A Probabilistic Emergent Routing Algorithm for Mobile Ad Hoc Networks

John S. Baras, Harsh Mehta

A Probabilistic Emergent Routing Algorithm for Mobile Ad Hoc Networks

John S. Baras and Harsh Mehta
Department of Electrical and Computer Engineering
and the Institute for Systems Research
University of Maryland, College Park, MD 20742, USA

Abstract

Mobile ad hoc networks are infrastructure-less networks consisting of wireless, possibly mobile nodes which are organized in peer-to-peer and autonomous fashion. The highly dynamic topology, limited bandwidth availability and energy constraints make the routing problem a challenging one. In this paper we take a novel approach to the routing problem in MANETs by using swarm inteligence-inspired algorithms. The proposed algorithm uses Ant-like agents to discover and maintain paths in a MANET with dynamic topology. We present simulation results that measure the performance of our algorithm with respect to the characteristics of a MANET, the varying parameters of the algorithm itself as well as performance comparison with other well-known routing protocols.

1 Introduction

A substantial research effort has gone into the development of routing algorithms for MANETs. A number of routing algorithms have been proposed. Some of these are DSDV, OLSR, CGSR, AODV, DSR, TORA, ZRP, LAR and several others [11, 13, 14, 15]. These protocols can generally be categorized as either *proactive* or *reactive* protocols. Proactive protocols build routes in the network constantly, even though there might not be packets to be transmitted between a certain set of nodes. Reactive (on-demand) protocols, on the other hand, attempt to establish multihop between pairs of nodes only when there are packets to be exchanged between these pairs of nodes. Recently there has been great interest in so called Swarm Intelligence [1], [2]; a set of methods to solve hard static and dynamic optimization problems using cooperative agents (usually called ants, since the method was inspired from collaborative efforts in insects). Ant-inspired routing algorithms were developed and tested by British Telecomm and NTT for both fixed and cellular networks with superior results [3, 4, 5, 6, 7, 8, 9, 10]. AntNet, a particular such algorithm, was tested in routing for data communication networks [3]. The algorithm performed better than OSPF, asynchronous distributed Bellman-Ford with dynamic metrics, shortest path with dynamic cost metric, Q-R algorithm and predictive Q-R algorithm [1, 3, 4, 5, 6, 7, 8].

MANETs operate in a distributed and asynchronous manner. Inspired by the success of ant-agent algorithms in routing and load balancing for fixed communication networks we first proposed applications of the swarm-intelligence ideas for dynamic adaptive routing in in MANETs in the proposal [16]. We initiated research on these ideas since October 2000, and a first presentation of our results was given in the seminar [18]. Interest in applications of ant-based routing in MANETs has risen and several papers have appeared recently on the subject [17, 19, 20]. For instance, Gunes *et al.* have proposed an Ant-based approach to routing in MANETs in [19]. Their approach uses ants only for building routes initially and hence is a completely reactive algorithm. They have also shown some performance comparisons with other MANET routing protocols based on the pause time of mobile nodes. Marwaha *et al.* [20] have explored a hybrid approach using both AODV and Ant-based exploration.

In our research we discovered early that there are two central challenges in making the promising swarm-intelligence ideas work successfully in the difficult domain of MANET routing. The first is to get the computations in such a form and implementation so as to to be fast and fast converging. This is necessary given the mobile nature of MANETs and the resulting changing topology. The second and most serious is to reduce the overhead (OH) created by these proactive algorithms. Straightforward application of ant-based routing, like AnNet or other algorithms that were successful in fixed topology networks, does not work well in MANETs due to a large OH. We addressed both challenges in our research to date on this problem. This paper describes primarily new methods and associated evaluations for combating the second and most serious challenge. We first describe an algorithm based on swarm-intelligence based on unicast communications for control and signaling packets (ants). We compare this algorithm (after improvements) to AODV (a popular routing algorithm for MANETs [14, 15]) and show that the overhead requirement of the swarm-intelligence algorithm significantly harms its competitiveness. Then we describe a new algorithm which utilizes the inherent broadcast nature of wireless networks to multicast control and signalling packets (ants). This second algorithm competes well with AODV and we show here several comparisons by simluations in a standard benchmark for MANETs [12, 15, 22]. We describe several additional innovations we have introduced in both algorithms and in particular the advantage o discovering, storing and using multiple (ranked) paths between source-destination pairs. For more details on our new algorithms and their

performance evaluation we refer to [21].

Our approach and methodology has a strong distributed optimization foundation, which leads towards promising analytical treatment; work on this is under way and will be reported elsewhere. In addition we have shown, in a different part of our recent work, that these new algorithms have superior routing security properties; a significant discovery given the weak state of affairs regarding security in all existing MANET routing protocols and routing protocols at large. Our work in security and trust for MANETs has introduced an important innovation via a novel optimization framework for security and trust. For further details about our results on security and trust in MANETs we refer to [23, 24].

2 A swarm intelligence based unicast algorithm for MANETs

In this section, we describe a routing algorithm for Mobile Ad Hoc Networks based on the swarm intelligence paradigm and similar to the swarm intelligence algorithms described in [3, 9]. The algorithm uses three kinds of agents - regular forward ants, uniform forward ants and backward ants. Uniform and regular forward ants are agents (routing packets) that are of *unicast* type. These agents *proactively* explore and reinforce available paths in the network. They create a probability distribution at each node for its neighbors. The probability or goodness value at a node for its neighbor reflects the likelihood of a data packet reaching its destination by taking the neighbor as a next hop. Backward ants are utilized to propagate the information collected by forward ants through the network and to adjust the routing table entries according to the perceived network status. Nodes proactively and periodically send out forward regular and uniform ants to randomly chosen destinations. Thus, regardless of whether a packet needs to be sent from a node to another node in the network, each node creates and periodically updates the routing tables to all the other nodes in the network.

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

2.1 The Operation of the algorithm

2.1.1 Bootstrapping of the routing tables

Initialization and neighbor discovery is done by single-hop, broadcast $HELLO$ messages that are transmitted periodically at an interval of $HELLO\_INTERVAL$ seconds. These messages are used at nodes to build the neighbor list, which is then used for the initialization of the routing table. The initial bootstrapping of the routing tables is done at a node when the first forward ant is being sent out to a certain destination.

At this time, there are no routing table entries (i.e. no probabilities for next hops) for that particular source-destination pair. The creation of the first forward ant at a node for the source-destination pair causes the routing table entries to be initialized with probabilities $1/N$ for each neighbor as the next hop for the respective destination, where N is the number of neighbors of the node where the routing table is being established. The uniform probabilities assigned to all the neighbors indicate that nothing is known about the state of the network. These probabilities are then adjusted by backward ants, when backward ants from the destination are received at the source node.

2.1.2 The routing table

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

The routing tables then contain the following data structures:

- The probability (goodness value) of taking as next hop node f at a node n , P^n_{fd} to eventually reach a certain destination d.
- The mean and the variance, (μ_{nd}, σ_{nd}) at node n to reach destination d.

2.1.3 Forward ants

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

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

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

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

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

2.2 Routing the Forward ants

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

is *probabilistic* and allows exploration of paths available in the network.

These agents are henceforth referred to as *Regular Ants*, similar to [9] to distinguish them from *Forward Uniform Ants*. When a forward ant is received at a node, it checks to see if it has previously traversed the node. If it has not previously traversed the node, the IP address of the node and the current time are pushed into the stack of the ant. In case the node IP address is found in its existing stack, the forward ant has gone into a loop and is destroyed.

2.2.1 Uniform ants

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

To make the algorithm fully adaptive to mobility and topology changes, we introduce another set of ants, called uniform ants. These are similar to the agents proposed in [9]. X % of the time, instead of creating regular ants, each node sends out uniform ants. These are created in the same manner as regular ants, however they are routed differently. Instead of using the routing tables at each node, they choose the next hop with uniform probability. If the current node has N neighbors, then the probability of taking a neighbor as the next hop is $1/N$. This is in contrast to regular ants which prefer taking next hops with higher probabilities more often.

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

2.2.2 Backward ants

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

The purpose of the backward ant is to propagate information regarding the state of the network gathered by the forward ants. The backward ant retraces the path of the forward ant by popping the stack, making modifications in the routing tables and statistic tables at each intermediate node according to one of the following learning rules:

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

$$P_{nd} \leftarrow P_{nd} - rP_{nd} \tag{2}$$

Here r is the reinforcement parameter.

2.
$$P_{fd} \leftarrow \frac{(P_{fd} + r)}{(1 + r)} \tag{3}$$

$$P_{nd} \leftarrow \frac{P_{nd}}{(1 + r)} \tag{4}$$

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

$$r = \frac{k}{f(c)},\ k > 0 \tag{5}$$

Here $f(c)$ is a monotone function of the metric and k is a constant. The backward ant also updates the existing estimates of the forward trip time at the source node as well as intermediate nodes. The trip time of this backward ant is used to update the statistics.

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

$$\mu_{kd} \leftarrow \mu_{kd} + \eta(o_{k\rightarrow d} - \mu_{kd}) \tag{6}$$

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

and

$$\sigma^2_{kd} \leftarrow \sigma^2_{kd} + \eta((o_{k\rightarrow d} - \mu_{kd})^2 - \sigma^2_{kd}) \tag{7}$$

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

2.2.3 Changes in routing tables due to node mobility

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

$$p_A = \delta, \delta \ll 1 \tag{8}$$

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

$$p_i = p_i - \frac{\delta}{N} \tag{9}$$

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

This allows the exploration of new routes through node A from node B by regular ants. If new and efficient routes are

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

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

2.2.4 Routing data packets

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

2.3 Algorithm parameters and other issues

The unicast routing algorithm proposed here has three key parameters:

- *The rate at which forward regular and uniform ants are sent*. This determines how quickly the algorithm discovers new paths, reinforces existing paths, destroys unavailable paths and adapts to new network topologies.
- *The percentage of forward and uniform ants*. This determines the balance between reinforcing already discovered routes and discovering new routes.
- *The update function and reinforcement*, used by the backward ants to reinforce the routing probabilities at each node.

2.3.1 Broadcast vs unicast in the MANET environment

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

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

3 The Probabilistic Emergent Routing Algorithm (PERA) for Mobile Ad Hoc Networks

In this section, we propose an algorithm based on the Swarm Intelligence paradigm that exploits the inherent broadcast capability available in the wireless environment. In this approach, the process of route discovery is carried out by using a flooding approach to obtain and maintain paths between source-destination pairs in the network.

Route discovery in the algorithm is done by two kinds of agents or ants - forward and backward. Uniform ants are no longer required or feasible as the forward ants are now broadcast rather than unicast. These agents create and adjust a probability distribution at each node for the node's neighbors. The agent packets, or *Ants* are of a relatively small (variable) size. The probability associated with a neighbor reflects the relative likelihood of that neighbor forwarding and eventually delivering the packet. Further, multiple routes between the source and the destination are created.

3.1 Bootstrapping the routing tables

As in the previous algorithm, neighbor discovery is done using 'HELLO' broadcast messages. However, the routing table entry for a destination is initialized at a node only after receiving a backward ant from the destinaiton. The routing table is of the same form as for the unicast algorithm.

The initialization of the routing table is done by incorporating all the neighbors of node n in the routing table. Each node is assigned an initial probability $1/N$, where N is the number of neighbors of node n. The routing tables are then modified to give a higher probability to the node that the backward ant just came from, as discussed in section 2.2.2, establishing a path toward the destination.

When the metric under consideration is delay, on the receipt of the first backward ant, the value of the time taken by the ant to travel to the destination from the current node, $T_{n \to d}$ is assigned to the mean, μ_{nd} and the variance, σ^2_{nd} is assigned a value of zero. Modifications to $(\mu_{nd}, \sigma^2_{nd})$ are made upon the arrival of later backward ants based on the learning rule as discussed in section 2.2.2. On the other hand, if the metric under consideration is the hop count for instance, the backward ants as well as the forward ants travel on high priority queues, leading to faster dissemination of information regarding the network status.

The routing table and the table of local statistics at each node can be visualized as in Figure 2.

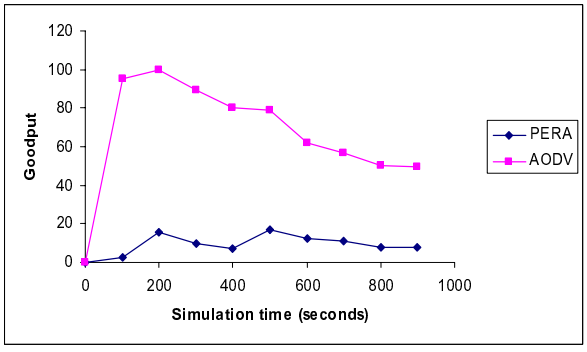

Figure 1: Goodput comparison of AODV and the unicast swarm based algorithm for 20 nodes in an area of 500m X 500m for speed of 1 m/s.

3.2 Forward ants

To carry out the process of Route Discovery, forward ants or agents are used. The forward ants are somewhat similar to the Route Request packets used by AODV [14] and DSR [11] routing protocols, but have some subtle differences.

Each forward ant contains the IP address of its source node, the IP address of the destination node, a sequence number, a hop count field and a dynamically growing stack. The stack contains information about the nodes that the forward ant traverses and the times at which these nodes have been traversed, ie. $(NODE\_ID, NODE\_TRAVERSAL\_TIME)$.

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

When a node receives a forward ant, it checks in the destination IP address field if the address corresponds to its own IP address. If the forward ant is not directed to the current node, the node pushes its own IP address and the time at which the ant was received at the node. Also, the hop count field of the forward ant is decremented by 1. Each forward ant is uniquely identified by the values of its source node IP address and the sequence number, i.e. the record $(Source\ IP\ address, Sequence\ Number)$. The sequence number for each ant is assigned at the source node and is unique for that source and forward ant. Thus, each node stores the pair $(Source\ IP\ address, Max\ Sequence\ number)$, where the $Max\ Sequence\ number$ is the highest value of the sequence number of an ant received from that source node. The node drops forward ants with a sequence number less than or equal to the $Max\ Sequence\ number$ that it receives from the same previous hop. This avoids the duplication of forward ants in the network as well as the establishment of only the best route through a node. If the $Max\ Sequence\ number$ value is greater than that previously recorded by the node, the node updates this value.

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

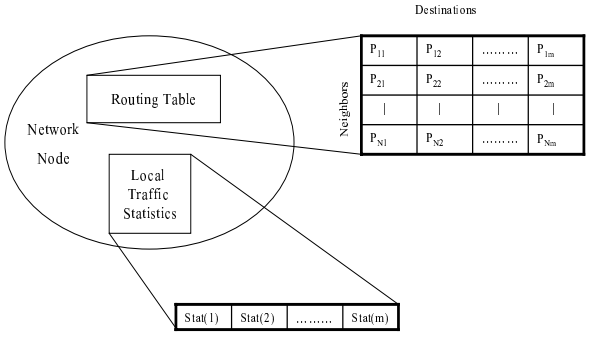

Figure 2: The routing table and local statistics maintained at each node.

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

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

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

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

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

3.3 Backward ants

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

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

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

The backward ant travels in *unicast* fashion back to the source node. It is forwarded on high priority queues. The stack of the forward ant is used to route it. Using the address at the top of the stack, the node forwards the backward ant to the correct next hop.

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

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

These values are used to update the routing and local statistics tables at the intermediate nodes f.

If routing table entries for destination d do not exist at node f, new ones are created with the neighbor list of the node f. All the neighboring nodes are given a probability of $1/N$, where N is the number of neighbors of the node f. The routing tables are then readjusted according to the probability rules discussed in section 2.2.2.

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

The update rules used are the same as those used for the previously discussed unicast algorithm and have been described in section 2.2.2.

To further illustrate the functioning of the algorithm for individual ants as well as individual nodes, Figure 3 depicts the algorithm flow for each ant, while Figure 4 depicts the algorithm flow at each node.

The changes required to the routing tables due to mobility of nodes are the same as for the unicast algorithm and have been discussed in section 2.2.3

3.4 Routing data packets

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

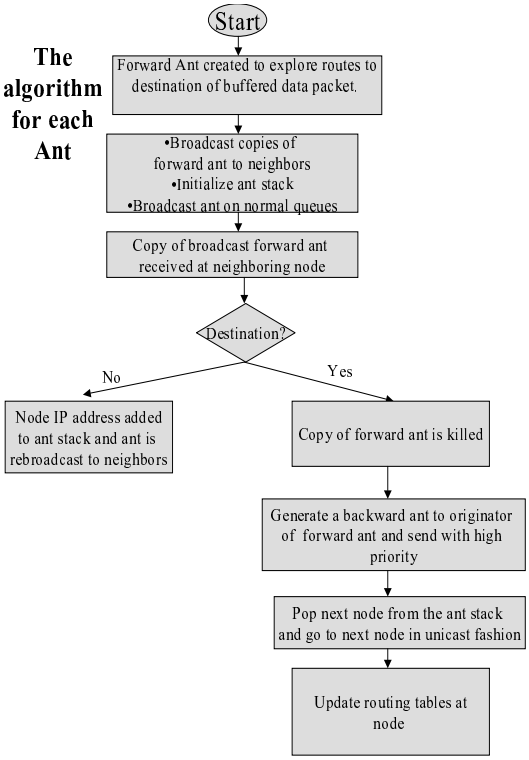

Figure 3: Algorithm for each ant.

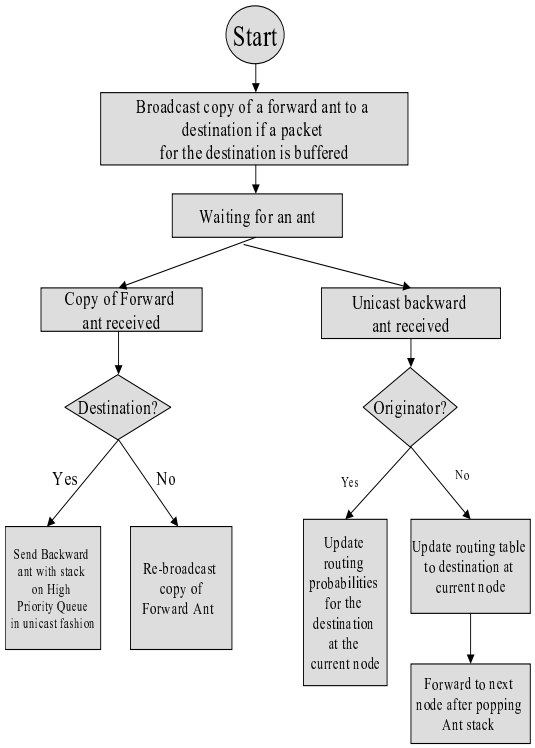

Figure 4: Algorithm at each node.

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

2. The data packets can also be routed probabilistically. Previous results [3] for swarm intelligence algorithms show excellent results for this method in the case of static networks with relatively small topologies. However, this might not be a suitable method for MANETs with rapid topology changes.

4 Simulation Results

Network Simulator 2 [22] discrete event simulator was used to simulate our algorithm. At the physical layer, radio propagation distance for each node was set to $250m$ and the channel capacity was $2Mbps$. Our model does not support radio capture [15] so, in the case of packet collisions all packets are dropped. The IEEE 802.11 Distributed Coordination Function (DCF) [12] as implemented in NS2 was used as the Medium Access Control (MAC) protocol. The communication medium is broadcast and nodes have bidirectional connectivity. Each simulation was run for 900 seconds. Multiple runs with different seed values were conducted for each scenario and the collected data were averaged over those runs. The algorithm was developed as a separate NS2 routing layer protocol. The mobility model used was the Random Waypoint model.

We use the throughput, the goodput and the average end-to-end packet transmission delay for comparisons. All the simulations were carried out with the same seed for the given simulation scenario and hence the results can be directly compared for the routing algorithms.

$$Goodput = \frac{Data\ packets\ received\ at\ routers * 100}{Total\ packets\ received\ at\ routers} \quad (10)$$

$$Throughput = \frac{Data\ packets\ recvd\ at\ destinations * 100}{Data\ packets\ sent\ from\ sources} \quad (11)$$

The end-to-end delay is the interval between the instant a source generates a packet and the time at which the destination receives the packet. The end-to-end delay is aggregated for each packet for each source-destination pair. The average per packet end-to-end delay through time intervals of 100 seconds is then calculated as the number of source-destination pairs and the number of packets received is known.

4.1 Hop count based optimization

In these experiments, we used the hop count as the metric for operation of the algorithm (instead of delay). The network consisted of 20 nodes, randomly placed in an area $500m$ x 500. 4 source and destination pairs were randomly chosen from these 20 nodes. Each source transmitted $1\ packet/sec$. Nodes in the simulation were mobile.

4.2 Mobility speed

In these experiments,the mobility speed was varied between 0 to $20\ m/s$, *i.e.*, . (0,5,10,20,15,20)m/s.

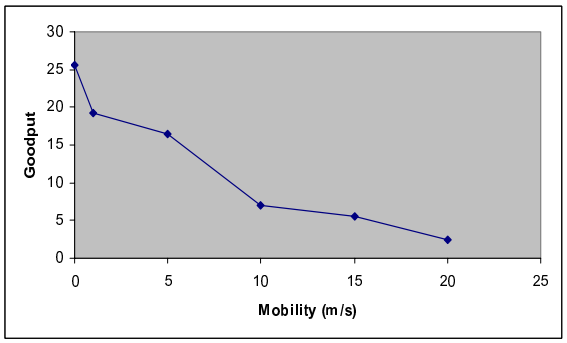

Figure 5: Variation in goodput with mobility.

Figure 5 shows the goodput as a function of the node mobility speed. It is seen that the goodput decreases with increase in mobility. This is to be expected since with an increase in mobility, a larger number of forward ants are required to be sent to discover new routes and modify and update existing routes which are no longer available for packet transmission.

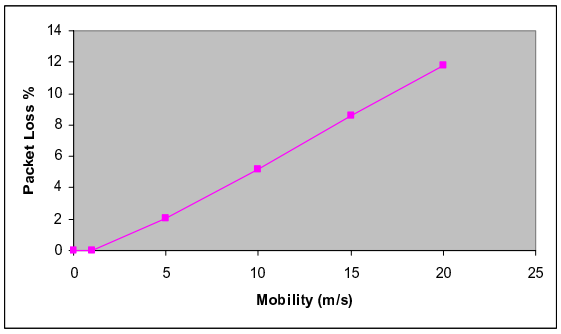

Figure 6: Percentage packet loss for varying mobility.

Figure 6 shows the percentage packet loss as a function of the mobility. With 0 and low mobility ($1\ m/s$), the packet loss is 0. With speeds of $5\ m/s$, the packet loss is under 2%. However, with increasing mobility, the packet loss increases linearly. Thus, even the increased rate of sending ants (as evidenced by the decreased goodput) does not serve to maintain a low percentage of packet loss. To keep the packet losses low, the rate of sending ants has to be increased non-linearly.

4.3 Rate of sending forward ants

In these experiments the rate of sending forward ants was varied for different mobility speeds and the behavior of the algorithm was studied.

Table 1 shows the variation in goodput and percentage packet loss as a function of the $ANT\_INTERVAL$ (the time period between the transmission of two forward ants) for 20 nodes in an area of $500m\ X500m$ with speeds of

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

Table 1: Goodput and % Packet Loss with ANT_INTERVAL with mobility of 10 m/s.

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

Table 2: Goodput and % Packet Loss as functions of ANT_INTERVAL for mobility of 5 m/s.

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

Table 2 shows similar results with speeds of $5m/s$. For a low value of $ANT\_INTERVAL$, the packet loss is lower than for a higher value. Further, it is important to note that the packet loss for values of $ANT\_INTERVAL$ 100 and 150 are the same. This is because the increase in the number of forward ants that are sent is not sufficient to cause an increase in performance in terms of goodput and packet loss. The goodput therefore goes down since the packet loss remains constant.

Table 3 shows similar results with speeds of $1m/s$. Since the mobility is very low, the adaptivity required of the algorithm is relatively low. Even by sending ants at a higher rate, there is no change in the packet loss, since a single forward ant sent at the start of the simulation obtains enough data for all data packets to be successfully routed.

4.4 Reinforcement

The learning rule used in our experiment is rule 2 in section 2.2.2, which allows using a cost function, $f(c)$ as described in section 2.2.2. In this experiment the value of the reinforcement used to update the routing tables at the nodes is varied between 0.1 and 0.5 (0.1, 0.15, 0.20, 0.30, 0.40, 0.50).

Figure 7 shows the variation of the goodput as a function

ANT_INTERVAL	Goodput	% Packet Loss
300	17.23	0
900	19.18	0

Table 3: Goodput and % Packet Loss as functions of mobility with 1 m/s.

of the value of the reinforcement. The speeds of the nodes are $5\ m/s$ and $10\ m/s$. The goodput is higher for speed $5\ m/s$, due to the fewer number of ant packets required to discover available routes.

Figure 8 shows the variation of the percentage packet loss as a function of the value of the reinforcement for speeds of $5\,m/s$ and $10\,m/s$. As expected, the packet loss is lower for lower speed of movement. However, it is important to note that with an increasing value of the reinforcement applied, the packet loss first increases and then decreases. This is because a weak (small) applied reinforcement implies that routes do not get positively reinforced to a sufficiently high degree. In the situation where mobility exists in the network, this reduces the adaptivity of the algortihm, leading to stale routes being used for the transmission of data packets. If the reinforcement applied is increased beyond a certain value, it causes the routes to be reinforced too fast. This leads to routes that may not actually be the best routes being used for the transmission of data packets.

5 Comparison with AODV

We compared the algorithm proposed in section 3 with AODV [14, 15] in terms of throughput, delay and goodput.

5.1 Goodput comparison

Figure 9 shows a comparison of the goodput for AODV and PERA for a scenario with 20 nodes in an area of

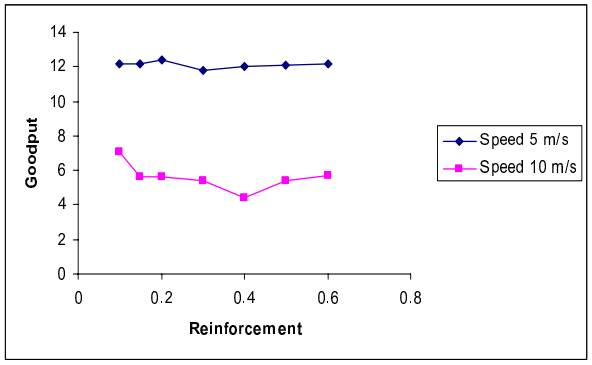

Figure 7: Variation in goodput vs reinforcement parameter.

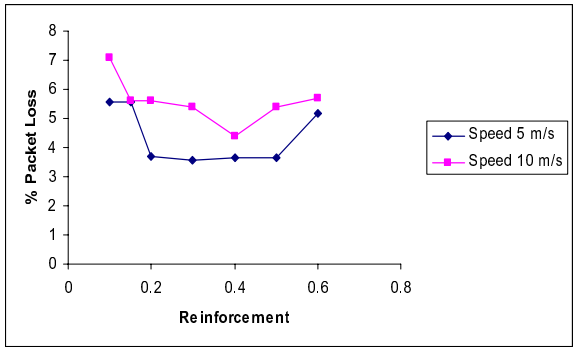

Figure 8: Percentage packet loss with varying reinforcement parameter.

$500m\ X\ 500m$ with the nodes moving with speeds of $1\ m/s$ and a pause time of $100\ secs$. Since the mobility is low, the overall goodput for both algorithms is high.

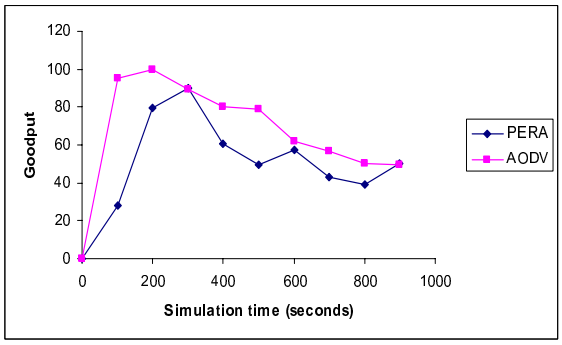

Figure 9: Goodput comp. of PERA and AODV at 1 m/s.

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

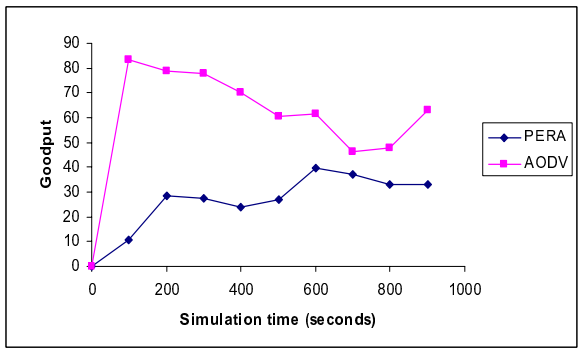

Figure 10: Goodput comp. of AODV and PERA at 10 m/s.

5.2 Throughput

Figures 11 and 12 show the throughput comparisons for AODV and PERA for mobility speeds of $1\ m/s$ and $10\ m/s$ and pause time $100\ secs$. At the lower speed, the throughput is the same for both AODV and PERA, however, at the higher speed, the throughput is slightly less for PERA in some cases. This is because with mobility, PERA adjusts gradually to the changes in topology.

5.3 Delay

Figures 13 and 14 show the comparison of delay for AODV and PERA. Both algorithms show a large initial delay, which is required for routes to be set up. Subsequently, AODV shows large delays again in situations with high mobility. PERA on the other hand, shows low delays in all cases, as instead of buffering data packets until a new route id found, PERA delivers the data packet through an alternate route.

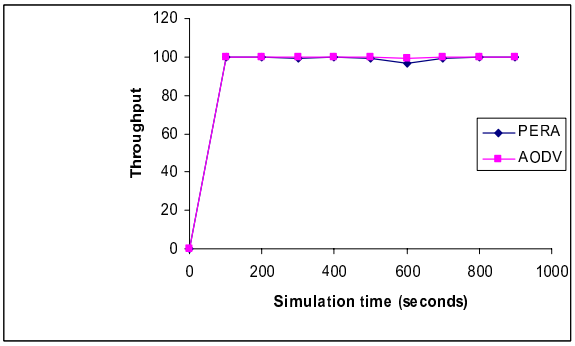

Figure 11: Throughput comp. AODV/PERA, 1 m/s.

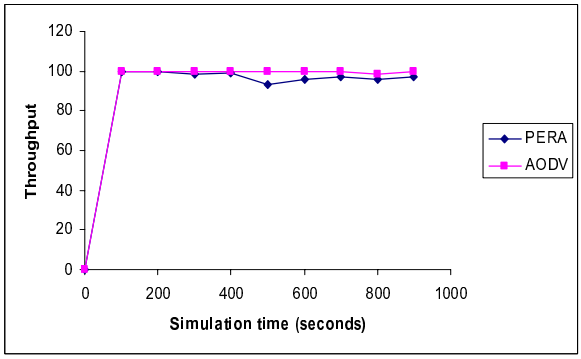

Figure 12: Throughput comp. AODV/PERA, 10 m/s.

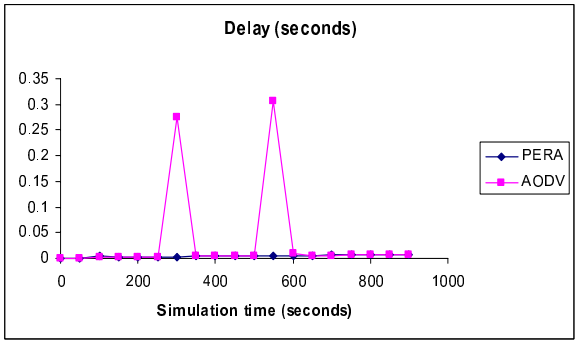

Figure 13: Delay comp. of AODV and PERA at 1 m/s.

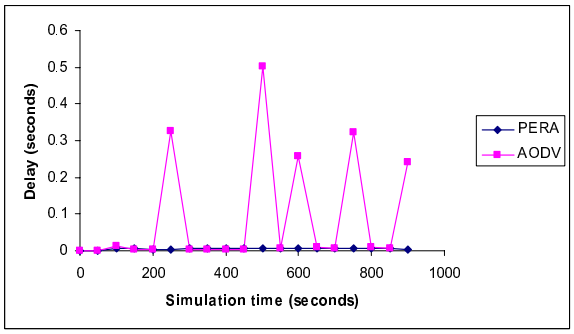

Figure 14: Delay comp. of AODV and PERA at 10 m/s.

6 Conclusion

In this paper we have proposed a set of routing algorithms for MANETs based on the swarm intelligence paradigm. In our experiments we observe that end-to-end delay for swarm based routing is low compared to AODV. However, the goodput for these algorithms is lower than for AODV in scenarios with high mobility.

Acknowledgement

This research was supported in part through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011.

References

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

[2] M. Dorigo and G. DiCaro, "Ant Colony Optimization: a New Meta-Heuristic", *Proc. 1999 Congress on Evolutionary Computation*, July 6-9, 1999, pp. 1470-1477.

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

[4] G. DiCaro and M. Dorigo, "Ant Colonies for Adaptive Routing in Packet-Switched Communications Networks", *Proc. PPSN V - Fifth Internatinal Conference on Parallel Problem Solving from Nature*, pp. 673-682, Amsterdam, Holland, September 27-30, 1998.

[5] G. DiCaro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communication Networks", *Journal of Artificial Intelligence Research*, vol. 9, pp. 317-365, 1998.

[6] E. Bonabeau, F. Henaux, S. Guerin, D. Snyers, P. Kuntz and G. Theraulaz, "Routing in Telecommunications Networks with "Smart" Ant-like Agents", *Proc. of 2nd International Workshop on Intelligent Agents for Telecommunications Applications '98*, Paris, France, July, 1998.

[7] M. Heusse, D. Snyers, S. Guerin and P. Kuntz, "Adaptive Agent-Driven Routing and Load Balancing in Communication Networks", *Proc. ANTS '98, First International Workshop on Ant Colony Optimization*, Brussels, Belgium, October 15-16, 1998.

[8] R. Schoonderwoerd, O. E. Holland and J. L. Bruten, "Ant-Like Agents for Load Balancing in Telecommunications Networks", *Proc. First ACM International Conference on Autonomous Agents*, pp. 209-216, Marina del Rey, California, 1997.

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

[10] T. White and B. Pagurek, "Towards Multi-Swarm Problem Solving in Networks", *Proc. Third International Conference on Multi-Agent Systems (ICMAS '98)*, July 1998, pp. 333-340.

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

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

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

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

[15] C.E.Perkins (Edt.), *Ad Hoc Networking*, Addison Wesley, 2001.

[16] J. S. Baras, "Dynamic Adaptive Routing in MANETs", Proposal for a Collaborative Technology Alliance in Communications and Networking, Consortium led by Telcordia, with the University of Maryland as a partner, Technical Area 1, Project 1.2 write-up for Autoconfiguration and Dynamic Routing in MANETs, October 2000.

[17] I. Kassabalidis, M. A. El-Sharkawi, R. J. Marks II, P. Arabshahi and A. A. Gray, "Swarm Intelligence for Routing in Communication Networks", in *Proceedings of IEEE Globecom 2001*, San Antonio, Texas, 2001.

[18] J. S. Baras, "Dynamic Adaptive Routing in MANETs: New Algorithms Using Swarm Intelligence", Distinguished Lecture in the ARL CTA C & N Technical Talk Series, July 25, 2002; presentation available at www.isr.umd.edu/People/faculty/Baras.html.

[19] M. Gunes, U. Sorges and I. Bouazizi, "ARA - The Ant Colony Based Routing Algorithm for MANETs", in Stephan Olariu, editor, *Proceedings of the 2002 ICPP Workshop on Ad Hoc Networks (IWAHN 2002)*, pp. 79-85. IEEE Computer Society Press, August 2002.

[20] S. Marwaha, C. K. Tham and D. Srinivasan, "Mobile Agents Based Routing Protocol for Mobile Ad Hoc Networks", in *Proceedings of IEEE Globecom*, 2002.

[21] H. Mehta, *Dynamic Adaptive Routing in Mobile Ad Hoc Networks*, M.S. Thesis, ECE Dept., University of Maryland, December 2002.

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

[23] L. Eschenauer, V. Gligor and J. S. Baras, "On Trust Establishment in Mobile Ad-Hoc Networks", in *Security Protocols, Proc. of 10th International Workshop*, Springer Lecture Notes in Computer Science (LNCS), Cambride, UK, April, 2002.

[24] L. Eschenauer, *On Trust Establishment in Mobile Ad-Hoc Networks*, M.S. Thesis, ECE Dept, University of Maryland, May 2002.