An Adaptive Multipath Ant Routing algorithm for Mobile Ad Hoc Networks

Sh. Samadi and H. Beigy

***Abstract*—Mobile Ad hoc networks (MANET) are communication networks that consist of wireless nodes, which placed together in an ad hoc manner. Nodes can enter or leave the network at any time, so that the network topology changes frequently. In this paper we are going to propose an algorithm for routing in mobile ad hoc networks. Routing task has some challenging problem in MANETs, based on having dynamic and unplanned nature. For facing with these challenges we propose an Adaptive Multipath Ant Routing algorithm (AMAR) by combining ideas from artificial intelligence (AI) and multipath routing in this algorithm for improving the network performance. In an extensive set of simulation experiments, we compare this algorithm with AntHocNet and AODV and OLSR which are three reference algorithms in this research area.**

***Index Terms*—MANET, ant colony optimization, multipath routing**

I. Introduction

One of the most important developments in recent years in the field of telecommunication networks is the increased use of wireless communication. In this field, we can make a difference between infrastructure networks and infrastructure less networks [1]. In infrastructure networks, a fixed, wired backbone is available, and all communication is directed over this backbone. In infrastructure less networks, such a backbone does not exist, and wireless devices communicate directly with one another through point-to-point connections. They are also referred to as ad hoc networks. Mobile Ad hoc Networks (MANETs) take place in infrastructure less networks. One of the biggest challenging problems in these networks is routing. Routing is the task of constructing and maintaining the paths that connect remote source and destination nodes, while maximizing network performance. This task is particularly hard in MANETs due to issues that result from the particular characteristics of these networks. The first important issue is the fact that MANETs are dynamic networks. The second issue is the unreliability of wireless communication. The third issue is caused by the limited capabilities of the MANET nodes. Finally, the last important issue is the network size. So due to these different challenges, it is important to design algorithms that are adaptive, robust and self-healing. Moreover, they should work in a localized way, due to the lack of central control or infrastructure in the network. Nature's self-organizing systems show precisely these desirable properties. Because of these same properties, they have recently become a source of inspiration for the design of routing algorithms for dynamic networks.

In this paper we are going to propose an Adaptive Multipath Ant Routing algorithm (AMAR). This algorithm is based on a specific self-organizing behavior observed in ant colonies, the shortest paths discovery, and on the related optimization framework of Ant Colony Optimization (ACO) [2], with the use of multipath routing [3].

The rest of this paper is organized as follows. In Section 2 we describe related work. Section 3 contains the description of proposed algorithm and in Section 4 simulation results are presented.

Manuscript received February 9, 2012; revised March 12, 2012.

H. Beigy is with Computer Engineering Department in Sharif University of Technology in Tehran, Iran.

S. Samadi is with Shiraz Azad University and University of Technology, International Campus in Kish Island.

II. Related Litrature

In this section related literature is proposed. In first part we give a short introduction to MANET routing algorithms, in second part basic elements of ACO routing are described. Finally in third part an overview of existing implementations of ACO routing for MANETs are given.

A. Routing in MANETs

Routing task is hard in MANETs, because of dynamic nature of the network so the topology can change constantly, and paths between sources and destinations that were initially efficient can quickly become inefficient. This means that routing information should be updated more regularly than in traditional wired telecommunication networks. However, this can be a problem in MANETs, with their limited bandwidth and node resources, and their possibly unreliable communication channels. In recent years a large number of MANET routing algorithm have been proposed. These algorithms all deal with the dynamic aspects of MANETs in their own way, using reactive or proactive behavior, or a combination of both.

Under proactive routing protocols, all nodes of the network try to maintain consistent routing information about all other nodes at all times. Examples of proactive algorithms is and Optimized Link State Routing (OLSR) [4]. Proactive algorithms have the advantage that routing information is always readily available when data needs to be sent. On the downside, these algorithms can become quite inefficient when a lot of changes need to be tracked (in highly dynamic networks). Under reactive routing protocols, nodes only gather the routing information that is necessary needed. This is the case when a new data session is started, or when a currently used path fails. Example is Dynamic Source Ad-Hoc On-Demand Distance Vector Routing (AODV) [5]. Reactive algorithms can greatly reduce the overhead they create, so that they are in general more

efficient. In MANETs, where all nodes are mobile, there are a lot of topology changes; the general preference is for reactive algorithms. There is another classification in MANETs, which is a combination between both reactive and proactive routing protocol which is called hybrid algorithms. They try to combine the advantages of both approaches. The classic example is the Zone Routing Protocol (ZRP) [6].

B. ACO Routing Algorithms

ACO routing is a class of adaptive routing algorithms that form an alternative to the more traditional approaches to routing. ACO routing was originally inspired by mechanisms found in biology: it is based on principles that are present in the foraging behavior of ants in nature. ACO routing algorithms work in a highly distributed way, and have properties such as adaptivity, robustness and scalability. This makes them particularly interesting to deal with the challenges in MANET routing that were described before.

The basic idea behind ACO algorithms for routing [2] is the acquisition of routing information through sampling of paths using small control packets, which are called ants. The ants are generated simultaneously and independently by the nodes, with the task to test a path to an assigned destination. An ant going from source node *s* to destination node *d* collects information about the quality of the path, and uses this on its way back from *d* to *s* to update the routing information at the intermediate nodes.

C. ACO Routing in MANETs

A wide range of ACO routing algorithms are using for wired networks such as, AntNet [7], and Ant-Based Control (ABC) [8].

Because of having adaptivity and robustness in ACO, MANETs is also attracted to use it. Most of proposed algorithms for these networks are quite similar to algorithms which are proposed for wired networks. The problem that is exist with following design of ACO routing algorithms for wired networks in MANETs is that, the result in proactive routing algorithms is not always the best approach to routing in MANETs .

Accelerated Ants Routing [9] uses ant-like agents which go through the network randomly, without a specific destination, updating pheromone entries pointing to their source. Ant-AODV [10] is a hybrid algorithm combining ants with the basic AODV behavior: a fixed number of ants keep going around the network in a more or less random manner, keeping track of the last n visited nodes and when they arrive at a node they proactively update its routing table. Ant-Colony-Based Routing Algorithm (ARA) [11] works mainly in an on-demand way, with ants setting up multiple paths between source and destination at the start of a data session. During the data session, data packets reinforce the paths they follow. Also Probabilistic Emergent Routing Algorithm (PERA) [12] works in an on-demand way, with ants being broadcast towards the destination (they do not follow pheromone) at the start of a data session. Multiple paths are set up, but only the one with the highest pheromone value is used by data (the other paths are available for backup). AntHocNet [13] which is a hybrid routing algorithm is a strong algorithm, which has improved network performance significantly. AntHocNet is much related to our work presented here.

III. Algorithm Description

Adaptive Multipath Ant Routing algorithm (AMAR) is a hybrid multipath algorithm, designed along the principles of ACO routing and multipath routing. It consists of both reactive and proactive components. The algorithm is has four phases, *Reactive path setup, Stochastic data routing,* Proactive *path maintenance and exploration and Link failures.* A 2-dimensional Euclidean space is assumed for the node area. In what fallow, first the data structures that are maintained in each node is described then detailed description of the different components of this routing algorithm is given.

A. Data Structures

Each node has three tables, pheromone table, neighbor table and path table. Pheromone table $\mathcal{T}_i$ which is a two dimensional matrix, contains information about the route from node *i* to destination *d* over neighbor *j* (T_{ji}^{d}). The regular pheromone, the virtual pheromone, and the average number of hops are included in the pheromone table's information. The regular pheromone value τ_{ij}^{d} is an estimate of the goodness of the route from *i* to *d* over *j*. Goodness is defined as the inverse of the cost. Regular pheromone and the average number of hops are updated by backward ants. The virtual pheromone value ω_{ij}^{d} forms an alternative estimate of the goodness of the route from node *i* to node *d* over node *j*. This value is obtained through information bootstrapping using goodness values reported by neighbor nodes during the proactive route maintenance process. Each node also maintains a neighbor table. The neighbor table N_i kept by node *i* is a one-dimensional array with one entry for each neighbor. All possible paths from any source node *i* to any destination *d*, with a prediction about the stability of each path, are saved in path table.

Besides all nodes are equipped with a location aware chip or they can compute their location by means of a localization algorithm [14]

B. Reactive Path Setup

When a source node *s* starts a communication session with a destination node *d*; and it does not have any routing information available for d, it broadcasts a reactive forward ant [13]. Due to this initial broadcasting, each neighbor of *s* receives a copy of forwarded ant. Each ant has a job to find a path connecting *s* to *d*. At each node, an ant is either unicased or broadcasted, according to whether or not the current node has routing information in its pheromone table T_i for *d*. Next hop (*n)* is chosen with probability P_{nd} as shown in equation 1.

$$P_{nd} = \frac{\left(T_{nd}^{i}\right)^{\beta 1}}{\sum_{j \in N_d^i} (T_{jd}^{i})^{\beta 1}}, \quad \beta_1 \geq 1, \tag{1}$$

where N_d^i is the set of neighbors of *i* over which a path to *d* is known, and β_1 is a parameter value which can lower the exploratory behavior of the ants. In case the intermediate node *i* does not have routing information for *d*, it broadcasts the reactive forward ant. Due to this broadcasting, a reactive

forward ant can proliferate quickly over the network, with different copies of the ant following different paths to the destination.

Each forward ant keeps a list ρ of the nodes [1,…, n] it visited. At destination *d*, the reactive forward ant is converted into a reactive backward ant, which follows the list ρ, on its way back to *s*.

At each intermediate node *i*, coming from neighbor *n*, the ant updates the entry τ_{ij}^{d} in its pheromone table as shown in equation 2. The way an entry is updated depends on the path quality metrics [15] used to define the pheromone variables (cost values):

$$\tau_{ij}^{d} = \gamma\tau_{ij}^{d} + (1-\gamma)(c_i^d)^{-1},\ \gamma \in [0,1] \qquad (2)$$

where, γ is a parameter regulating the speed of adaptation of the pheromone to new cost values. The cost value c_i^d is inverted to calculate the pheromone value τ_{ij}^{d}, as pheromone indicates the goodness of a route, rather than its cost.
Reactive backward ant also used to updates the number of hops ($h^d{}_{in}$) *i*'s pheromone table [15].

C. Stochastic Data Routing

Data routing and proactive path maintenance and exploration phase work synchronized. Consider that the volume *D* of data packets needs to be routed from source *s* to destination *d*. if *D* is large; usually it is impossible to send data in a single path in a mobile environment. As our aim in this research is to have multipath routing in our network, we divide the total data volume *D* into comparatively smaller data volume packets $D_1, D_2, \ldots$ called temporal data sets [3]. Since at this time we have only one available path, which had been constructed in reactive route setup, for keeping the network busy, the first temporal data set is going to be routed from *s* to *d* from that route. For routing other temporal data sets we have to wait till the proactive path maintenance and exploration phase build a full mesh of multiple routes around the initial route created during the reactive route setup process [13] (we refer to the description of proactive phase in the next subsection). Then base on stability of each path, the network choose more stable ones, and start to route other temporal data sets from them. Node *i*, calculate the estimated time needed for routing each data set from saving the time it send a reactive forward ant and receiving a reactive backward ant. Also each node calculate the local estimate

Data packets are forwarded from their source to their destination in hop-by-hop fashion, taking a new routing decision at each intermediate node. Routing decisions for data packets are based only on regular pheromone.

Nodes forward data stochastically. When a node has multiple next hops for the destination *d*, it will randomly select one of them; with probability P_{nd}. P_{nd} is calculated in the same way as for the reactive forward ants [16] as shown in equation 3, but with a higher exponent, in order to be greedier to choose paths with high quality:

$$P_{nd} = \frac{\left(T_{nd}^{i}\right)^{\beta 2}}{\sum_{j \in N_d^i} (T_{jd}^{i})^{\beta 2}}, \quad \beta_2 \geq \beta_1, \qquad (3)$$

The probabilistic routing strategy leads to data load spreading according to the estimated quality of the paths. If the estimates are kept up-to-date, this leads to automatic load balancing.

D. Proactive path Maintenance and Exploration

The proactive route maintenance process used to update and extend available routing information. It allows building a mesh of multiple routes around the initial route created during the reactive route setup process. It consists of two subprocesses: pheromone diffusion and proactive ant sampling [15].

Pheromone diffusion is aimed at spreading available pheromone information over the network through the use of periodic update messages called hello messages and information bootstrapping. Hello messages are short messages broadcasted every t_{hello} seconds asynchronously by all the nodes of the network in their whole lifetime. These messages used to allow nodes to find out which are their immediate neighbors, and to detect link failures.

Hello messages are also used to carry information in the pheromone diffusion process and carrying the information about the position of each node *i* to its neighbors.

Each node put *K* number of destination *d* it has routing information, in its hello message. For each one of these destinations *d*, the hello message contains the address of *d*, the best pheromone value that *i* has available for *d*,v_i^d, and a bit flag. This best pheromone value is taken over all possible values for regular pheromone and virtual pheromone associated with *d* in *i*'s pheromone table. The bit flag is used to indicate whether the reported value was originally regular or virtual pheromone. A neighboring node *j* receiving the hello message from *i* goes through the list of reported destinations. For each listed destination *d*, it derives from the hello message an estimate of the goodness of going from *j* to *d* over *i*, by applying information bootstrapping [15]: it combines the reported pheromone valuev_i^d, with the locally maintained estimate of the cost c_j^i of hopping from *j* to *i*. The result of the calculation (equation 4) is called as bootstrapped pheromone value ($k^d{}_{ji}$).

$$K_{ji}^{d} = \left(\left(v_i^d\right)^{-1} + c_j^i\right)^{-1} \qquad (4)$$

Bootstrapped pheromone value is gain from a cheap procedure, but with low reliability. SinceK_{ji}^{d} is derived from the estimate v_i^d reported by *i*, it is only correct as long as v_i^d is correct. This can be problematic in a highly dynamic environment like MANETs, where routing information can get out of date quickly, and especially if the value v_i^d reported by *i* was in itself the product of pheromone diffusion. So K_{ji}^{d} is only used for updating the virtual pheromone value in *j*'s pheromone table. This way, the pheromone obtained via the pheromone diffusion process is kept separate from the regular pheromone, which is the product of ant based route sampling and is therefore considered more reliable and data packets only consider regular pheromone when choosing a next hop. Virtual pheromone is used for forwarding proactive forward ants towards their destination. One could say that the potentially unreliable bootstrapped pheromone provides hints about possible routes, which are then explored and verified by the proactive forward ants.

The proactive ant sampling process is started by the

source node of a session at the moment the first data packet of a new session is received, and continues for as long as the session is going on [15]. These ants can follow regular pheromone, or virtual pheromone. While the former leads the ants to update goodness estimates of existing routes, the latter allows them to find new routes based on the hints provided by the pheromone diffusion process. This way, the single route that was initially constructed in the reactive route setup process is extended to a full mesh of multiple paths [15]. The proactive forward ant takes a new routing decisions at each intermediate node *i*, as shown in equation 5.

$$P_{in}^{d} = \frac{\left[\max\left(\tau_{in}^{d}, \omega_{in}^{d}\right)\right]^{\beta_2}}{\sum_{j \in N_i^d}\left[\max\left(\tau_{ij}^{d}, \omega_{ij}^{d}\right)\right]^{\beta_2}}, \quad \beta_2 \geq 1, \tag{5}$$

where τ_{in}^{d} is regular pheromone, ω_{in}^{d} , shows virtual pheromone and N_d^i is the set of neighbors.

As we can see unlike reactive forward ants, proactive forward ants rely both on regular and virtual pheromone for their routing decisions: they use the maximum between regular and virtual pheromone to calculate the probability of each next hop. Also different from reactive forward ants is that proactive forward ants are never broadcast: when they arrive at a node that does not have any routing information for their destination, they are discarded.

When a proactive forward ant arrives at its destination, it is converted into a proactive backward ant, which is sent back to the source. Proactive backward ants have the same behavior as reactive backward ants. An important aspect to note here is that while the proactive forward ants can follow both regular and virtual pheromone, proactive backward ants always deposit regular pheromone. This way, the proactive ant sampling process can investigate promising virtual pheromone, and if the investigation is successful turn it into a regular route that can be used for data. The Proactive ant sampling is used for another purpose in this algorithm. As we mention before, each hello message coming from node *i*, also contains the information about the node *i* position, and send this information to all its neighbors. A proactive backward ant is going to calculate and save the *affinity* of all the links which the proactive forward ant had been visited, on its way back to source. When the proactive backward ant reaches to source, the source can determine the stability of this path from reported affinity values. *Affinity* a_{mn} (affinity of the link which connect node *n* to node *m*), associated with a link l_{mn} (link from node *m* to node *n*), is a prediction about the span of life of the link l_{mn}, in a particular context. So the stability of the connectivity between *m* and *n* depends on a_{mn}. To find the affinity a_{mn}, we should calculate the distance between node *m* and *n*, by using the information that each hello message carry, about the position of each node. Since node *m* and *n* are neighbors, they are aware of each other position (by receiving the hello messages from each other every t_{hello} seconds). So node *m* can calculate d_{mn} by using eucleadian distance function (equation 6).

$$d_{mn} = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2} \tag{6}$$

where (x_1, x_2) shows node *m* position and (y_1, y_2) shows node *n* position. If *M* is the average velocity of the nodes, the average worst-case affinity a_{mn} at time *t* is calculated as fallow (equation 7):

$$\text{the average worst} - \text{case affinity } a_{mn} = \frac{(\mathrm{R_n} - \mathrm{d_{mn}})}{\mathrm{M}} \tag{7}$$

where R_n is the transmission range of *n.* assuming at time *t*, the node *m* has started moving outwards with an average velocity *M*. Given any path *p* from any node *i* to another node *m* as *p* = (*i, j, k, ..., 1, m*), the stability of path *p* will be determined by the lowest affinity link defined as: min [a_{ij}, a_{jk}, …, a_{lm}]. As mentioned before, more stable paths are selected for routing temporal data sets [3].

E. Link Failures

In MANETs, link failures can occur due to physical changes such as the movement or disappearance of a node, or due to changes that influence the connectivity of the wireless communication. Since MANETs are usually highly dynamic, such events are expected to occur frequently, and MANET routing algorithms should be prepared to deal with them effectively.

The first step in dealing with link failures is their detection. Unicasting warning messages to all neighbors to notify them about the link failure is the next step. Next step is to starts a local route repair process to try to repair the route to *d*, so that the data packet can still be delivered. A final aspect of dealing with link failures in AntHocNet is the use of unicast warning messages. These are emergency messages that are needed when link failure notification messages are not delivered correctly [15].

TABLE I: BASE SCENARIO PROPERTIES

Parameter	Value
Number of nodes	100
Network size	2400*800m^2
Mobility model	Random WayPoint
minimum speed	0m/s
Maximum speed	10*m/s*
Pause time	30s
experiment duration	900*s*
Experiment repeated	20 times
CBR	20 data sessions
Session duration	180s
session generation	4 packets of 64 bytes/s
MAC layer protocol	IEEE 802.11 DCF
Transport layer protocol	UDP
physical layer protocol	IEEE 802.11
transmission rate	2*Mbit/s*
radio range	250*m*

IV. ANALYSIS AND SIMULATION RESULTS

We are going to compare AMAR with AntHocNet [15], which is a state-of-the-art routing algorithm for MANETs, also AODV [7] which is a purely reactive algorithm, and OLSR [6] which is purely proactive routing algorithm.
The performance of AMAR in a range of different scenarios is investigated using QualNet simulator. In order to do tests in a controlled way, we define a common base scenario, from which all other scenarios are derived by varying parameters such as the node speed, the data send rate, the network size, etc.. Table I shows base scenario properties.

For evaluation measures, we can make distinguish between measures of effectiveness and efficiency. Measures of effectiveness are external measures of performance, such as the data delivery ratio and end-to-end packet delay.

Measures of efficiency are internal evaluation measures, such as the overhead in number of packets and the overhead in number of bytes.

Now we can vary a different environmental property, in order to investigate its effect on the performance of the algorithms independently. Fig. 1 shows the end-to-end packet delay when the node speed varies from 1*m/s* to 30*m/s*. For all algorithms the delivery ratio decreases as node speed goes up. Best results are obtained by AntHocNet and AMAR. This behavior shows that these two algorithms are able to adapt well to the fast changes in the highly dynamic environments. AMAR used more stable paths, so it is able to deliver more data sets from source to destination even in highly dynamic network. AODV gives less good results. As we can see, by increasing node's speed, the difference in performance between AODV and AntHocNet and AMAR decrease. The eventual decrease in the difference between AMAR or AntHocNet and AODV shows that there is a limit to the adaptivity in AntHocNet and AMAR. At the highest levels of mobility, the proactive mechanisms of AntHocNet and AMAR get more difficulties keeping up with the changes in the network. OLSR shows the worst results which confirms that proactive algorithms aren't adaptive in highly dynamic environments.

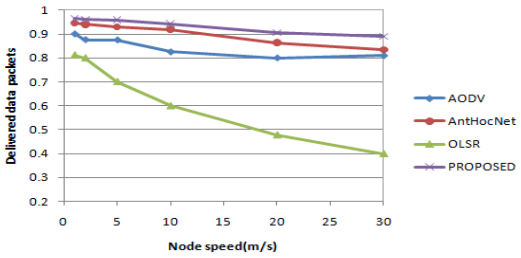

Fig. 1. Delivery ratio under increasing node speed

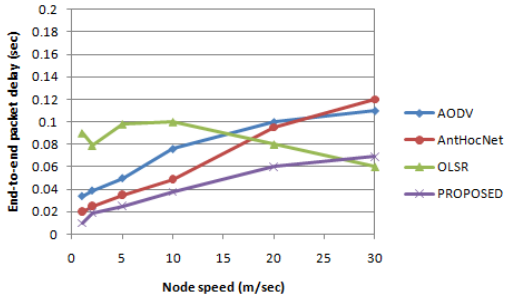

Fig. 2. End-to-end delay under increasing node speed

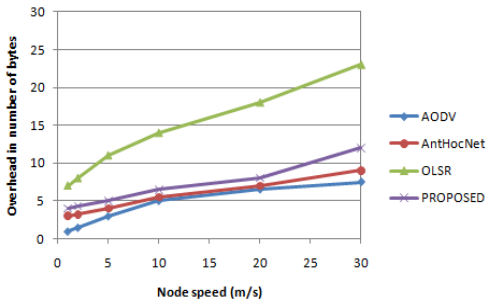

Fig. 3. Overhead in number of bytes under increasing node speed

The results for average delay show similar results with some changes (Fig. 2); in the highest speed scenarios, OLSR gives good performance. It is probably in the situation where it delivers only a very low percentage of the packets.

In the results for the overhead measures, there is a difference between the overhead in number of packets and the overhead in number of bytes. When considering the number of packets, we obtain the same order as before. When considering number of bytes, the ACO algorithms sufferer a bit more.

AMAR became worse that AntHocNet and AnHocNet becoming slightly worse than AODV. This indicates that our proposed algorithm and AntHocNet use considerably larger control packets than AODV and OLSR. One reason for this is that ants gather information about the full path that they have followed. In the case of AntHocNet, an obvious other cause of large control packets is the piggybacking of routing information in hello messages. AMAR also suffer from this large control packets which caused by putting the node position information on top of hello messages. On the other hand using large control packets can actually be an advantage for link failure detection [15].

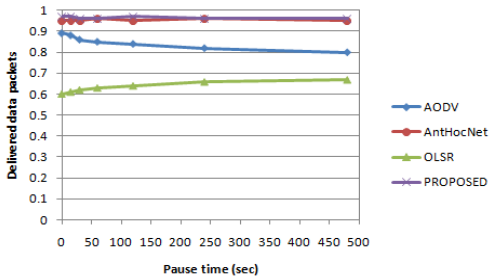

Fig. 4. Delivery ratio under increasing pause time

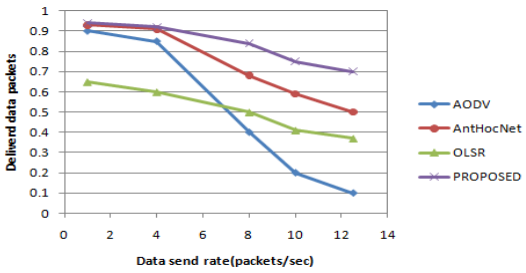

Fig. 5. Delivery ratio under increasing data send rate

In next test, to change node mobility, we vary the pause time for RWP mobility model (Fig. 4). Increasing the pause time has two effects on the general properties of the scenario: First, decrease in node mobility (nodes stay still for longer time). Second, having low node density in the center of the network area (nodes are more spread out, giving a lower effective node density to the network). Hence, increasing the pause time can both decrease and increase the difficulty of the scenario for routing, depending on whether the used routing algorithm is more sensitive to high mobility or to low node density. In test results for the delivery ratio (Fig. 4), AMAR and AntHocNet show the best performance. They are almost insensitive to the increase in pause time. AODV shows a decrease in delivery ratio for higher pause times. This is an indication that it is more sensitive to the decrease in connectivity than to the decrease in mobility. Finally, OLSR shows an opposite trend.

Finally for changing data traffic, the data send rate varies from 1*packet/s* to 12.5*packet/s* (Fig. 5). By increasing the data rate, the network loads higher, so that congestion and interference become more likely. In terms of delivery ratio,

all algorithms decreasing network performance. As was expected AMAR shows the best results because it use more stable paths for sending data sets in multipath. AntHocNet comes right after AMAR. As we can see, the performance drops very suddenly in our AODV. The increase of data load increases the congestion, which lead to packet loss. AODV face with this packet loss as an indication of a link failure, and reacts to it with a route repair or a new route setup. This reaction in turn strongly increases the load in the network. The hybrids algorithms face with this event by rely on their proactive mechanism and avoid the need to execute a route setup process, use their other multipath routes available. We can mention here that AMAR leads the network to have much less packet loss, compare with other proposed algorithms. OLSR which is a purely proactive routing algorithm is almost insensitive to this kind of interactions.

Network throughput also improved in AMAR, base on its multipath nature.

V. Conclusion and Future Work

In this paper, we proposed an Adaptive Multipath Ant Routing algorithm that combines ideas from ACO and multipath routing for finding the most stable paths in the network between any source and destination. In order to evaluate the performance of the proposed algorithm, we compare it to ANntHocNet, OLSR and AODV. Through the simulation experiments, we showed that the proposed algorithm improves the performance of the network by decreasing end-to-end delay and increasing packet delivery. The algorithm could be modified to take into account some aspects that have not been addressed in this work, and that can be interesting subject of future research. Reducing of the overhead on the network can be considered as a future work. This reduction can be obtained by improving the performance in both reactive and proactive phases.

References

[1] E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999.

[2] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5(2):137{172, 1999.

[3] S. K.Das, A. Mukherjee,, S. Bandyopadhyay, K. Paul, D. Saha: Improving Quality-of-Service in Ad hoc Wireless Networks with Adaptive Multi-path Routing. *IEEE Global Telecommunications Conference*. 2000.

[4] T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, A. Qayyum, and L. Viennot. Optimized link state routing protocol. In *Proc. of IEEE INMIC*, 2001.

[5] C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[6] J. Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proc. of the IEEE International Conference on Universal Personal Communications*, 1997.

[7] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (*JAIR), 9:317{365, 1998.

[8] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169{207, 1996.

[9] K. Fujita, A. Saito, T. Matsui, and H. Matsuo. An adaptive ant-based routing algorithm used routing history in dynamic networks. In *Proc. of the 4th Asia-Pacific Conf. on Simulated Evolution and Learning*, 2002.

[10] S. Marwaha, C. K. Tham, and D. Srinivasan. Mobile agents based routing protocol for mobile ad hoc networks. In P*roc. of IEEE Globecom*, 2002.

[11] M. G unes, U. Sorges, and I. Bouazizi. ARA - The ant-colony based routing algorithm for MANETs. In *Proc. of the ICPP International Workshop on Ad Hoc Networks (IWAHN)*, 2002.

[12] 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.

[13] G. Di Caro, F. Ducatelle, and L. M. Gambardella. AntHocNet: an adap-tive nature-inspired algorithm for routing in mobile ad hoc networks. *European* Transactions *on Telecommunications (ETT)*, 16(5), 2005.

[14] A. Pal, "Localization Algorithms in Wireless Sensor Networks: Current approaches and future challenges," Network protocols and algorithms, vol. 2, 2010, pp. 45-73.

[15] F. Ducatelle, "Adaptive Routing in Ad Hoc Wireless Multi-hop Networks," Ph.D. dissertation, *Universit µa della Svizzera italia*na, 2007.

Hamid Beigy was born in Shiraz. He received his Bachelor of Computer Engineering from Shiraz University in Shiraz in 1992. He also received his Master of Science degree in Computer Engineering from the same University in Shiraz; in 1995, and he got his Ph. D in Computer Engineering from Amirkabir University of Technology in Tehran, Iran, in 2003. He is an Associate Professor in Computer Engineering Department in Sharif University of Technology in Tehran, Iran. Dr. Beigy has published many Journal papers, Conference papers, Book Chapters and Technical reports. His research interests are in Learning Systems, Computer & Mobile Networks, Parallel Algorithms and Soft Computing

Shahrooz Samadi was born in Shiraz, Iran in 1983. She received her Bachelor of Computer Engineering degree from Shiraz Azad University and her Master of Science degree in Information Technology in Sharif University of Technology, International Campus in Kish Island. This paper has published based on her thesis in partial fulfillment of the Requirements for the Degree of Master of Science. Her research interests are in Networking, Peer to Peer Networks, Ad Hoc Mobile Networks and Swarm intelligence.

.