DYNAMIC ADAPTIVE ROUTING IN MANETs [(1)]

John S. Baras and Harsh Mehta

Electrical and Computer Engineering Department
and the Institute for Systems Research
University of Maryland College Park
College Park, MD 20742

ABSTRACT

Mobile ad hoc networks are infrastructure-less networks consisting of wireless, possibly mobile nodes that 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. The Swarm Intelligence paradigm has recently been used in solving the routing problem in static computer networks with encouraging results. These algorithms have been proven to be robust and resilient to topology changes. In this paper we present performance results on a new routing algorithm for MANETs based on the swarm intelligence paradigm. We present simulation results that measure the performance of our algorithm with respect to the characteristics of a MANET, the varying parameters of the algorithm itself as well as performance comparison with other well-known MANET routing protocols.

INTRODUCTION

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, [9], [11], [12], [13]. These protocols can generally be categorized as either *proactive* or *reactive* protocols. Proactive protocols build routes in the network constantly, while reactive (on-demand) protocols attempt to establish multi-hop routes 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. 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]. 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. Interest in applications of ant-based routing in MANETs has risen and several papers have appeared recently on the subject [14], [15], [16]. For instance, Gunes *et al.* have proposed an Ant-based approach to routing in MANETs in [15]. Their approach uses ants for building routes initially and hence is a completely reactive algorithm. Marwaha *et al.* [16] have explored a hybrid approach using both AODV and Ant-based exploration.

In this paper we describe a new algorithm that utilizes the inherent broadcast nature of wireless networks to multicast control and signaling packets (ants). This algorithm competes well with AODV and we show here several comparisons by simulations in a standard benchmark for MANETs [10], [13], [18]. This algorithm also allows for 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 [17].

THE PROBABILISTIC EMERGENT ROUTING ALGORITHM

In this section, we propose the probabilistic emergent routing algorithm (PERA) based on the Swarm Intelligence paradigm. In this approach, the process of route discovery is carried out by using a flooding approach to discover and maintain multiple 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. 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.

---

(1) Prepared through collaborative participation in the Collaborative Technology Alliance for Communications & Networks sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0011.

A. Bootstrapping the Routing Tables

In our algorithm, neighbor discovery is done using 'HELLO' broadcast messages. The routing table entry for a destination is initialized at a node only after receiving a backward ant from the destination. 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, 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, $T_{n\rightarrow d}$ is assigned to the mean μ_{nd} , and the variance, σ_{nd}^2 is assigned a value of zero. Modifications to $(\mu_{nd}, \sigma_{nd}^2)$ are made upon the arrival of later backward ants based on the learning rule as discussed in the section on backward ants. On the other hand, if the metric under consideration is the hop count, 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 1.

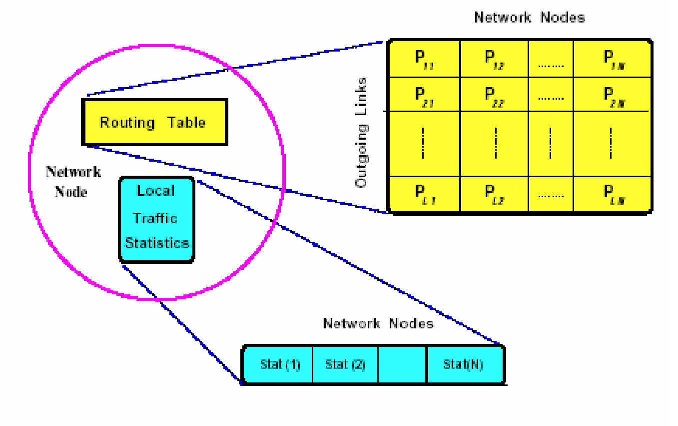

Fig. 1. The routing table and statistics at each node

B. Forward Ants

To carry out the process of Route Discovery, forward ants or agents are sent to a destination beginning at the time at which a data packet for the destination is first received by the node. 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). 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). Duplicate ants and ants that loop back to a node are destroyed.

If the metric under consideration is delay, 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. 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 the source address, the hop count and the stack from the forward ant. The forward ant is then deallocated. It is important to note that since 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*. Further, the forward ant also collects information about each of these paths.

C. Backward Ants

When a forward ant reaches the destination node that it is intended for, the destination node creates a backward ant that 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. The backward ant is similar to the forward ant but has a non-unique sequence number. 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 backward node makes changes to the probability values at the intermediate and final (source for forward ant) node according to the following update rules:

1) $$P_{fd} \leftarrow (P_{fd} + r)/(1+r)$$
$$P_{nd} \leftarrow P_{nd}/(1+r)$$

2) $$P_{nd} \leftarrow P_{nd} - rP_{nd}$$
$$P_{fd} \leftarrow P_{fd} + r(1 - P_{fd})$$

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. Here, ***r = k / f(c) ,*** where ***k > 0*** and ***f(c)*** is the cost function**.**

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 are updated using the following rules:

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

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

D. Routing Data Packets

The data packets can be routed based on the highest probability next hop neighbor or probabilistically. Previous results [3] for swarm intelligence algorithms show excellent results for this method in the case of static networks with relatively small topologies.

SIMULATION RESULTS

Network Simulator 2 [18] discrete event simulator was used to simulate our algorithm. At the physical layer, radio propagation distance for each node was set to 250*m* and the channel capacity was 2 *Mbps*. Our model does not support radio capture [13] so, in the case of packet collisions all packets are dropped. The IEEE 802.11 Distributed Coordination Function (DCF) [10] as implemented in NS2 was used as the Medium Access Control (MAC) protocol. The communication medium is broadcast and nodes have bi-directional connectivity. Each simulation was run for 900 *seconds*. Multiple runs with different seed values were conducted for each scenario and the collected data were averaged over those runs. The algorithm was developed as a separate NS2 routing layer protocol. The mobility model used was the *Random Waypoint model*. As performance metrics we used 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.

$$Goodput = \frac{Data\ packets\ received\ at\ routers}{Total\ packets\ received\ at\ routers} * 100$$

$$Throughput = \frac{Data\ packets\ recvd\ at\ destints}{Data\ packets\ sent\ from\ sources} * 100$$

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. We evaluated the performance of the routing algorithm based on the hop count metric. The network consisted of 20 nodes, randomly placed in an area 500*m* x 500*m*. 4 source and destination pairs were randomly chosen from these 20 nodes. Each source transmitted 1 *packet*/*sec*. Nodes in the simulation were mobile.

A. Mobility Speed

In these experiments, the mobility speed was varied between 0 to 20*m/s* (0, 5, 10, 15, 20). Figure 2 shows the goodput as a function of the node mobility speed. It is seen that the goodput decreases with increase in mobility.

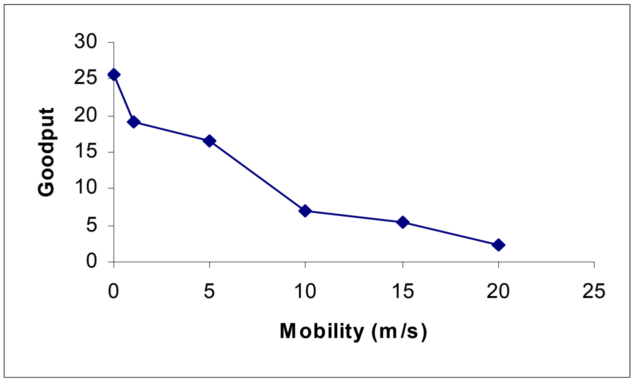

Fig. 2. Goodput *vs*. mobility; 20 nodes in 500*m* x 500*m*

COMPARISON WITH AODV

We compared the proposed algorithm with AODV [12], [13] in terms of throughput, delay and goodput.

A. Goodput Comparison

Figure 3 shows a comparison of the goodput for AODV and PERA for a scenario with 20 nodes in an area of 500*m* x 500*m* 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.

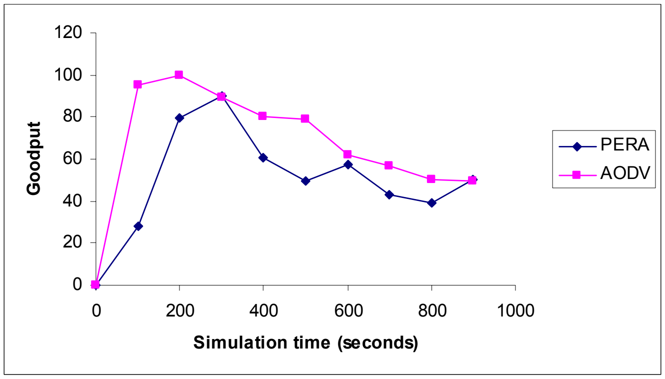

Fig. 3. Goodput comparison of AODV and PERA at 1*m*/*s*

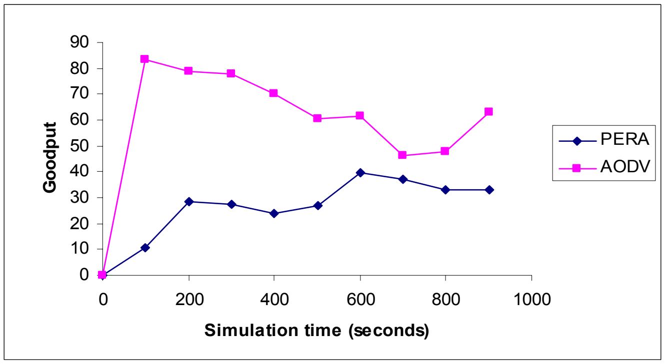

Fig. 4. Goodput comparison of AODV and PERA at 10*m*/*s*

Figure 4 shows a comparison of PERA and AODV for the same scenario as above, but with a mobility speed of 10m/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.

B. Throughput Comparison

Figures 5 and 6 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.

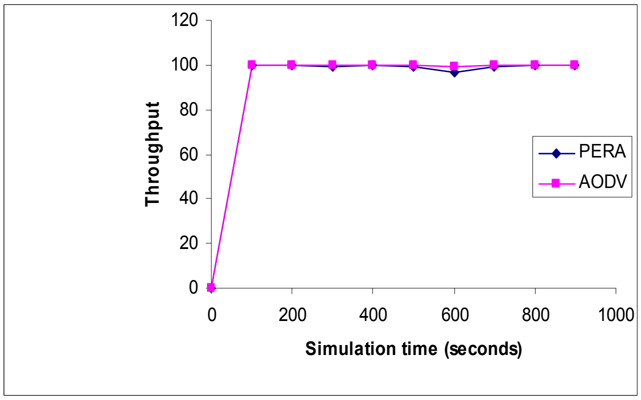

Fig. 5. Throughput comparison AODV vs. PERA, 10*m*/*s*

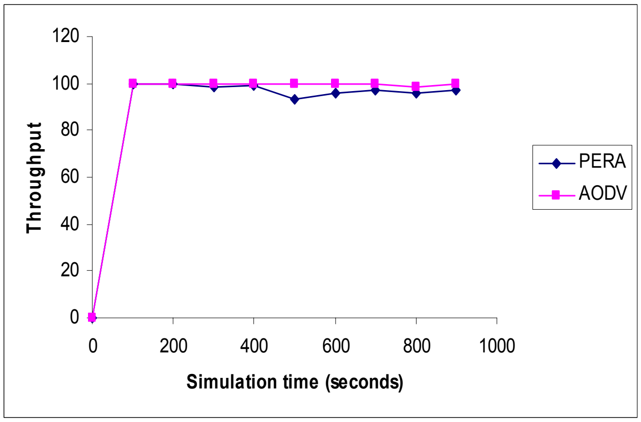

Fig. 6. Throughput comparison AODV vs. PERA, 10*m*/*s*

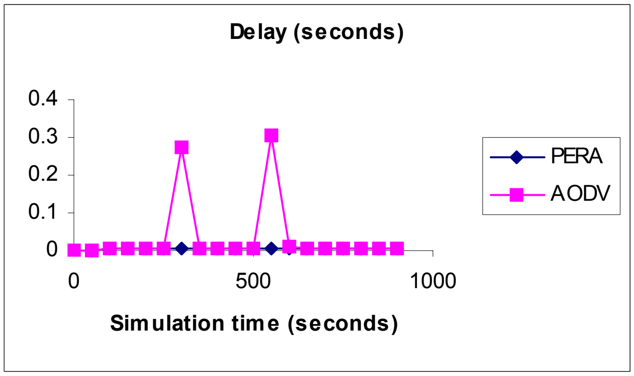

Fig. 7. Delay comparison AODV vs. PERA, 1 *m*/*s*

C. Delay Comparison

Figures 7 and 8 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 is found, PERA delivers the data packet through an alternate route.

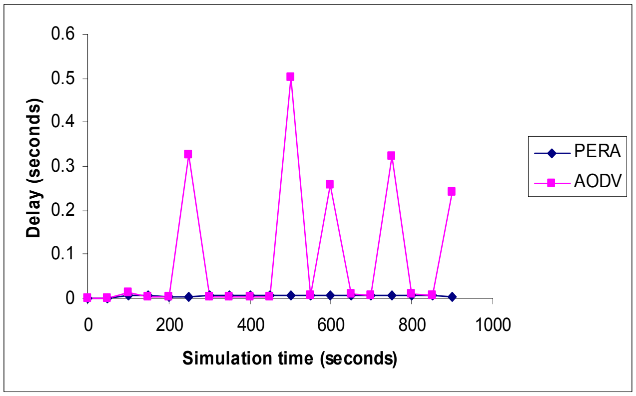

Fig. 8. Delay comparison AODV vs. PERA, 10*m*/*s*

CONCLUSIONS

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.

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.*

[3] G. DiCaro and M. Dorigo, “AntNet: A Mobile Agents Approach to Adaptive Routing”, *Technical Report IRIDIA/97-12.*

[4] G. DiCaro and M. Dorigo, “AntNet: Distributed Stigmergetic Control for Communication Networks”, *J. Artif. Intel. Res.*, V.9, pp.317-365, 1998.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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, Norwell, Mass., 1996, pp. 153-181.

[10] IEEE Computer Society LAN MAN Standards Committee, *Wireless LAN Medium Access Protocol (MAC) and Physical Layer (PHY) Specification*, IEEE Std 802.11-1997.

[11] J. Broch, D. Maltz, D. Johnson, Y. Hu and J. Jetcheva, “A Performance Comparison of MultiHop Wireless Adhoc Network Routing Protocols”, Carnegie Mellon MONARCH Project, Oct. 1998.

[12] 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.

[13] C.E.Perkins (Edt.), *AD HOC NETWORKING*, Addison Wesley, 2001.

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

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

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

[17] H. Mehta, *Dynamic Adaptive Routing in Mobile Ad Hoc Networks*, M.S. Thesis, Electrical and Computer Engineering Department, University of Maryland, December 2002, available as a Technical Report from the Institute for Systems Research.

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

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.