_{\vec{AB}} \tag{11}$$

Algorithm 4 SelectNextHop

Input: node N

calculate P_{gr} {using Equation 6}

r = rand(0,1) {r is a random number between 0 and 1}

if $p_{gr} \geq r$ **then**

{greedy trails are selected}

for i = 1 to the number of N's neighbors **do**

$p_i = \frac{\phi_{gr_i}}{\sum_{j=1}^{k} \phi_{gr_j}}$

return neighbor i with probability p_i

end for

else

{regular trails are selected}

for i = 1 to the number of N's neighbors **do**

$p_i = \frac{\phi_{reg_i}}{\sum_{j=1}^{k} \phi_{reg_j}}$

return neighbor i with probability p_i

end for

end if

As mentioned before, this is to reduce the effect of ants taking non-optimum routes to reach the destination, especially at the beginning of the route establishment process.

At the beginning of route establishment process, a special control packet which we call *compass packet* is launched to the destination. This packet uses *compass* routing

algorithm to reach the destination. If the destination node receives this packet, it destroys it and sends a *backward compass* packet back to the source node. This packet takes the same path as the compass packet but in reverse to reach the source node. When the source node receives this packet, enters a greedy mode. In this mode when launching a forward ant, the sender marks it by setting a flag. The first time a node like A receives a marked forward packet for a special destination, uses Equation 12 to update the greedy pheromone trails on the outgoing links and also increases the value of C_{gr} in equation 6.

$$\phi_{gr\,\vec{AB}} = u(\vec{AB}) \cdot \phi_{gr\,\vec{AB}} \tag{12}$$

In the above formula B is one of the neighbors of A and $u(AB)$ is a weight function which value is depended on the zone of A regarding the destination in which B is residing.

$$u(\vec{AB}) = \begin{cases} u_1 > 1 & \text{if B is in } zone_1 \text{ of } A \\ u_2, u_1 \geq u_2 \geq 1 & \text{if B is in } zone_2 \text{ of } A \\ u_3 = 1 & \text{if B is in } zone_3 \text{ of } A \end{cases} \tag{13}$$

If the marked forward ant is the first forward ant for a specific destination which is arriving on A, the above update takes place after initializing the pheromone trails in the routing table of A.

The motivation of using a compass packet is that compass routing finds a route whose length is close to the length of the optimum route in most cases when the average degree of nodes is high. Moreover, the overhead of sending a single control

packet in the network is negligible while using this packet may significantly reduce the convergence time.

6.3 Sending data packets

After routes are established between the source and the destination, sending data packets should start. As we mentioned before, each node which is involved in the route establishment process keeps greedy and regular averages and standard deviations of the reported distances to the destination. To decide which time is the best time to start sending data packets, HYBNET uses the standard deviations and averages kept in the source node for that destination. Sending data packets too late increases delay and sending them too early increases packet loss and the delay caused by following bad routes. So picking a precise time to start sending data packets is important.

If one of the following conditions become true, HYBNET stops launching control packets from the source node and starts sending data packets instead.

- If $\sigma_{gr_{src}} < t$ *and* $\alpha_{gr_{src}} < \alpha_{reg_{src}}$
- If $\sigma_{reg_{src}} < t$ *and* $\alpha_{reg_{src}} < \alpha_{gr_{src}}$
- If $\sigma_{gr_{src}} < t$ *and* $\sigma_{reg_{src}} < t$

In the above equation, t is a threshold value and has the same value as in Equation 6. $\alpha_{gr_{src}}$, $\alpha_{reg_{src}}$, $\sigma_{gr_{src}}$ and $\sigma_{reg_{src}}$ in the above equation stand for the average

and standard deviation of the reported distances to the destination for greedy and regular trails which are kept in the source node.

If $\sigma_{gr_{src}}$ is small enough, we can assume that the forward ants which decide to use greedy trails to select one of the neighbors of the source node as the next hop are following routes with almost the same length to the destination. The same applies when $\sigma_{reg_{src}} < t$. So when $\sigma_{gr_{src}} < t$ and $\sigma_{reg_{src}} < t$ we can say that the algorithm has converged to a route or a group of routes with similar lengths. Thus it is a reasonable time to start sending data packets.

If the first condition is true, it means that the ants which used regular trails to select one of the neighbors of the source node as the next hop experience more delay than the ants used greedy trails. Also since $\sigma_{gr_{src}}$ is relatively small, the probability that this condition changes is low. Regarding equation 6 for calculating the values of P_{gr} and P_{reg}, in this case the packets will use greedy trails to select the next hop and so will use the routes that the algorithm is converged to. The same could be used to justify the second condition.In HYBNET, data packets are treated like forward control packets.

Since ACO routing algorithms eventually establish a group of routes with almost the same lengths to the destination [8], [11] and [29], at least one of the above conditions eventually becomes true and sending data packets will start. Our experiments given in the next section show that sending data packets starts quite soon in practice.

6.4 Failure recovery

If due to the mobility of nodes or any other reason a link between two nodes say A and B breaks while a connection is running between a source S and a destination D, HYBNET performs a failure recovery as follows (the idea used here is similar to the idea used in POSANT). For each destination, each node defines a mode that can have two values, *proper mode* and *broken mode*. Initially each node is in the proper mode. If A realizes that the link to B is broken and there are pheromone trials corresponding to link $\vec{AB}$ for D in the pheromone table of A, its mode will be changed to broken mode. In the broken mode, if A has another outgoing link to which pheromone trails are assigned for destination D, the stochastic data routing will be continued. Otherwise, A sends a message to its neighbors to inform them that there is no route from A to the destination any more. Upon receiving this message, each node enters broken mode and follows the same algorithm. If a packet arrives for the second time to A while it is in the broken mode, a message will be sent to its neighbors to tell them there is no route to D from A. It is to prevent loops after a link failure. When a node receives a backward ant which should pass through a broken link, the backward ant will be dropped. If the source node has only one outgoing link that contains pheromone trails for D and this link breaks or a message from this link is received that states there is no route to D, a new route establishment process will begin and sending data packets will be suspended until new routes are established. After a specific time which is defined to be in the order of milliseconds is passed from

the moment that the link failure is detected, the mode of A will be changed to the proper mode.

6.5 Handling new links

When a new node joins the network, or when a node is moving and approaching some other nodes, new links may appear in the network. Suppose node A figures out that it can directly communicate with a new node C. A has routing information about some destinations like D in its pheromone table. HYBNET assigns a pheromone trail to link $\vec{AC}$ as follows. A calculates the average of the value of the pheromone trails for destination D in its pheromone table. If C is located in $zone_1$ of A (i.e. for destination D), then a pheromone trail whose value is equal to this average will be assigned to $\vec{AC}$. Otherwise if C is located in the other two zones, the half of this average will be assigned to this link. The same approach is applied in node C for assigning pheromone trails to $\vec{CA}$ (i.e. for the destinations that node C has routing information about them). In the case that node C does not have routing information for D (i.e. because node C is a new node or it is not reached in the routing process before), after it receives a packet heading to D, a pheromone initialization process (as explained before) assigns pheromone trails to its outgoing links.

Using the mentioned strategies for failure recovery and new link handling, HYBNET performs well in cases where nodes move quite fast in the network (refer to Section 6.6).

6.6 Performance evaluation

In this section we study the effect of different parameters settings on the performance of HYBNET. Also we compare the behavior of HYBNET, POSANT, ANTNET, ANTHOCNET and GPSR on different networks and study how these algorithms behave when the topology of the network changes. The comparison is made based on the simulation results of these algorithms. The simulation program was written in VC++ environment.

Randomly generated graphs are used to evaluate the mentioned routing algorithms. Each node in the generated graphs has a different transmission range in different directions and two nodes are connected if they are in the transmission ranges of each other. Nodes are placed randomly in the area following a Poisson distribution. All the generated graphs are connected. For different evaluations different graph models are used and more detailed specification of the network graphs is described in each section.

In the remainder of this section, we first study the effect of changing the values of the parameters of HYBNET on its performance. We compare the performance of different routing algorithms including HYBNET, POSANT, ANTNET, ANTHOCNET and GPSR on a set of network graphs with similar topology characteristics. The main parameters which characterize a network topology are the number of nodes, the average degree of nodes and the shape of the area in which nodes are distributed. We then study the behavior of the mentioned algorithms when one of the above parameters

Table 3: Parameters defined in HYBNET and their values.

Parameter	Value	Description
n	1	number of forward ants generated by the source node at each clock time
t	1	standard deviation threshold
$(\nu_{gr_1}, \nu_{gr_2}, \nu_{gr_3})$	(20,1,1)	used in pheromone initialization
$(\nu_{reg_1}, \nu_{reg_2}, \nu_{reg_3})$	(10,10,10)	used in pheromone initialization
μ	0.95	used in pheromone evaporation
$(\omega_1, \omega_2, \omega_3)$	(1.2,1,0.8)	used in Equation 9
(u_1, u_2, u_3)	(10,1,1)	used in Equation 13
c_{gr}	0.8	used in Equation 6
window size	50	defined in route establishment phase
$g(d)$	$\frac{1000}{d^2}$	used in Equation 8 and 10

changes.

In this section, it is supposed that it takes one millisecond for each packet to go from one node to a neighbor node and this time is always the same all over the network.

Table 3 lists the parameters of HYBNET and their corresponding values used in our experiments. We found that HYBNET performs better with this setting after trying different values assigned to the above parameters. The parameters of POSANT and ANTHOCNET and their corresponding values are listed in Table 4 and 5.

6.6.1 Selection of the parameters

In this part, the influence of different parameters of HYBNET is investigated by trying this algorithm with different settings on a set of network graphs. Among the parameters listed in Table 3, the effect of n, ν_{gr}s, ν_{reg}s and c_gr on the performance of

Table 4: Parameters defined in POSANT and their values.

Parameter	Value	Description
t	1	refer to [19]
(ν_1, ν_2, ν_3)	(20,1,1)	used in pheromone initialization
μ	0.95	used in pheromone evaporation
window size	50	refer to [19]
$g(d)$	$\frac{1000}{d^2}$	refer to [19]

Table 5: Parameters defined in ANTHOCNET and their values.

Parameter	Value	Description
μ	0.95	used in pheromone evaporation
α_1	2	
a	20	used in pheromone evaporation

HYBNET is studied. The set of networks that is used in this part includes 100 graphs. Each graph has 80 nodes distributed uniformly over an area of size 500×500. The transmission range of nodes varies from 50 to 70 units. For each network 10 source-destination pairs are selected randomly and the result is averaged. In the following comparisons, except the parameters whose effect is studied, the other parameters have the values listed in Table 3. The performance is evaluated in terms of *delivery rate* which is the ratio of the number of packets received by the destination node to the number of packets sent by the source node, and *average packet delay*. Average packet delay is the average time it takes for sent packets to reach the destination. When all the packets are taking routes with almost the same length, the delay they experience will be almost the same. Also when all packets are experiencing the same delay, we

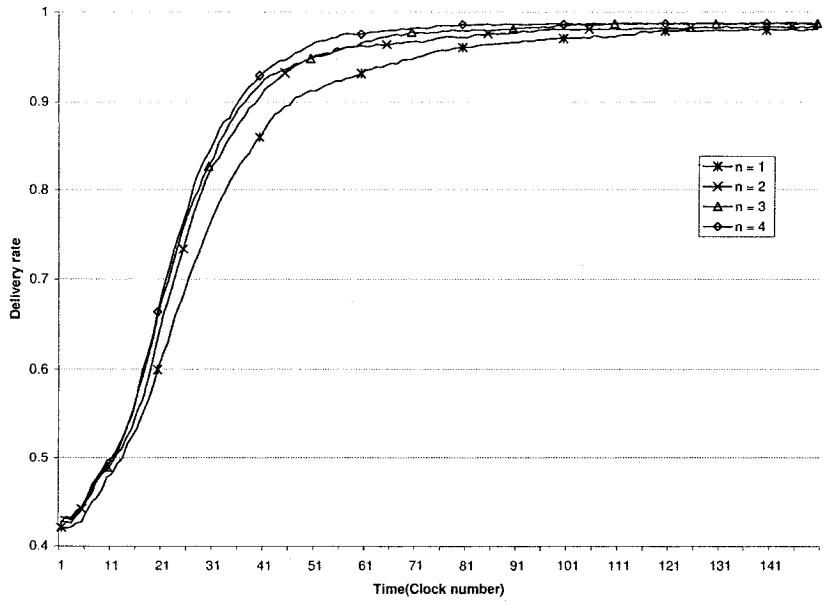

Figure 21: Average delivery rate of HYBNET with different values assigned to parameter n.

can conclude that the algorithm is converged to routes with almost the same length. So the standard deviation of delays can be used to compare the average time it takes for the algorithm to converge to some routes with almost the same length.

In Figure 21, 22, 23 the effect of n, the number of generated forward ants at each clock time by the source node, on the performance of HYBNET is studied. As these figures show, the performance becomes better when two forward ants are launched at each clock time, but it doesn't change significantly when n is three. Figure 24, 25, 26 show the effect of C_{gr} on the performance of HYBNET. The performance of HYBNET with $C_{gr} = 80\%$ is the best. In Figure 27, 28, 29 the effect of ν_{gr}s on the performance of HYBNET is studied. In these figures, (a,b,c) denotes the values of ν_{gr_1}, ν_{gr_2} and ν_{gr_3} in order. The best result is gained when we assign high values to ν_{gr_1} and small values to ν_{gr_2} and ν_{gr_3}.

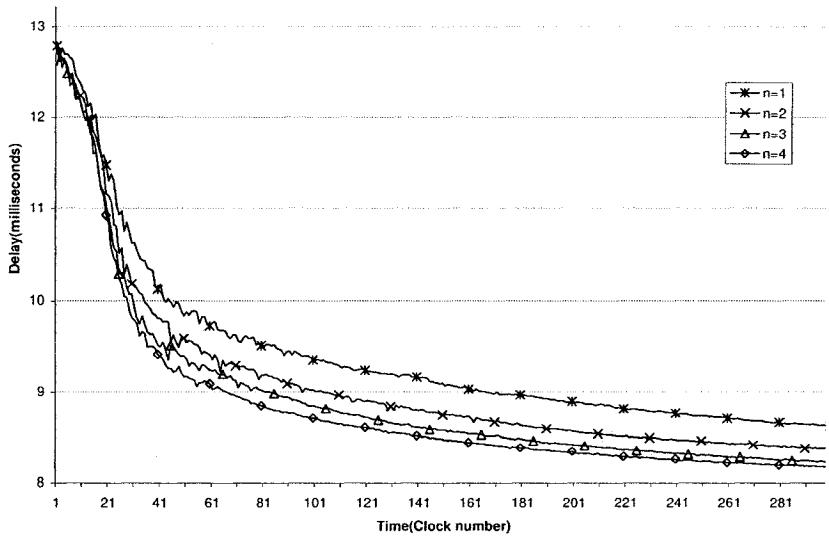

Figure 22: Average packet delay in HYBNET with different values assigned to parameter n.

6.6.2 Mobility of nodes

In this part the performance of HYBNET when the nodes are moving is investigated. We used a variation of Manhattan mobility model [1, 30] in which nodes are moving at a constant speed. In this mobility model, each node takes a step in the same direction as its previous step with probability 0.50. Also it can take an orthogonal step (i.e. turn left or right) with probability 0.25 (i.e. for each direction). HYBNET is tried on a set of 100 networks. Each network graph has 60 nodes spread over an area of size 400×400. The transmission range of nodes is between 40 and 60. The performance is evaluated for different speeds from 0 to 30 units per clock time.

The results of this simulation is presented in Figure 30, 31, 32. In higher speeds, it takes longer time for the algorithm to converge but after it converges, even though nodes are moving, the performance is not affected significantly.

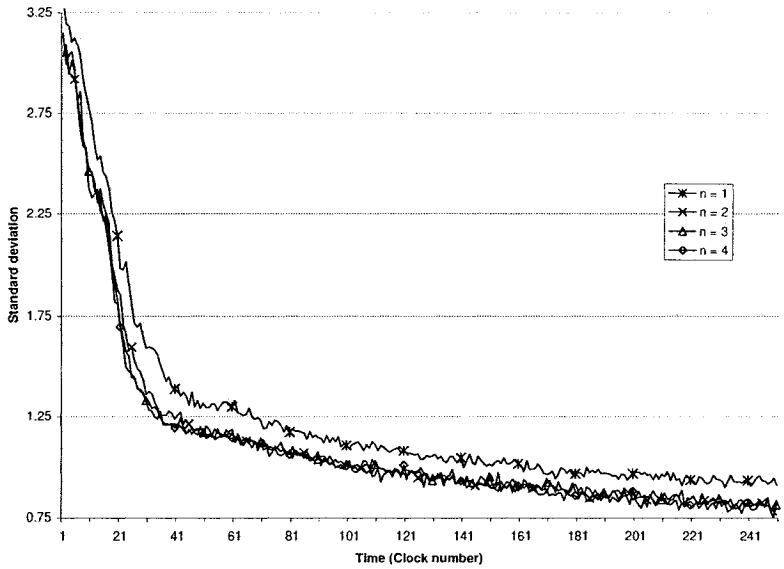

Figure 23: Standard deviation of packet delays in HYBNET with different values assigned to parameter n.

6.6.3 Networks with similar topology characteristics

We compare here the performance of different routing algorithm by trying them on a set of network graphs with similar topology characteristics. In these network graphs, the number of nodes, average degree of nodes and the shape of the area within which the nodes are distributed are the same. The nodes are distributed over an area of size 600×600 square units. The transmission range of nodes varies from 40 to 60 units. Each network has 90 nodes and there is a limit of 6 on the degree of nodes so each node has at most 6 neighbors. The algorithms are tried on 200 networks and 10 search-destination pairs are selected randomly in each network. The results are averaged and presented below.

If a routing algorithm fails to deliver a packet to the destination, the sender must somehow detect this failure and try to resend the packet hoping that this time the

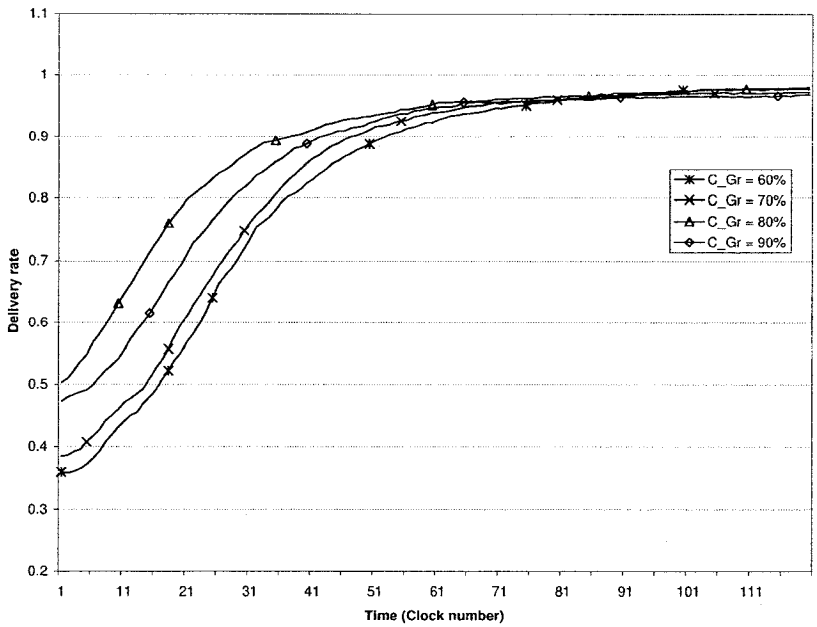

Figure 24: Average delivery rate of HYBNET with different values assigned to parameter C_{gr}.

algorithm delivers it. This increases the traffic in the network and also the delay experienced by the receiver. Thus it is very important to guarantee high delivery rate. In Figure 33 the average delivery rate of HYBNET, ANTNET, ANTHOCNET, POSANT and GPSR is compared. This figure shows how the average delivery rate varies with time. As the time progresses, the delivery rate increases and eventually becomes almost 100% for all algorithms except GPSR. As the graph shows, ANTHOCNET reaches 100% delivery rate faster than the others. This is due to broadcasting of ants in this algorithm which results in a large overhead. GPSR always has a relatively low delivery rate. It is because this algorithm fails in some cases as a result of irregular transmission ranges of the nodes. HYBNET has the highest delivery rate among the others. ANTNET reaches to 100% delivery rate slower than

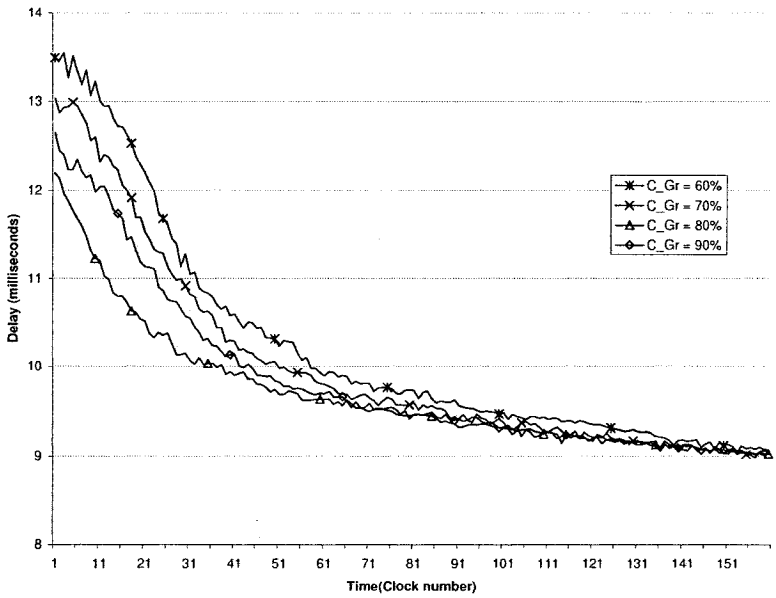

Figure 25: Average packet delay in HYBNET with different values assigned to parameter C_{gr}.

the other ant based algorithms.

The results of comparing average packet delays is presented in Figure 34. As mentioned before, hop count is used as the metric to measure packet delays. Figure 34 shows how the average packet delays for the different algorithms vary by time. In the beginning, the average packet delay of ANTHOCNET is smaller than the others. The average delay of HYBNET reaches to its minimum faster than the others. As the time passes on, all the algorithms converge to the paths with almost the same lengths and the packets experience almost the same delay. Because of the low delivery rate of GPSR in this network model, this algorithm is not considered for this comparison.

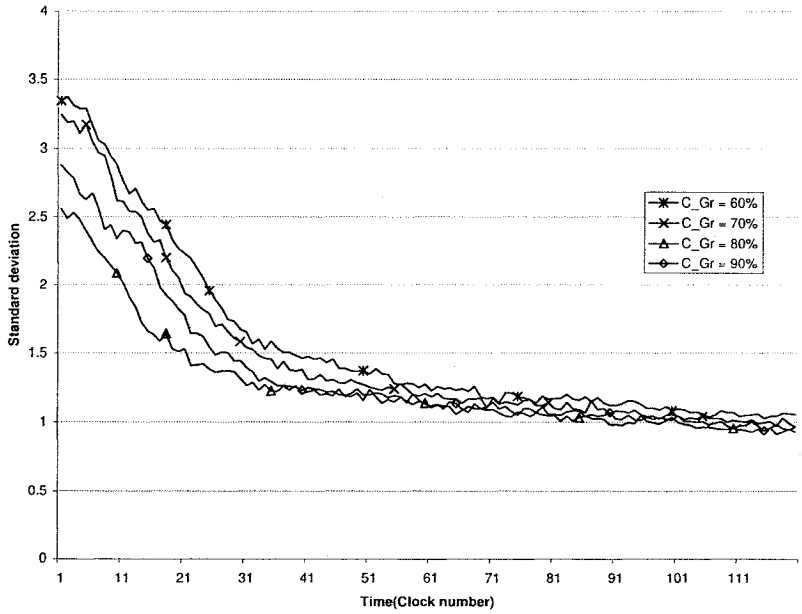

Figure 26: Standard deviation of packet delays in HYBNET with different values assigned to parameter C_{gr}.

6.6.4 Networks with different topology characteristics

In this section we study the behavior of different routing algorithms as a function of the parameters that characterize the network topology.

First we consider the size of the network (i.e. the number of nodes in the network graph). To make a comparison, randomly generated networks with the number of nodes starting from 10 and increasing up to 110 are used. For each size, 10 graphs are generated and 10 source-destination pairs for each graph are randomly selected. Each node has a transmission range between 40 and 70 in different directions and each node can have at most 6 neighbors. The average delivery rate of the routing algorithms in clock number 30 is compared in Figure 35. This figure shows how the average delivery rate changes in relation to the increase of the number of nodes. As

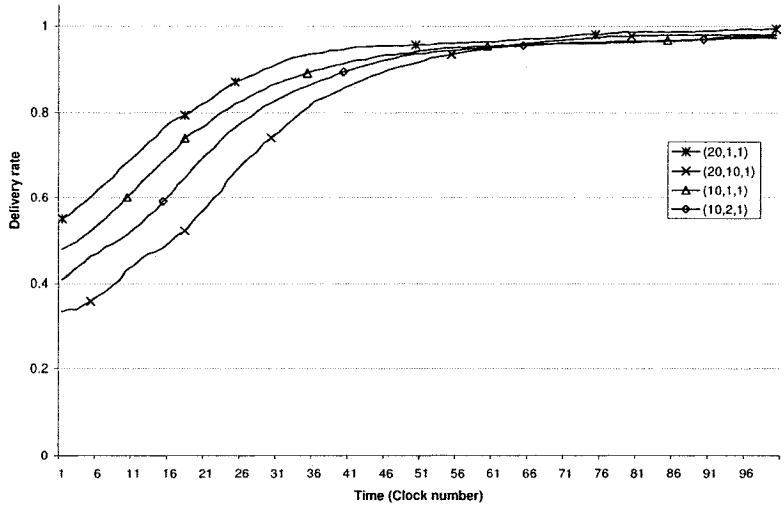

Figure 27: Average delivery rate of HYBNET with different values assigned to ν_{gr}s.

can be seen, when the size of the network is small, all the algorithms have almost 100% delivery rate after 30 clocks. But when the size increases, the delivery rate at that point of time decreases. It means that the ACO based routing algorithms need more time to converge. Also the figure implies that HYBNET and POSANT have higher delivery rate than the other algorithms in large networks when the average degree of nodes in the graph remains almost constant. Also the probability that GPSR fails to find a route increases. ANTNET converges slower in big networks.

Average degree of nodes in a network also has a great impact on the performance of different routing algorithms. The density of nodes, defined as the number of nodes per unit of the area, and also the transmission range of nodes determine this average. To compare the behavior of different routing algorithms when the average degree of nodes varies, we used randomly generated graphs with different average degree of nodes. The number of nodes in each graph is constant and equal to 120. Each

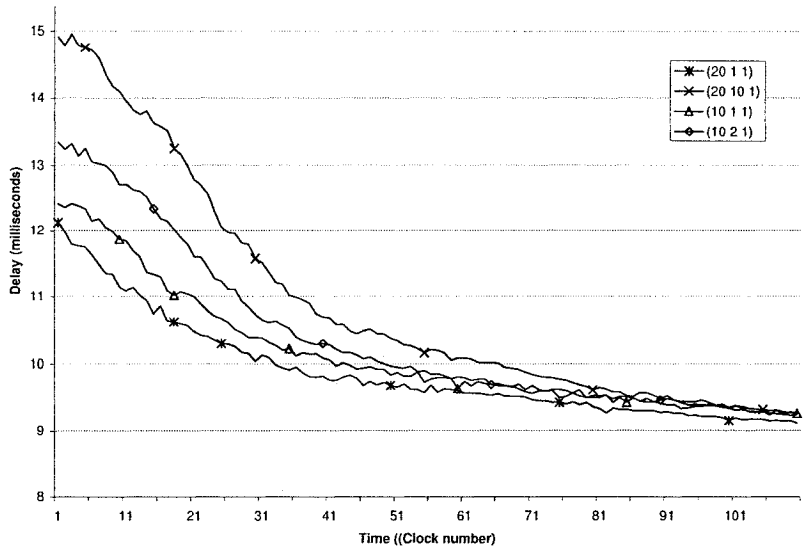

Figure 28: Average packet delay in HYBNET with different values assigned to ν_{gr}s.

node has a transmission range between 40 and 60. Source and destination nodes are located in constant positions and the distance between them is 425 units. Other nodes are randomly distributed such that a connected network with a specific density is generated. The delivery rate of the algorithms is measured after 35 clock times. Figure 36 shows the result of this comparison. When the average degree of nodes is small, GPSR has a very low delivery rate. POSANT also has a relatively lower delivery rate, because when the average degree if nodes is small, some cases may happen ina which POSANT converges slowly. HYBNET has a high delivery rate independent of the density of nodes. The delivery rate of ANTNET slightly decreases as the density increases.

The effect of very irregular area on the performance of different routing algorithms is studied in this part. To make a comparison, we generated 100 networks with nodes distributed over an area whose shape is shown in Figure 37. Each graph has 100 nodes.

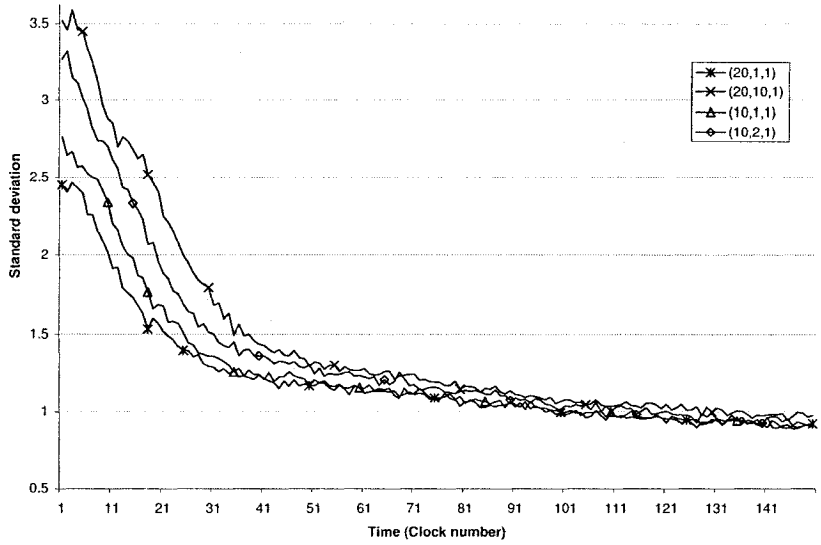

Figure 29: Standard deviation of packet delays in HYBNET with different values assigned to ν_{gr}s.

Each node is placed randomly within the area except source and destination nodes which are placed in fixed points shown in Figure 37. This is a challenging example for most routing algorithms. In Figure 38 the average delivery rate of the different algorithms is compared. As you can see GPSR has the lowest delivery rate. POSANT has a relatively poor delivery rate at the beginning. HYBNET has the highest delivery rate after ANTHOCNET. Figure 39 compares the average delay of the mentioned routing algorithms. Because GPSR has a very low delivery rate, it is not considered for this comparison. HYBNET has the shortest delay after ANTHOCNET. Also we can see that ANTNET has a shorter convergence time than POSANT. This comparison shows that for some network graphs with nodes spread over a very irregular area, the location of source and destination can make the convergence time of POSANT and ANTNET very long. Also in these cases position based routing algorithms may have

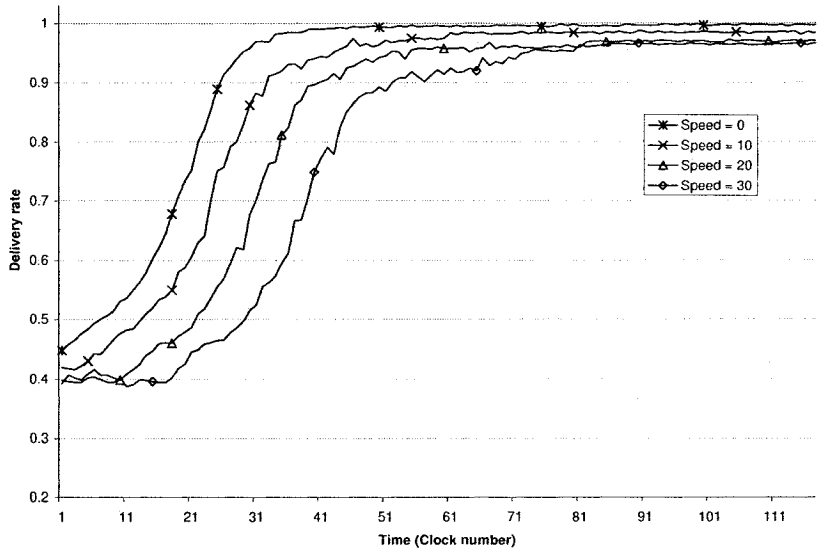

Figure 30: Average delivery rate of HYBNET when nodes are moving with different speeds.

a very low delivery rate. HYBNET has a relatively short convergence time in these cases.

6.6.5 Overhead of Algorithms

The overhead of a routing algorithm should be kept as small as possible. Parameters which are very important in evaluating the overhead of a routing algorithm is the size and amount of control messages produced by that routing algorithm. Producing many control packets may increase the traffic in the network, decrease the scalability of the routing algorithm and so on. In this subsection we compare the amount of generated traffic by each routing algorithm.

To make this comparison, we used a set of 100 network graphs with 80 nodes randomly and uniformly spread over a square of length 500 units. In Figure 40

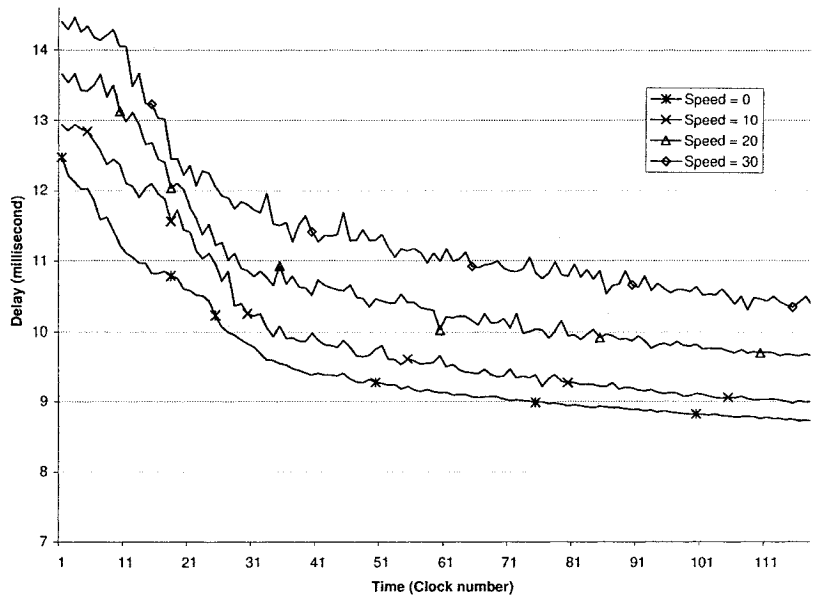

Figure 31: Average packet delay in HYBNET when nodes are moving with different speeds.

the average number of generated ants by ANTHOCNET in different clock times is shown. As the graph shows, a burst of ants is generated at the beginning of route establishment when the algorithm broadcasts an ant. The number of generated ants by HYBNET, POSANT and ANTNET is constant in each clock time. HYBNET generates n ants in each clock time (i.e. n is 1 in our experiments). ANTNET generates one ant in each clock time and POSANT generates at most 3 ants at each clock time (i.e. this value may be less than three because in some cases the source node may have no neighbor in some zones, resulting in fewer than three ants generated in each clock time). The total number of ants generated by HYBNET, POSANT, ANTHOCNET and ANTNET is compared in Table 6. The total number of ants generated by ANTHOCNET grows exponentially at the beginning while it is a linear function of time in HYBNET, POSANT and ANTNET. The huge number of

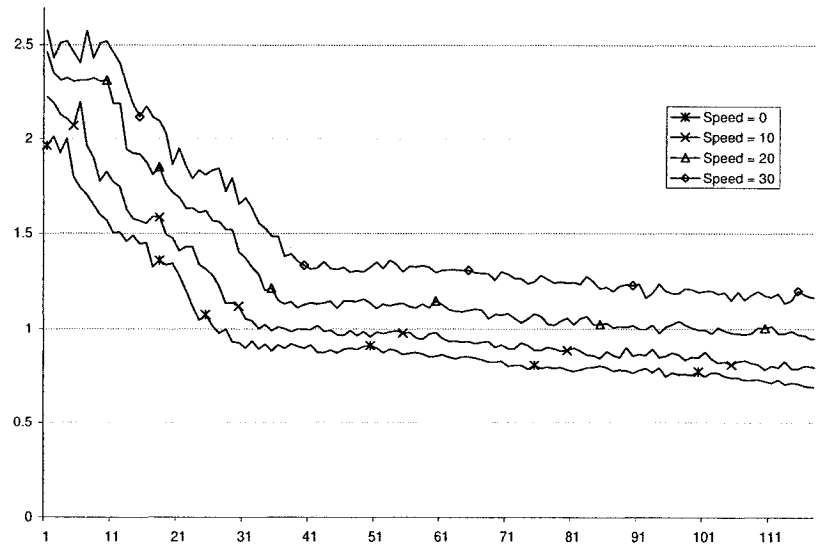

Figure 32: Standard deviation of packet delays in HYBNET hen nodes are moving with different speeds.

generated ants by ANTHOCNET when it establishes a new route makes it unsuitable as a reactive routing algorithm. Since the ants in ANTHOCNET contain a list of the visited nodes, the size of an ant can be relatively large which makes the overhead even worst. The small number of generated ants by HYBNET, POSANT and ANTNET doesn't affect the network's traffic.

Table 6: Total number of generated ants.

Clock time	0	5	10	300
HYBNET	1	5	10	300
POSANT	2.53	12.67	25.35	756.4
ANTNET	1	5	10	300
ANTHOCNET	1	10488	125048	125048

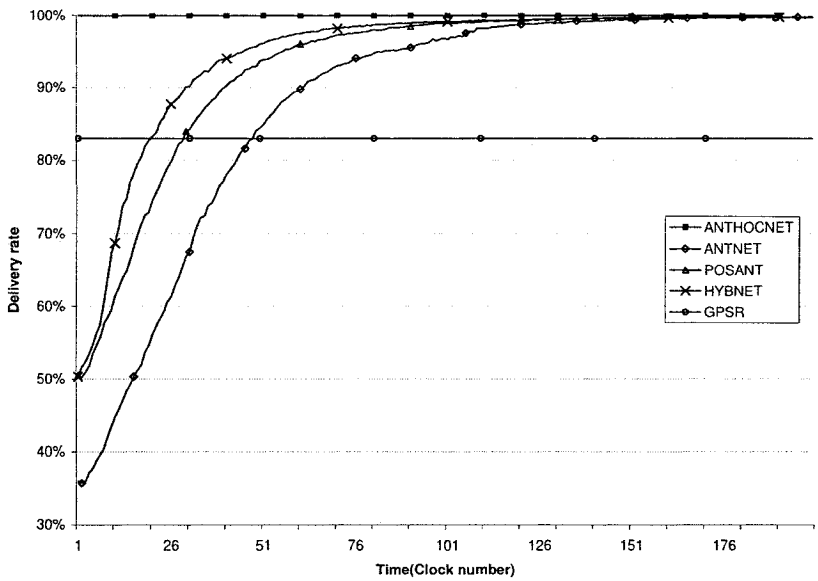

Figure 33: Average delivery rate of HYBNET, ANTNET, POSANT, ANTHOCNET and GPSR.

6.7 Discussion

There are some parameters that can characterize the topology of a mobile ad hoc network and here we discuss how HYBNET performs when these parameters change in a network. As stated before, these parameters include the number of nodes, the average degree of nodes and shape of the area in which the nodes of the network are spread. A mobile ad hoc network may contain less than ten nodes or may contain hundreds of nodes. A network with a certain number of nodes may spread over a wide area or it may be limited to a building. The existence of obstacles and noise may cause the network graph has an irregular shape. Also noise, obstacles and difference in the power of mobile nodes makes the transmission range of nodes irregular which increases the difficulty of the routing problem. In the next parts we discuss how each of these parameters may affect the performance of HYBNET and other routing

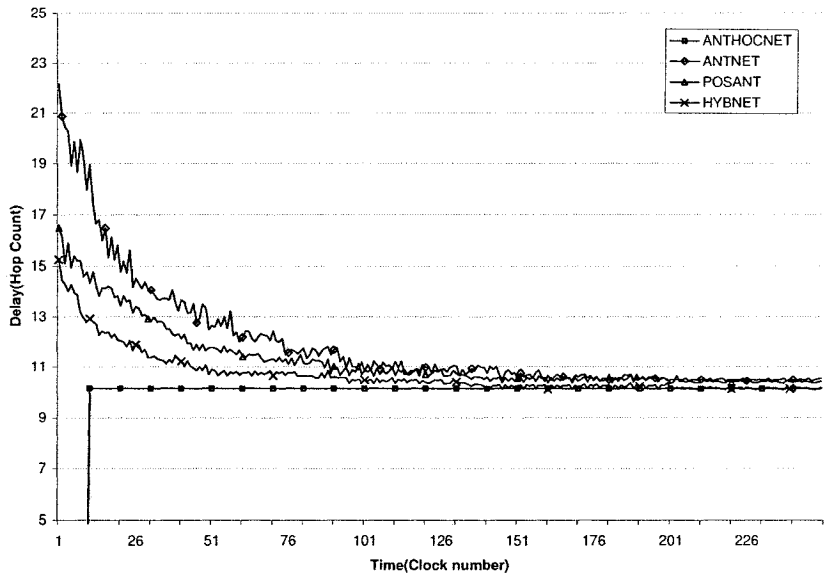

Figure 34: Average packet delay of HYBNET, ANTNET, POSANT and ANTHOCNET.

algorithms.

6.7.1 Network size

The size of a network is one of the most important characteristics of its topology. Routing in small networks is much simpler than in huge networks. Most of the routing algorithms work well in small networks. The challenging problem is routing in large networks. ANTHOCNET is a suitable algorithm for networks with small number of nodes but increasing the number of nodes exponentially increases the number of generated ants in this algorithm. ANTNET converges to optimum routes relatively fast in small networks but increasing the number of nodes in the network increases the convergence time of this algorithm. For greedy position based routing algorithms like compass routing, increasing the number of nodes in a certain area (i.e increasing

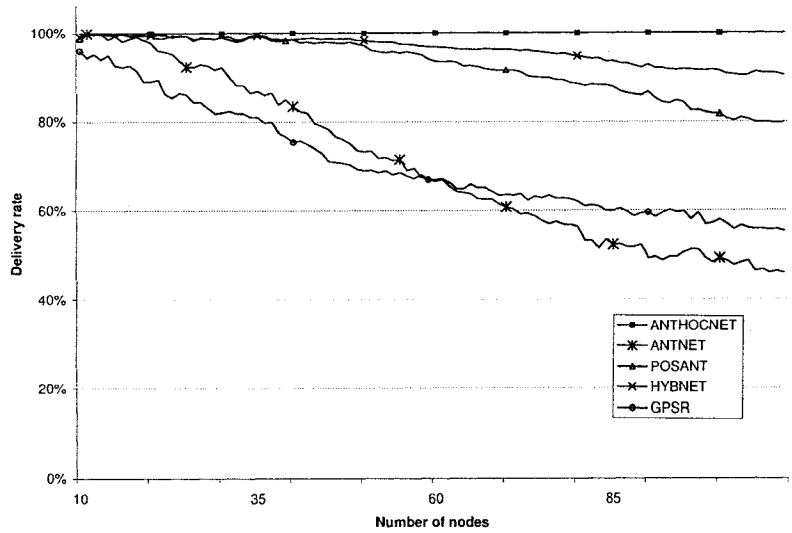

Figure 35: The change of the delivery rate when the size of the network grows.

the density) improves the efficiency of the algorithm. But if the network has more number of nodes while the average degree of nodes remains constant, the risk of failure is higher. Face routing algorithms find a route to the destination regardless of the size of the network unless the transmission ranges of the nodes are the same but the route that these algorithms find may be non-optimum if the network is huge and the average degree of nodes is low. The convergence time of POSANT in very large networks may be relatively long but it is much faster than ANTNET and other ACO routing algorithms except in some rare cases which will be explained later. HYBNET as a hybrid algorithm, which combines different strategies in establishing routes performs better than the other mentioned algorithms in large networks. Also using the compass packet in HYBNET causes the convergence time be very short in network models where compass routing can find an optimum route.

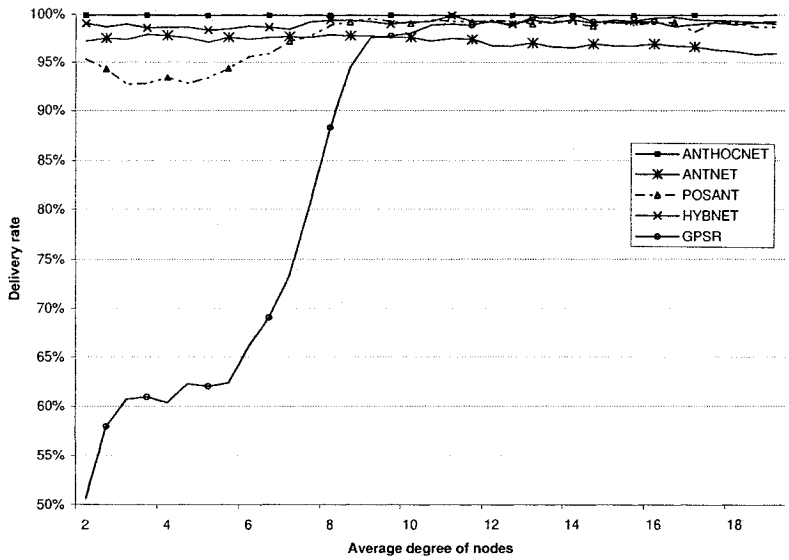

Figure 36: The change of the delivery rate when the average degree of nodes in the the network increases.

6.7.2 Average degree of nodes

Average degree of nodes affects the performance of most routing algorithms. Average number of nodes in each square unit of the area where the network is distributed and the average transmission range of each node are main factors in determining the average degree of nodes. Decreasing the average degree of nodes may slightly enhance the performance of ANTNET. It is because in a sparse graph a node has less number of neighbors so the ants have less choices for the next hop and so the convergence time may be faster. In contrast, most position based routing algorithms including greedy and face routing algorithms perform much better when the average degree of nodes is high. Greedy position based routing algorithms like compass routing have a very high delivery rate and find optimum routes when the average degree of nodes is high but their performance significantly degrades when the average degree

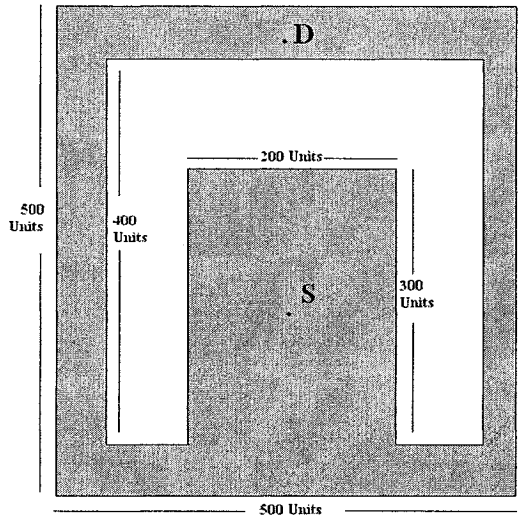

Figure 37: Nodes of the graphs are distributed in the shaded area. Source and destination nodes are shown.

of nodes decreases. Face routing algorithms may find non-optimum routes when the average degree of nodes is small. For example if the network graph is a tree, the route that these algorithms find may be much longer than the optimum route. Since POSANT mixes the idea of ACO with the information about the position of nodes, its performance is high in both dense and sparse routing algorithms. HYBNET has a better performance than POSANT in graphs where the average degree of nodes is high because it is using compass routing algorithm to decrease the convergence time. Also in the graphs where the average degree of nodes is low, the hybrid ACO route establishment strategy converges faster to optimum routes.

6.7.3 Shape of the network graph

In position based routing algorithms shape of the network graph is very important. Existence of holes or local minima in the network graph may result to a failure in

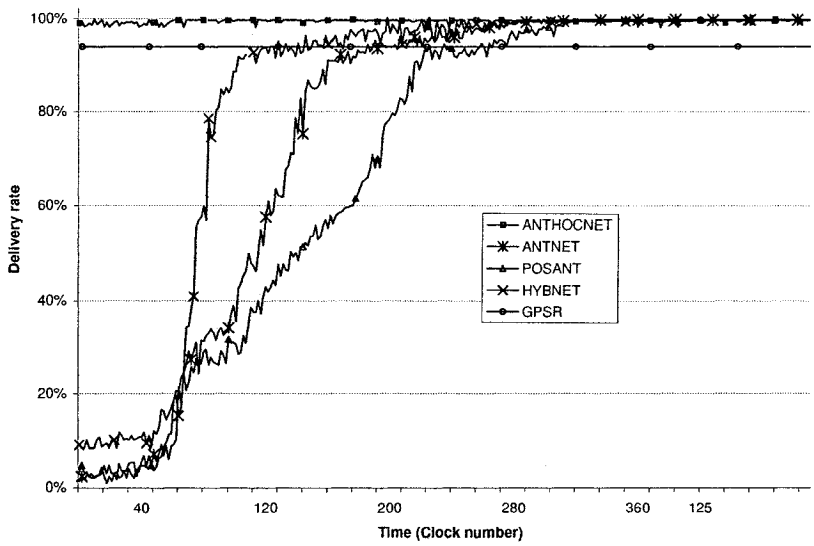

Figure 38: Average delivery rate of different routing algorithms in a set of graphs with a special shape when the source and destination nodes are fixed.

these routing algorithms. Also these irregularities in the shape may cause face routing algorithms to find routes which are much longer than optimum routes. The shape of the graph doesn't affect the performance of ANTNET and ANTHOCNET. POSANT as an ACO routing algorithm does not fail to find optimum routes in networks with irregularities in shape. Moreover in most cases even with irregular network shapes, POSANT converges faster than other ACO routing algorithms (i.e except the ones that use flooding). Using the information of the position of nodes in the network and updating pheromone trails regarding this information is similar to giving the destination a gravity power to absorb ants and it makes the ants to take better routes toward destination. However in some rare cases the convergence time of POSANT may be less than ANTNET. Figure 41 shows an example of a network graph in which POSANT converges slower to a destination than ANTNET. This is because

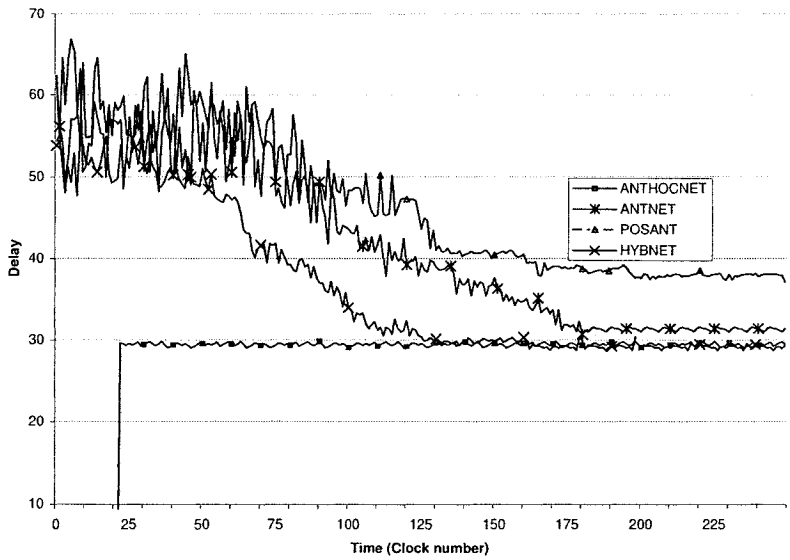

Figure 39: Average delay of different routing algorithms in a set of graphs with a special shape when the source and destination nodes are fixed.

POSANT promotes ants to select the nodes which are in the direction of destination as the next hop but in this case these nodes are actually far from the destination. HYBNET has a high performance in these irregular shapes because the hybrid ACO route establishment will not be affected by the existence of local minima.

6.7.4 Nodes transmission ranges

Network models that consider the transmission ranges of all nodes the same are not close to the reality. In real mobile ad hoc networks nodes may have different transmission ranges in different directions. The position based routing algorithms can not guarantee finding a route to the destination when the network contains nodes with irregular transmission ranges also the route that they find may be longer than optimum routes. On the other hand, ACO routing algorithms find optimum routes

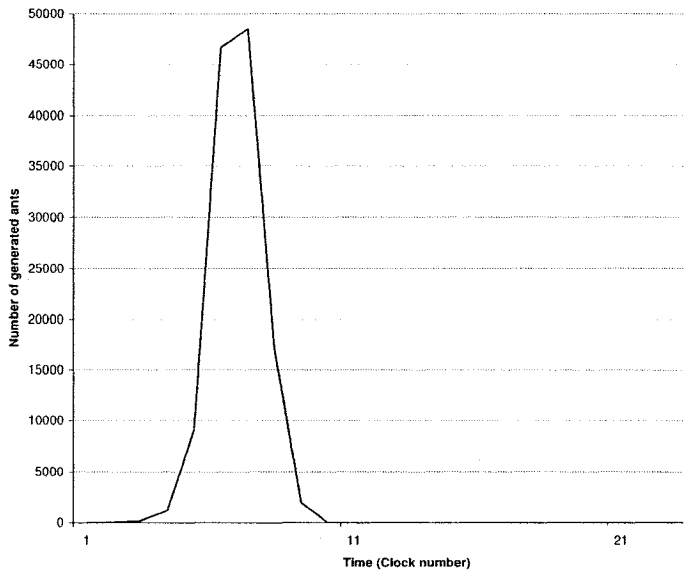

Figure 40: Number of generated ants in ANTHOCNET algorithm at each clock time.

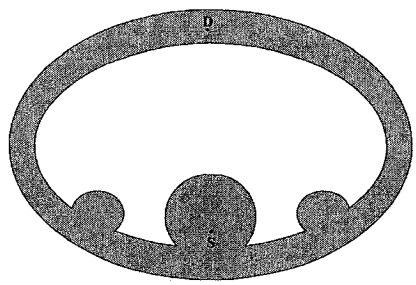

Figure 41: An example of a network in which POSANT is slower than ANTNET in establishing a route from S to D. The shaded area indicates the places that the nodes of the network are located.

in networks with irregular transmission ranges. So the performance of HYBNET is not affected by existence of nodes with irregular transmission ranges.

Chapter 7

Conclusions and future works

In this thesis we first proposed POSANT, a new ant colony based routing algorithm that uses the information about the position of the points to increase the efficiency of regular ant-routing algorithms. In contrast to other position based routing algorithms, POSANT does not fail when the network contains nodes with different transmission ranges. Unlike the regular position based routing algorithms which are single path, POSANT is a multipath routing algorithm. While in some cases regular position based routing algorithms find a route which is much longer than the shortest path, POSANT converges to routes which are close in length to the shortest path. The use of location information as a heuristic parameter resulted in a significant reduction of the time needed to establish routes from a source to a destination which is important for a reactive routing algorithm. In addition to having a short route establishment time, POSANT keeps the number of generated control messages very small while some ACO routing algorithms like ANTHOCNET use flooding to reduce this time

which makes them unscalable. Our simulations show that in our network model, POSANT has a higher delivery rate with a shorter average packet delay than GPSR as a position based routing algorithm. POSANT reaches a stable behavior faster than ANTNET. The number of control messages used in POSANT for route establishment is far less than ANTHOCNET. Thus, POSANT is a robust scalable reactive routing algorithm suitable for mobile ad hoc networks with irregular transmission ranges.

The topology of a mobile ad hoc network affects the performance of almost all the currently existing routing algorithms. The number of nodes in a network, average degree of nodes and shape of the network are the main factors that determine its topology. We proposed HYBNET, a routing algorithm which adapts itself with most network topologies. HYBNET uses a hybrid approach for establishing routes between a source and destination. This hybrid approach gives the algorithm a flexibility to perform well when some aspects of the network topology change. In most cases HYBNET converges relatively fast to optimum routes using a small number of control packets. This algorithm, like most ACO routing algorithms, does not fail to establish routes when the network includes nodes with irregular transmission ranges and establishes multiple routes between the source and destination. Our simulations show HYBNET has higher delivery rate and a shorter packet delay than ANTNET, POSANT and GPSR in our network models. When the number of nodes in the network is increased the performance of HYBNET is less affected than the other mentioned routing algorithms. Also unlike other position based routing algorithms, when the average degree of nodes in the network decreases, the delivery rate of HYBNET

remains high. HYBNET has an acceptable performance in networks with especial shapes where other routing algorithms perform relatively poor. ANTHOCNET has a high delivery rate and low packet delay regardless of the topology of a network but its overhead resulted by broadcasting ants makes it less useful as a reactive routing algorithm. Overall, regarding the time needed to establish routes in the network and the number of control packets used, HYBNET is a robust routing algorithm which perform well in most network topologies. It makes this routing algorithm suitable for mobile ad hoc networks with irregular topologies.

Nodes of most mobile ad hoc networks are distributed in three dimensional space. Currently our proposed routing algorithms are supposed to work on two dimensional graphs. Adapting our routing algorithms to three dimensional distribution of nodes is very important. To do so, we must modify the definition of zones in POSANT and HYBNET routing algorithms. Also for HYBNET, the compass packet should follow a modified version of compass routing algorithm. The performance of POSANT and HYBNET on three dimensional network graphs should be investigated.

In some applications, the nodes of a mobile network may have very high speeds and the network graph may change very fast. Based on our simulations, it seems that in very high speeds, it takes a long time for the mentioned algorithms to converge to some routes. The performance of POSANT and HYBNET should be investigated for different mobility models. If the performance is poor under a specific mobility model, different strategies for handling failed and new links should be considered.

Bibliography

[1] F. Bai and A. Helmy. A survey of mobility modeling and analysis in wireless adhoc networks. *Book Chapter, Wireless Ad Hoc and Sensor Networks*, 2004.

[2] L. Barriere, P. Fraigniaud, L. Narajanan, and J. Opatrny. Robust position based routing in wireless ad hoc networks with unstable transmission ranges. *Proc. of 5th ACM Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 19–27, 2001.

[3] L. Barrière, P. Fraigniaud, L. Narayanan, and J. Opatrny. Robust position-based routing in wireless ad-hoc networks with irregular transmission ranges. *Wireless Communications and Mobile Computing Journal*, pages 141–153, 2003.

[4] S. Basagni, I. Chlamtac, V. Syrotiuk, and B. Woodward. A distance routing effect algorithm for mobility (dream). *Proc. MOBICOM*, pages 76–84, 1998.

[5] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7:609–616, 2001.

[6] T. Camp, J. Boleng, and L. Wilcox. Location information services in mobile ad hoc networks. *Proceedings of IEEE Conference on Communications*, 5:3318–3324, 2002.

[7] S. Datta, I. Stojmenovic, , and J. Wu. Internal nodes and shortcut based routing with guaranteed delivery in wireless networks. *Cluster Computing*, 5:461–466, 2001.

[8] G. Di Caro and M. Dorigo. Ant colonies for adaptive routing in packet-switched communications networks. In *Proceedings of the 5th ACM International Conference on Parallel Problem Solving from Nature*, pages 673–682, 1998.

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

[10] G. Di Caro, F. Ducatelle, and L. M. Gambardella. Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *Special Issue on Self-Organisation in Mobile Networking*, 16:443–455, 2005.

[11] M. Dorigo and T. Stutzle. The ant colony optimization metaheuristic: Algorithms, applications, and advances. In *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research and Management Science*, pages 251–258. Kluwer, 2003.

[12] G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. *ISI Research Report*, pages 87–180, 1987.

[13] S. Giordano, I. Stojmenovic, and I. Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. *Ad Hoc Wireless Networking*, pages 103–136.

[14] M. Gunes, U. Sorges, and I. Bouazizi. Ara - the ant-colony based routing algorithm for manets. In *Proceedings of the 2002 International Conference on Parallel Processing Workshops*, pages 79–89, 2002.

[15] I. Haque, C. Assi, and J. Atwood. Randomized energy aware routing algorithms in mobile ad hoc networks. In *Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 71–78, 2005.

[16] T. Hou and V. Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34:38–44, 1986.

[17] P. Jeon and G. Kesidis. Pheromone-aided robust multipath and multipriority routing in wireless manets. In *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 106–113, 2005.

[18] S. Kamali and J. Opatrny. A topology adaptive routing algorithm for mobile ad-hoc networks. *The special issue of Elsevier Ad Hoc Networks Journal on*

Bio Inspired Computing and Communication in Wireless Ad Hoc and Sensor Networks.

[19] S. Kamali and J. Opatrny. POSANT: a position based ant colony routing algorithm for mobile ad-hoc networks. *ICWMC*, pages 21–26, 2007.

[20] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. ACM/IEEE MobiCom conference*, pages 243–254, 2000.

[21] Y. Ko and N. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6:307–321, 2000.

[22] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In *Proc. of 11th Canadian Conference on Computational Geometry*, pages 51–54, August 1999.

[23] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. *Proceedings of ACM/IEEE MOBICOM*, pages 120–130, 2000.

[24] R. Nelson and L. Kleinrock. The spatial capacity of a slotted aloha multihop packet radio network with capture. In *Proc. IEEE Transactions on Communications*, volume COM-32, pages 684–694, 1984.

[25] I. Stojmenovic. Voronoi diagram and convex hull based geocasting and routing in wireless networks. *SITE, University of Ottawa, TR-99-11*, 1999.

[26] I. Stojmenovic and X. Lin. Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.

[27] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions on Communications*, pages 246–257, 1984.

[28] H. Wedde, M. Farooq, and T. Pannenbaecker. BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 153–160, 2005.

[29] J.-H. Yoo, R. La, and A. Makowski. Convergence results for ant routing. In *Proceedings of CISS, University of Princeton, Princeton, NJ*, 2004.

[30] B. Zhou, K. Xu, and M. Gerla. Group and swarm mobility models for ad hoc network scenarios using virtual tracks. *Military Communications Conference, MilCom*, 1:289–294, 2004.

[31] S. Ziane and A. Mellouk. A swarm intelligent multi-path routing for multimedia traffic over mobile ad hoc networks. In *Proceedings of the 1st ACM international workshop on Quality of service and security in wireless and mobile networks*, pages 55–62, 2005.