A Dynamic Ant Colony Based Routing Algorithm for Mobile Ad-hoc Networks

EHSAN KHOSROWSHAHI-ASL AND MAJID NOORHOSSEINI AND ATIEH SABERI
ehsan@iranet.ir , majidnh@aut.ac.ir , atiehs@gmail.com
Department of Computer Engineering and Information Technology
Amirkabir University of Technology
Tehran, Iran

In this paper we study the performance of ant colony based routing algorithms in mobile ad hoc networks (MANETs) and present SAMP-DSR, a new algorithm to address the shortcomings of both ant colony based as well as the traditional DSR algorithms. Due to the unstable nature of MANETs memory based search algorithms such as ant based methods become inefficient and impractical as the network mobility increases. Our simulations show that ant based routing algorithms suffer from lack of accuracy while incurring extensive memory usage as well as valuable processing power and bandwidth costs in highly dynamic environments. The proposed ANT based routing algorithm exploits a dual-mode approach. Each node can independently operate in a local mode as well as in global mode using the information provided by ants. The network topology changes are constantly monitored. When the rate of topology change gets too high for ants to converge efficiently, a node switches to local mode to rely less on the information learnt through ants. Our simulations show how this switching approach significantly improves the performance of the network by quickly adapting to the dynamics of the environment.

Keywords: MANETs, Dynamic Source Routing, Ant Colony Optimization, Reliability, Quality of Service

1. INTRODUCTION

Routing in Mobile Ad Hoc networks (MANETs), due to their highly dynamic topology and limited bandwidth is a challenging issue. Routing protocols in traditional wired networks could exploit high processing power and bandwidths available for them to implement broadcasting link-state like protocols or high-process demanding AI based algorithms to solve quality of service related optimization problems. One of the routing protocols considering quality of service is AntNet[1]. AntNet is based on the shortest path behavior observed in ant colonies and on the related optimization framework of Ant Colony Optimization (ACO) [1].

It has been experimentally observed that ants in a colony can converge on moving

over the shortest among different paths connecting their nest to a source of food [2, 3]. Real ants wander stochastically around their nests to forage (Search for food). Upon finding food, they return back to their nests and simultaneously deposit pheromone trails along the paths. Since ants tend to follow the pheromone trails, they are more likely biased towards such paths, and as a result they may not keep on traveling quite randomly. Therefore, they likely move through these paths and reinforce the existent pheromone. This kind of indirect communication is called stigmergy[4] in the biology and entomology literatures. These simple behaviors are sources of inspiration for a class of algorithms called Ant Colony Optimization (ACO). ACO has been applied successfully to a variety of combinatorial problems such as traveling salesman, vehicle routing, job scheduling as well as to routing.

AntNet [5] is one of the well-known ACO based routing protocols introduced by M. Dorigo and G. Di Caro for packet switched networks. It is an alternative routing algorithm to the well-known OSPF protocol traditionally used for packet switched networks. AntNet, similar to most other ACO based routing algorithms introduced after AntNet, exhibits a number of interesting properties: it works in a fully distributed way, is highly adaptive to network and traffic changes, uses lightweight mobile agents (called ants) for active path sampling, is robust to agent failures, provides multipath routing, and automatically takes care of data load spreading. These properties are very appealing to distributed MANET routing. Many algorithms have already implemented ACO based routing for MANETs. Some of these algorithms like ARA[6] and PERA[7] have reduced the number of control packets (ants) being send pro-actively to overcome the bandwidth and energy limitations which degrade the overall network performance. AntHocNet[3] has proposed a comprehensive routing algorithm based on ACO for MANETs. AntHocNet is comprised of both a proactive and a reactive component. The reactive component is triggered during route setup phase at the start of a data session. The paths are then monitored, maintained and improved proactively while the data session is in progress. Therefore, the ants are not dispatched pro-actively to explore the network before the start of a data session and hence no topology and link-status information are learnt by the ants. This can mitigate the quality of paths discovered particularly when some QoS parameters other than delay is to be optimized. Since intermediate nodes have very limited information about network status, AntHocNet cannot perform much better than random broad-

casting algorithms like AODV and DSR. In [12-14] some improved ACO based methods to improve performance have been proposed but none of them control the number of ants periodically being produced. In [8] the authors show that the original AntNet algorithm with minor adjustments and improvements cannot improve overall performance over AODV or DSR in MANETs particularly on higher mobility and topology change rates. In a previous work we have introduced Enhanced Multipath Dynamic Source Routing (EMP-DSR) algorithm which uses ants pro-actively to feed its re-active path discovery process with global data[10]. EMP-DSR's re-active path discovery process is based on MP-DSR [9] which is a QoS-aware multipath source routing protocol, based on Dynamic Source Routing protocol (DSR). It is a fully distributed QoS protocol, which creates and selects routes based on the end-to-end reliability QoS parameter. MP-DSR routing decisions in source and intermediate nodes are based on local link connectivity and reliability information. However, selecting a reliable link in an intermediate node solely based on local information may not necessarily lead to a reliable end-to-end path. The problem originates from the fact that there is no global end-to-end reliability related information available to each node. To alleviate this problem EMP-DSR utilizes ants to gather global information about network status to find optimized routes to different destinations. However, EMP-DSR suffers from high overhead and convergence problems in networks where the topology change rate is high.

In this paper, we have analyzed how topology changes in MANETs affect the performance of ACO based routing algorithms. ACO like many other AI based search algorithms requires some time to converge. Drastic changes in search space can render the learned data and current search results obsolete. With the possibility of having scenarios of high topology change rates in MANETs it is essential to make some adjustments in the routing protocols which are based on AI search algorithms (ACO in our case). On high change rates, protocols need to switch back to fully reactive mode and stop sending control packets proactively. Route discovery decisions have to be made relying more on local information instead. Although in this paper we have focused on ACO, the proposed algorithm can be generalized for other distributed optimization solutions where as mentioned frequent changes in the network topology can nullify all the search efforts. We have adopted our previous work EMP-DSR as a ACO based routing protocol to analyze ACO based algorithms in highly changing environment and provide our solutions. As

mentioned above, EMP-DSR enhances MP-DSR with some global information provided by an ACO module reactively running in the background. MP-DSR tries to find a set of multiple disjoint paths which can satisfy a minimum specified end-to-end reliability. However both of these algorithms have some shortcomings in different network situations. The method proposed in this paper, the Switched ACO based MP-DSR (SAMP-DSR), tries to overcome these shortcomings by exploiting both methods. The two aforementioned protocols have similar goal, which is satisfying a minimum end-to end reliability through a set of disjoint paths, while one relies on local node information and the other relies on global information provided by ants.

SAMP-DSR utilizes ants when they can be useful and stops sending them and relies more on local node information when topology changes are high. The rest of paper is organized as follows: Section 2 briefly describes MP-DSR and ACO. Section 3 explains SAMP-DSR in detail. Section 4 shows the simulation results, and finally section 5 concludes the paper.

2. PRELIMINARIES

In this section, the fundamental concepts and approaches upon which our proposed method is based are introduced. MP-DSR as one of the base routing methods used by SAMP-DSR is described in subsection 2.1. Section 2.2 provides a brief introduction to Ant Colony Optimization and AntNet.

2.1 MP-DSR

MP-DSR or Multi-Path Dynamic Source Routing introduced in [8], tries to find multiple disjoint paths from a given source to a destination while guaranteeing that these paths altogether satisfy a given end-to-end reliability P_u where $0 < P_u \leq 1$. MP-DSR achieves route discovery by determining the number of paths to be discovered (m_0) and the lowest reliability (P_{lower}) that each of the m_0 paths must have so that the P_u reliability can be satisfied. Once, the appropriate values for the parameters at the source node is calculated, the source node sends m0 Route Request (RREQ) messages to search for feasible paths. Each RREQ message contains useful information such as, the path it has traversed so far (*pathVector*), the *pathVector's* corresponding accumulated reliability P_{acc} and the lowest reliability P_{lower}. When an intermediate node receives an RREQ message,

it investigates whether or not P_{acc} is greater than P_{lower}. If pace is less than P_{lower}, the message is discarded. Otherwise, it appends itself to the *pathVector* and calculates the new Pacc and then forwards at most m_0 copies of the modified message to its neighbors. After a while, the destination node collects some of the RREQ messages. Using the *pathVectors* stored in RREQ messages, the destination node uses a path selection algorithm to pick the set of disjoint paths that can satisfy P_u. Two paths are considered disjoint if they only share a source and a destination node and not any intermediate node. The destination node subsequently responds to the source by sending Route Reply (RREP) messages through the selected paths. The source node commences sending the data packets through the selected paths thereafter.

2.2 Ant Colony Optimization (ACO)

ACO, a famous swarm intelligence approach, has taken its inspiration from the social behaviors of real world ants. Most often real ants are wandering stochastically around their nests to forage. Upon finding food, they return back to their nests and simultaneously deposit pheromone trails along the paths. Since ants tend to follow the pheromone trails, they more likely biased towards such paths, and as a result they may not keep on traveling quite randomly. Therefore, they likely move through these paths and reinforce the existent pheromone. This kind of indirect communication is called stigmergy [5] in the biology and entomology literatures. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. Analogously, ACO [4], one of the state-of-the-art paradigms in designing metaheuristic algorithms for combinatorial optimization problems, utilizes artificial mobile agents namely ants which are capable of solving various kinds of routing and congestion problems. At regular intervals from every network node several ants are launched toward the destination node to discover the feasible low cost path to that node. Each ant in ACO considers two parameters to select its next hop. The first one is the amount of pheromone deposited on the path to the next node, and the other is a kind of heuristic parameter such as the queue length associated with the link. AntNet [5] is an ACO approach to adaptive learning of routing tables in communication networks. Each node k in the network stores some data structures within itself which are responsible for keeping local traffic statistics, and routing table. Local traffic statistics defines a simple parametric statistical model for traffic distribution over the network as seen by node *k*. In fact, it keeps track of the amount of traf-

fic flows towards each possible destination. Routing table, for each possible destination d and for each node n, stores a probability value P_{nd} which expresses the desirability of selecting n as the next node when the destination node is d. In fact it shows amount of pheromone deposited on the link (k,n). When an ant at node k heads toward a destination node d, it selects the next neighbor node n with the probability P'_{nd} where we have:

$$P'_{nd} = \frac{p_{nd} + \alpha \times l_n}{1 + \alpha \times \left(\left|N_k\right| - 1\right)} \qquad (1)$$

where:

$$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{|N_k|} q'_{n}} \qquad (2)$$

Where $|N_k|$ is the number of the neighbors of node k, q_n is the length of the queue associated with the link connecting k to n and α is the weight of the importance of the heuristic function with respect to the pheromone deposit. When an ant reaches the destination node, it can then evaluate the goodness of the path. The goodness of the path can be defined according to an application's requirement. AntNet itself uses the ants trip time and the parameters of the local statistical model. Quantifying the goodness value, the destination node creates a backward ant. The backward ant takes the exact path by the corresponding ant, but in the opposite direction, and deposits an amount of pheromone on its path to the source node. This amount is commensurate with the goodness of the path.

3. SAMP-DSR

MP-DSR tries to compute a set of unicast routes that can satisfy a minimum end-to-end reliability requirement. It then maintains this requirement throughout the lifetime of transmission. MP-DSR is fed with just local information for discovery of routes. Following a high reliable link which connects the current node to one of its neighbors, cannot guarantee an end-to-end reliable path, since the selected neighbor may not have good links towards destination node. As a result the path with the required end-to-end

reliability might not be achievable. In a previous work we have introduced EMP-DSR that fortifies MP-DSR by providing it with global information using an ACO-based algorithm. EMP-DSR uses an ACO-based algorithm running silently in the background to provide the required global information for MP-DSR. However, as mentioned ACO-based algorithms in general suffer from low performance in highly dynamic environments.

To mitigate the above problems, we have introduced SAMP-DSR which is based on our previous work EMP-DSR [10]. The approach presented here is a QoS-aware routing protocol, considering end-to-end reliability parameter as a constraint while trying to minimize other parameters such as delay. Our goal in this paper is to analyze the performance of ACO-based routing algorithms (here EMP-DSR) as the rate of topology change increases, and then propose a solution to deal with these topology changes.

SAMP-DSR consists of two main components. The first component is the proactive ACO-based component running all time in the background and the later is the reactive path discovery component which is triggered upon a route request. In addition, SAMP-DSR has two working modes, one for when network is stable enough for ACO to perform well and one for when network mobility is high. Details of both working modes and both components are described in following sections.

3.1 ACO mode and local mode

SAMP-DSR has two working modes namely ACO mode and local mode. In ACO mode the ACO-based proactive component plays a dominant role in providing global information and path selection preferences. On the other hand, in local mode the decisions are based on local information which each node gathers solely from its own neighbors. Each node selects its mode of operation independently. Therefore at any given time different nodes of the same network would be working in any of the two modes based on its own selection.

3.2 ACO-based proactive component

SAMP-DSR has two working modes namely ACO mode and local mode. In ACO mode the ACO-based proactive component plays a dominant role in providing global

information and path selection preferences. On the other hand, in local mode the decisions are based on local information which each node gathers solely from its own neighbors. Each node selects its mode of operation independently. Therefore at any given time different nodes of the same network would be working in any of the two modes based on its own selection.

SAMP-DSR uses an improved version of AntNet to populate its own routing table. The most significant change made to AntNet goes towards the way the goodness value is obtained. In our approach, we have used end-to-end reliability as the goodness parameter. A forward ant starts the trip to the destination node, and then it arrives at an intermediate node, where it needs to update its P_{acc} field, by multiplying P_{acc} with the link availability of the link the forward ant has just traversed. This process continues until the ant eventually reaches its destination node. As a result, acc at the destination is the end-to-end traversed path reliability. Using P_{acc} and trip time (T) obtained from all forward ants during an observation windows; the paths goodness is calculated from equation below:

$$\text{PathGoodness} = c1(P_{acc}) + c2\left(\frac{W_{best}}{T}\right) \qquad (3)$$

where W_{best} is the best trip time experienced by all the forward ants traveling toward the destination over the last observation window. The maximum size of the window is set to a constant. The first term is the end-to-end path reliability. The second term implies the goodness of the round trip time (T) relative to recent best trip time (W_{best}). In a typical application where reliability is more important the first term is usually given a higher weight. The coefficient c_1 and c_2 weigh the importance of each term. In our implementation we have set $c_1 = 0.6$ and $c_2 = 0.4$. The second term alleviates the stagnation problem [5], by dispatching some of the ants to less reliable but less congested or shorter paths. Stagnation occurs when ants are attracted by one optimal path which will then cause this path to be heavily congested.

After the forward ant reaches the destination node, the destination node generates a backward ant and transfers all the information of the forward ant to the new ant. The backward ant takes the same path as that of its corresponding forward ant, but in the opposite direction. The backward ant updates the routing table at intermediate nodes for all the entries related to the forward ant's destination node. As a result of this update the related probabilities will be increased by a value proportional to both path goodness and

the previous value of the probability (P_{nd}) using equation 4.

$$P_{nd} = P_{nd} + PathGoodness \times (1 - P_{nd}) \quad (4)$$

Launching several ants at regular intervals at different nodes, the whole process continuously runs in the background and updates routing tables. In local mode, however, the rate of launching of ants is decreased significantly. Although it seems that it would be best to completely stop sending ants in local mode so that we have no ant overhead but as discussed in detail in experiments section ants are our best source of feedback about the network status; the information gathered by them determines whether the node should operate in local mode or in ACO mode.

3.3 Reactive path discovery component

Upon receiving a connection request from an application with a certain reliability criterion (P_u), the source node initiates the *PathDiscovery* algorithm with the given P_u (shown in Figure 1). *PathDiscovery* process needs to determine the number of paths (m_0), the minimum reliability that each of the m_0 paths requires to guarantee P_u (P_{lower}), and the time window that this end-to-end reliability holds (t_w). P_{lower} is calculated from the given P_u and m_0 through equation below [9]:

$$\prod_{lower} = 1 - \sqrt[m_0]{1 - p_u} \quad (5)$$

Upon In *PathDiscovery*, t_w is initially set to the constant t_{wMax} (100 seconds in our implementation), the upper-bound of the time window. m_0 is initialized to 1 from which P_{lower} is calculated (line 8). Then, procedure iterates through all links associated with the neighbors of the source node and checks if there are m_0 numbers of end-to-end paths with at least P_{lower} reliability (lines 9-13). Basically, the procedure tries to find the minimum number of end-to-end paths which altogether can satisfy P_u. There is also a maximum threshold (*mmax*) for number of disjoint paths to be discovered (line 5). Thus, if m_0 violates *mmax* or the number of neighbors (L), the process will need to decrease t_w due to the fact that links can be more reliable in the shorter time window (line 31). However, it is preferred not to have smaller time windows, since source node is required to send route check messages to validate whether the end-to-end reliability can still satisfy P_u or not.

This validation incurs extra overhead especially when it is done at short intervals periodically. Therefore, a minimum threshold is adopted for t_w called t_{wMin} (here 20 seconds), if the algorithm fails for the minimum threshold then it is a failure for the application request (line 30). Otherwise, the source node will send RREQ messages to the m_0 highest reliable links. In ACO mode the highest reliable links are the ones who have the highest value in ant generated table for the requested end destination whereas in local mode the highest reliable links are determined based on local information gathered by directly communicating with neighbors (lines 14-28).

Upon receiving an RREQ message, each intermediate node runs *HandleRREQMessage* Algorithm (shown in Figure 2). The intermediate node first checks whether or not it is the destination. If not then the *HandleRREQMessage* algorithm will try to forward the RREQ towards the destination. An intermediate node only forwards m_0 instances of a RREQ messages, and discards other instances. This is accomplished via the bookkeeping variable *numOfForwardedMsg* (lines 1-2). In order to keep the required reliability criterion along the path, the RREQ contains a reliability field P_{acc} that keeps reliability value from source to the current node. Each time it passes a link, it multiplies P_{acc} with the links associated reliability it has just traversed (line 4). The procedure then calculates $L_i(t_w)$ according to equation 6 (line 5).

$$L_i(t_w) = \frac{\prod_{lower}}{\prod_{acc}(t_w)} \qquad (6)$$

With the current P_{acc}, $L_i(t_w)$ is the minimum reliability that the rest of path must hold in order to keep the end-to-end path reliability greater than or equal to P_u. The highest reliable links retrieved from ant populated table of the node, which also satisfy the $L_i(t_w)$, are picked and the RREQ messages are forwarded through them (lines 8-17). To avoid loops, the selected neighbors must not have been visited before (lines 9-10).

Algorithm 1 - PathDiscovery (P_u)

```
t_w = t_wMax
While t_w ≥ t_wMin do
    set A ∈ A_s,1(t_w), …, A_s,t(t_w);
    m_0 = 0;
    While m_0 ≤ L and m_0 ≤ m_max do
        path = 0;
        m_0 = m_0 + 1;
        ∏_lower = 1 − ⁿ√(1 − p_u) ;
        for ∀neighborj ∈ setA do
            if A_s,j(t_w) ≥ ∏_lower then
                path = path + 1;
            end if
        end for
        if path ≥ m_0 then
            Π_end-to-end = {(n_1,v_1), …, (n_n,v_n)};
            { where the list is sorted according local reliability value or ACO's proba-
          bility routing table depending whether its operating on local or ACO mode }
            rreq = newRREQ(m_0, Π_lower, t_w);
            n = numberOfNeighbors of source node;
            numRREQMsg = 0;
            for i = 1 to i = n do
                if A_s(t_w) ≥ Π_lower then
                    Send(RREQ, Π_endToend[i].neighbor);
                    numRREQMsg ++;
                    if numRREQMsg = m_0 then
                        return success;
                    end if
                end if
            end for
        end if
    end while
    return error
    t_w = 0.9 × t_w;
end while
```

Figure 1 – Path discovery initiation algorithm

Algorithm 2 - HandleRREQMessage(RREQ)

```
if numOfForwardedMsg[RREQ] > m0 then
    return ;
end if
Π_acc = Π_acc × A_{j,i}(t_w);
L_i(t_w) = ∏_lower / ∏_acc ;
n = num of the neighbors of the intermediate node;
P_end-to-end = {(n1,v1), …, (nn, vn)};
    { where the list is sorted according local reliability value or ACO's probability
routing table
        depending whether its operating on local or ACO mode }
 for  k = 1 to n do
    if P_end-to-end[k].neighbor ∈ pathVector then
        continue;
    end if
    if A_{j,k}(t_w) ≥ L_k(t_w) then
        numOfForwardedMsg[RREQ]++;
        forward(RREQ, P_end-to-end[k].neighbor);
        return ;
     end if
end for
```

Figure 2 – Handle message algorithm for intermediate nodes

When the destination node receives the first RREQ, it triggers a timer to somehow limit the time that should wait for the same consecutive RREQs. The higher the time limit, the greater the number of RREQs received at the destination. On the other hand, higher time limits increase path discovery delay. Therefore, it is advisable to determine the time limit according to the network size. As soon as the timer elapses, the destination node launches its path selection algorithm. Each of the RREQ messages contains a path with at least P_{lower} reliability. A set of paths (*TraceSet*) is then selected by the destination node, so that P_u is collectively satisfied. To attain this, it sorts all the available paths according to their reliability in descending order, and builds a *CandidateSet*. The path selection algorithm is recursive. At each iteration, the algorithm picks a new path from *CandidateSet* and tries to add it to the *TraceSet* upon condition that the new path is disjoint with respect to the previously added paths in *TraceSet*. The algorithm recursively invokes itself and attempts to add further paths to the *TraceSet*. If none of the paths remaining in the *CandidateSet* were eligible to be included in *TraceSet*, the algorithm re-

moves the previously added paths from *TraceSet*, and backtracks to look for other possible paths from *CandidateSet*. The whole procedure continues until either the algorithm finds an adequate number of paths to satisfy desired reliability, or exits with failure.

4. EXPIREMENTS

We evaluated SAMP-DSR algorithm and compared its performance to MP-DSR and EMP-DSR algorithms in a number of simulations. The terms of our evaluations are path discovery success ratio for a given P_u, average end-to-end delay per packet and delivery ratio. We have used Omnet++[11] with its mobilityFramework plug-in as our simulation tool. The environment consists of 100 nodes placed in random positions inside an area of 2000x2000 m2. 50 constant bit rate (CBR) UDP data flows are used. The MAC Layer is based on 802.11b protocol which has already been implemented in mobilityFramework plug-in. Channel date rate is set to 2Mbit/s. Transmitter power is set to an amount leading the radio propagation range of nodes to be around 187 meters. Random Waypoint Mobility algorithm with different speed and wait time parameters has been used in several experiments. The above values for the parameters have been used in all of our experiments. The performance of SAMP-DSR and AntHocNet on high mobility modes are examined in section 4.1. In section 4.2 the performance of SAMP-DSR algorithm is then compared with EMP-DSR, MP-DSR and some other well-known MANET routing protocols.

4.1 Case Study 1: ACO Based routing algorithms on high mobility modes

In this experiment we study the performance of ACO based algorithms in a high node mobility network. The results show how the performance of ACO based algorithms degrades as network mobility increases. We have compared EMP-DSR as an ACO based algorithm with MP-DSR since they are both analogous in goals and differ only in algorithm they use as one uses ACO. Therefore the difference in the performance is a direct

effect of the ACO module. In the first set of experiments Random Waypoint Mobility has been used to compare path discovery of EMP-DSR and EM-DSR. The stop time has been set to a short time of 2 seconds in order to emphasize more on node mobility and high change rates in network topology. The given P_u is 0.6. Figure 3 shows the percentage of successful path discoveries as a function the speed of node movement.

As expected the results show that when the topology changes slowly there is a higher chance for ants to converge to optimal solutions for providing routing algorithm with best possible end-to-end routes. As topology change rate increases long term learned data provided by ants becomes useless. Actually there is nothing to converge to when the search space itself is dynamic. So MP-DSR which is based on local information heuristics outperforms EMP-DSR on higher speed modes. As mentioned in section 3, SAMP-DSR can be set to work on high speed or low speed modes. A reasonable approach is to let nodes decide in which mode to work. Nodes need a readily available evaluation method in order to be able to determine network status. Experiments on different available parameters show that nodes can exploit ant messages' loss rate and the pheromone level of the best neighbor to determine the point where local based decisions outperform ACO based decisions. Keep in mind that nodes are independent in decision making and rely only on the data available to them. Therefore at times some nodes may operate in local mode whereas some others may be operating in ACO mode.

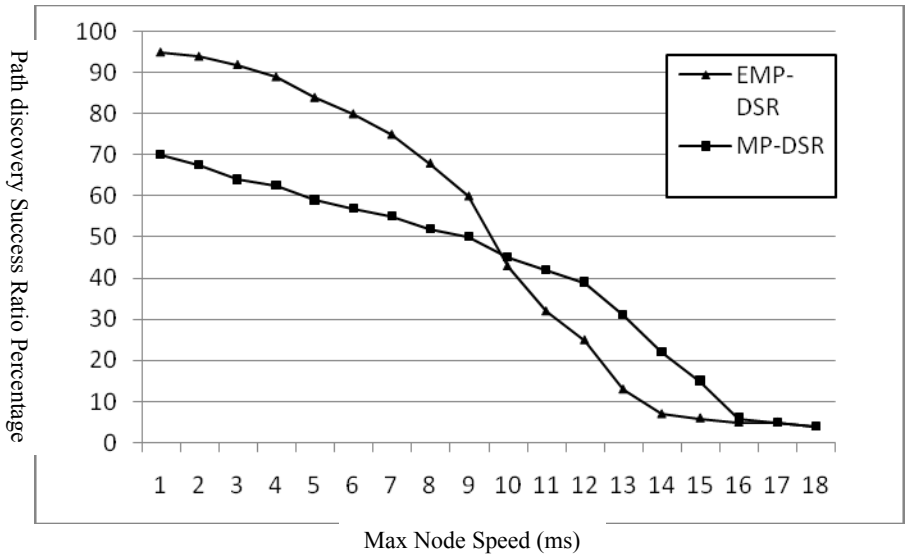

Figure 3 - Percentage of successful path discoveries as a function of node speed

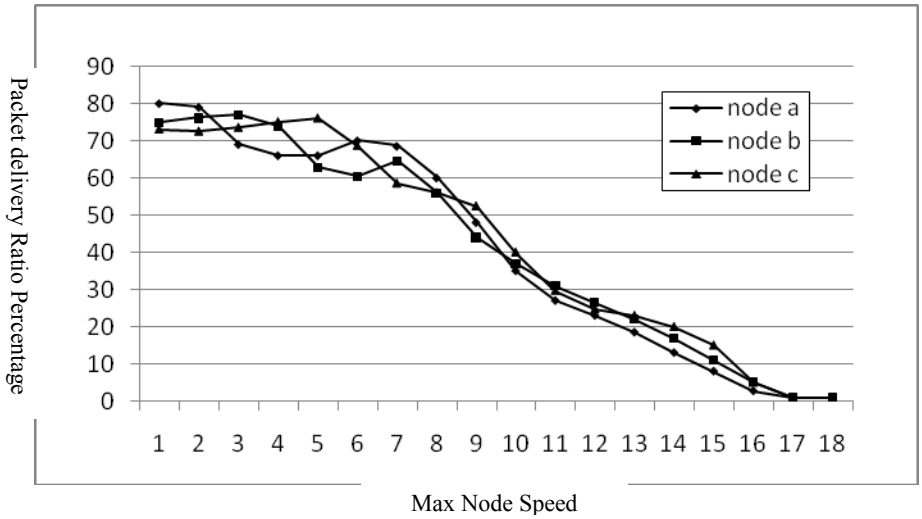

Figure 4 - The packet delivery ratio of ant packets experiences by three different nodes

Figures 4, shows the packet delivery ratio of ant packets experiences by three different nodes. According to results, as nodes mobility speed increases, the success rate of ant messages decreases accordingly. As you can see the three different nodes experience

different ant message loss ratio. The reason is that many other factors than node mobility affect loss rate. One of them is the average distance hop count of a node to its destination. Putting the results together, you can see that MP-DSR has better results when packet lost is around 35% for node a, around 40% for node *b* and around 37% for node *c*. Nodes can safely choose 35% as a threshold for switching between local mode and ACO mode in SAMP-DSR. We have chosen the lower probability of 35% favoring local mode since it also has less network and processing overhead. The other parameter that can assist in determining the mode of a node is the value of pheromone probability percentage of best neighbor. Ants tend to saturate pheromone values on only one path when the search space is static. But in a dynamic environment the saturation does not happen and therefore ACO does not converge to a unique final path. Thus as network topology rate increases, the values of pheromone probabilities will distribute over different paths. If it was only a few link breakages, ants would converge to the next best path fast enough. However this is impossible in situations where network topology changes are high rendering ants useless in those situations.

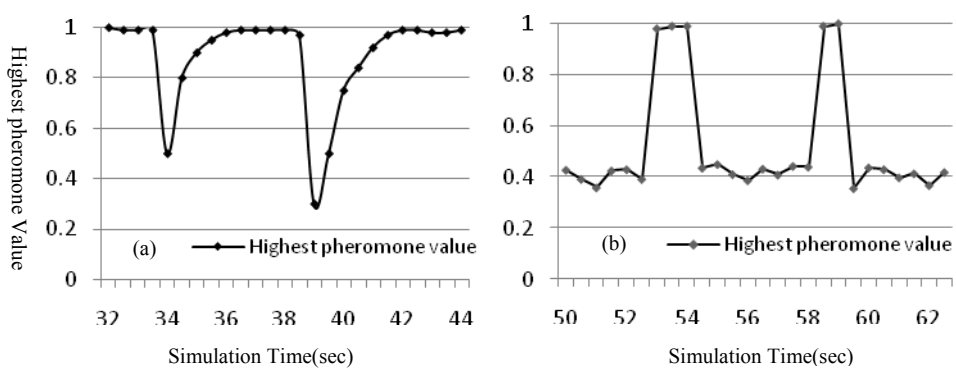

Figure 5 - Highest pheromone probability value for the simulated network of a random node

Figure 5a shows a snapshot of the highest pheromone probability value for the simulated network of a random node using Random Waypoint Mobility model with a speed of 2m/s. This movement speed is not causing rapid topology changes. When a link breakage along the pheromone saturated path happens, the ants converge to the second best path in a reasonable amount of time. Figure 5b shows the value of highest pheromone probability value of three different nodes in a simulated network with high mobili-

ty speed of 20m/s. Here ants have a hard time converging to a path. As you can see occasionally this value has risen up to 100% but that happens when node has only one neighbor in its radio range and the normalization function makes the probability value 100% instantly. These evaluations confirm that the density of pheromones is an indication of network topology change rate. An amount of less than 0.9 for the pheromone value in a reasonable time window is an indication of high mobility since ants couldn't converge to a unique path in that window. SAMP-DSR uses the value of 0.9 as a threshold value for switching to local mode.

4.2 Case Study 2: SAMP-DSR performance evaluation

In this case study, we have evaluated the performance of the SAMP-DSR compared to different routing algorithms. The algorithms are evaluated in terms of average end-to-end delay per packet and delivery ratio. We have also compared the amount of routing message. In these set of experiments stop time of 20 seconds have been used for Random Waypoint Mobility model. Figure 6 shows the average ratio of successful packet delivery. MP-DSR and EMP-DSR establish their routes on highest reliable links. The experiments prove that they have higher delivery ratio compared to AODV (Ad hoc On-Demand Distance Vector). EMP-DSR uses global information provided by ants so it's more accurate on finding high reliable end-to-end routes compared to MP-DSR. The SAMP-DSR behaves similar to EMP-DSR on lower mobility modes and to MP-DSR as network becomes more mobile.

Figure 7 shows routing message overhead of the same experiment. When it comes to control message overhead MP-DSR wins clearly. EMP-DSR is always sending high amount of ants periodically through the network so it has always almost constant highest amount of overhead. Instead of broadcasting to all nodes in a node's radio range, MP-DSR uses a threshold for forwarding route request messages (RREQs). SAMP-DSR significantly reduces the amount of ants being sent periodically when network becomes more dynamic avoiding unnecessary overhead.

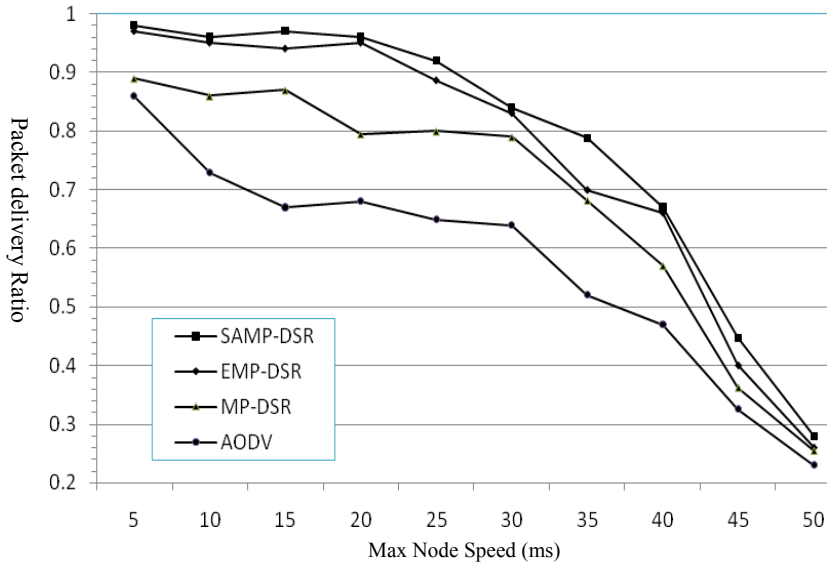

Figure 6 - The average ratio of successful packet delivery

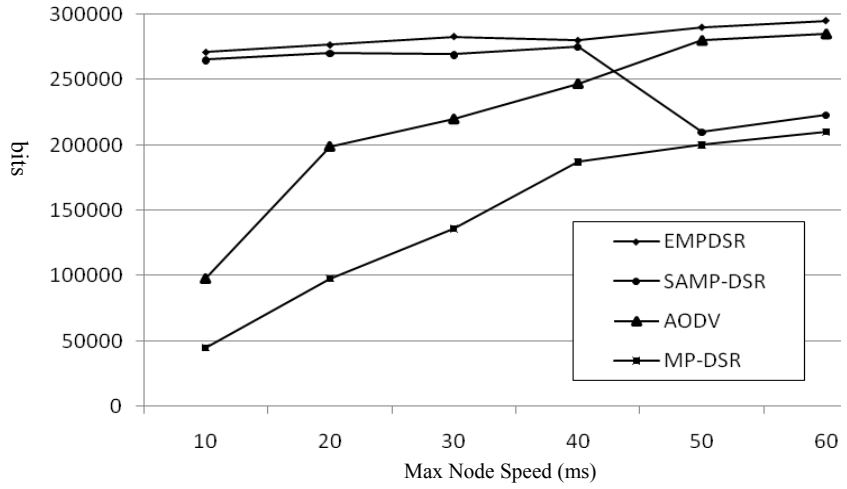

Figure 7 – The routing message overhead

Figure 8 shows the average end-to-end delay of data packets, on lower mobility ACO based algorithms (EMP-DSR and SAMP-DSR) clearly outperform other algo-

rithms since ants have enough time to converge to optimal solutions. However as network mobility increases ACO finds it hard to converge. As discussed before the superiority of ACO based algorithms diminishes in highly dynamic networks.

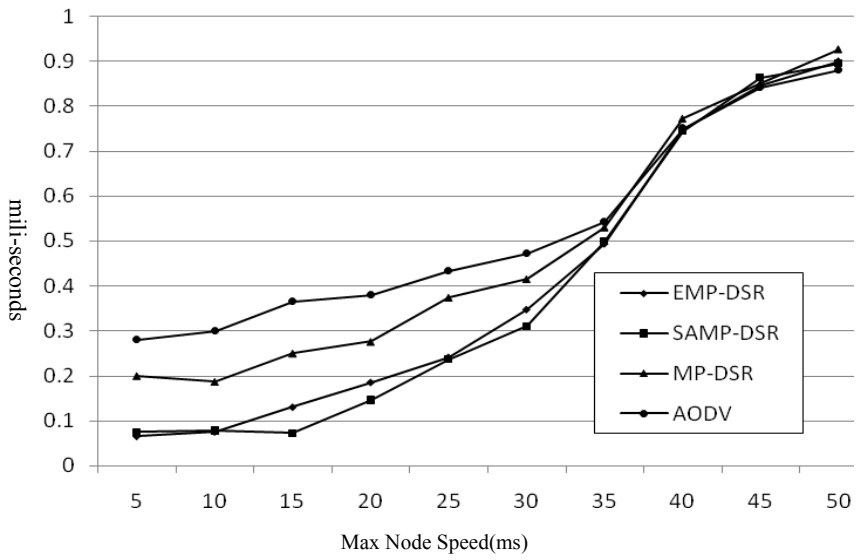

Figure 8 – The average end–to-end delay

5. CONCLUSION

In this case study, we have evaluated the performance of the SAMP-DSR compared to different routing algorithms. The algorithms are evaluated in terms of average end-to-end delay per packet and delivery ratio. We have also compared the amount of routing message. In these set of experiments stop time of 20 seconds have been used for Random Waypoint Mobility model. Figure 6 shows the average ratio of successful packet delivery. MP-DSR and EMP-DSR establish their routes on highest reliable links. The experiments prove that they have higher delivery ratio compared to AODV (Ad hoc On-Demand Distance Vector). EMP-DSR uses global information provided by ants so

it's more accurate on finding high reliable end-to-end routes compared to MP-DSR. The SAMP-DSR behaves similar to EMP-DSR on lower mobility modes and to MP-DSR as network becomes more mobile.

REFERENCES

1. G. D. Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR),* 9:317–365, 1998.
2. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. Artifcial Life, 5(2):137-172, 1999.
3. G.Di Caro, F. Ducatelle and L. M. Gambardella, "AntHocNet: An ant-based hybrid routing algorithm for mobile Ad hoc networks", in Proc. of *PPSN VIII.LNCS. Springer-Verlag*, 2004.
4. M. Dorigo and T. Stutzle. Ant Colony Optimization. *MIT Press*, 2004.
5.] G. D. Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.
6. M. Gunes, U. Sorges, and I. Bouazizi. ARA - the antcolony based routing algorithm for MANETS. *In Proceedings of the ICPP International Workshop on Ad Hoc Networks (IWAHN)*, 2002.
7. J. S. Baras and H. Mehta. A probabilistic emergent routing algorithm for mobile ad hoc networks. In *WiOpt03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
8. S. S. Dhillon and X. Arbona and P. Van Mieghem, Ant Routing in Mobile Ad Hoc Networks. In International Conference on Networking and Services (ICNS '07), 2007.
9. R. Leung, J. Liu, E. Poon, A.-L. C. Chan, and B. Li. MPDSR: A QoS-Aware multi-path dynamic source routing protocol for wireless ad-hoc networks. In *LCN*, pages 132–141. IEEE Computer Society, 2001.
10. E. KhosrowshahiAsl and M. Damanafshan and M. Abbaspour and M. Noorhosseini and K. Shekoufandeh, EMP-DSR: An Enhanced Multi-path Dynamic Source Routing

Algorithm for MANETs Based on Ant Colony Optimization, In *Third Asia International Conference on Modelling & Simulation*, 2009.

11. OMNeT++. http://www.omnetpp.org/, December 2009.
12. Laiquan Han, Jinkuan Wang, and Cuirong Wang, A Concurrent Ant Colony Optimization Multipath Forwarding Algorithm in IP Networks, In *Proceedings of PIERS 2009 in Beijing*, March 23-27, 2009
13. Sanjay K. Dhurandher, Sudip Misra, Harsh Mittal, Anubhav Agarwal and Isaac Woungang, Using Ant-Based Agents for Congestion Control in Ad-Hoc Wireless Sensor Networks", *Cluster Computing (Springer, Netherlands)*. In Press, 2009, ISSN: 1386-7857 (Print) 1573-7543 (Online), http://www.springerlink.com/content/g17245vj1wv55163/, 2009
14. Camilo, T. Carreto, C. Silva, J. S. Boavida, F., An Energy-Efficient Ant-Based Routing Algorithm for Wireless Sensor Networks, In *Lecture Notes in Computer Science, Springer-Verlag,* NUMB 4150, pages 49-59, 2006

Ehsan Khosrowshahi Asl received his M.S. degree in Information Technology from Amirkabir University of Technology in Tehran, Iran, in 2009 and his B.S. degree in Software Engineering from Tehran University, in 2006. His research interests include computer networks, wireless communication, and multimedia networking.

Seyed Majid Noorhosseini received the B.Sc. and M.Sc degrees from Amirkabir university of technology in 1986 and 1989 , respectively. He received his Ph.D degree from McGill University in Montreal ,Canada in 1994. He was a senior

scientist at Nortel Networks in Canada and U.S. during 1996-2005, working in different areas of networks and network management. He has a US patent 6707795 in alarm correlation method in long-haul networks. He is now with the Department of Computer Engineering and Information Technology at Amirkabir University of Technology.

Atieh Saberi received her B.S. degree in hardware engineering from Iran University of Science and Technology, Tehran, Iran, in 2009. Her search interests include computer networks, wireless communications, multimedia networking, and computer graphics.