Ant Routing in Mobile Ad Hoc Networks

S. S. Dhillon, X. Arbona and P. Van Mieghem
Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
P.O. Box 5031, 2600 GA, Delft, The Netherlands
s.dhillon, P.VanMieghem@ewi.tudelft.nl and f.j.arbonabernat@student.tudelft.nl

Abstract

We study the performance of ant routing for static and dynamic network topologies. We also compare the performance of ant routing with AODV and DSR for ad hoc networks. The simulations show that the ant routing algorithm performs well for static topologies. However, the routing overhead in ant routing is more than AODV and DSR. Therefore, ant routing performs worse than AODV and DSR in ad hoc networks because of the limited capacity and restrictions on the buffer size. Moreover, with mobility, a large number of data packets in ant routing experience loops that degrade the end-to-end performance.

1. Introduction

Routing consists of a routing protocol and a routing algorithm [15]. The routing protocol is used to exchange information about topology and link weights while the routing algorithm computes the paths between nodes. An example of the classical approach to routing is the Open Shortest Path First (OSPF). In OSPF, the routers exchange link-state information by flooding and the link state updates are generated only when the link status changes. Once a node has obtained topology information of the entire network, Dijkstra's and Bellman-Ford's algorithms are generally used to compute the shortest path [15].

A relatively new approach to routing is the *mobile agent based routing* (MABR) or ant routing [5, 4] which combines the routing protocol and the routing algorithm into a single entity. In MABR, the nodes maintain probabilistic routing tables which are updated periodically by mobile agents (ants) based on the quality of paths. The quality of paths is expressed in terms of metrics such as hopcount, end-to-end delay, packet loss etc. The probabilistic routing tables contain the probability of choosing a neighbor as the next hop for any destination. Data packets are forwarded according to the probabilistic routing tables, and therefore, no additional routing algorithm is required to compute the paths.

There are a number of differences between the above two approaches. MABR is a hop-by-hop routing approach. In MABR, traffic across the network is continuously monitored by the mobile agents. Due to the probabilistic routing in MABR, the paths are not loop-free and oscillations might occur. The hop-by-hop nature of routing in MABR cannot reserve paths that satisfy QoS constraints [15]. However, MABR provides soft-QoS since different paths are continuously monitored and quality of paths is reflected in the routing table values. In OSPF, traffic fluctuations and the end-to-end delay are not measured. MABR has been shown to perform load balancing, i.e. distribute traffic along multiple paths, and automatic adaptation to node or link failure in static topologies [5, 14, 4].

In ad hoc and mesh networks [1], mobile nodes communicate with each other using multihop wireless links. A key challenge in these networks is the development of routing protocols that can efficiently find routes between communicating nodes. In addition, in ad hoc networks the routing protocol must be able to cope with the node mobility that changes the network topology frequently. We refer to the frequently changing network topology as dynamic topology. Routing protocols for ad-hoc networks can be classified into different categories such as Pro-active or table driven (e.g. DSDV [3]) and Reactive or on-demand (e.g. AODV [11] and DSR [9]) based on the routing information stored at the nodes. In on-demand protocols, routes are created only when desired by the source node. In table-driven routing protocols, each node maintains one or more tables to store routing information and attempts to maintain consistent, up-to-date view of the network topology. In general, on-demand protocols perform better than reactive protocols in ad hoc networks [3] and mesh networks [1].

The performance of MABR for mobile ad hoc networks is an open issue. MABR algorithms do not take into account the mobility of the nodes. Moreover, load balancing for frequently changing topologies is a challenging issue.

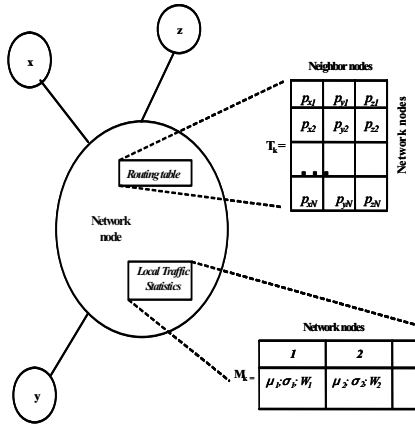
Load balancing involves distributing traffic along multiple paths depending on the traffic conditions such that the congested nodes or the unavailable links are by-passed. With node mobility, the paths are not stable making load balancing difficult. Furthermore, characteristics of traffic in ad hoc networks is not known. In this paper, our aim is to study whether MABR algorithms can be applied for routing in mobile ad hoc networks.

We study the performance of a MABR algorithm (W_AntNet) described in Section 3, for static and mobile wireless networks. W_AntNet is an adaptation of the AntNet algorithm proposed by Di Caro and Dorigo [5]. In section 4, we compare the performance of W_AntNet algorithm with Dijkstra's shortest path algorithm for static topology. We also compare the performance of W_AntNet with ad hoc routing protocols AODV [11] and DSR [9].

2. Related Work

A variety of MABR algorithms have been proposed for wired and ad hoc networks. AntNet and Ant-Based Control (ABC) proposed by Schoonderwoerd *et al*. [14] are examples of MABR algorithms for wired networks. AntNet [5] has been shown to perform better than OSPF under varying and near saturation traffic loads. In addition, AntNet has been shown to converge to the shortest path with non-varying traffic loads [4]. ABC is similar to AntNet but it is designed for load balancing in circuit-switched networks and uses one-directional mobile agents [14].

MABR algorithms for ad hoc networks, such as AntHocNet [6], Ad hoc Networking with Swarm Intelligence (ANSI) [12], Ant-Colony-Based Routing Algorithm (ARA) [7], Ant-AODV [10] and Termite [13], use on-demand mobile agents for discovering routes. The on-demand flooding of mobile agents is similar to flooding of route request (RREQ) packets in on-demand protocols AODV and DSR. In ANSI, ARA and Termite, the pheromone decays and as a result after a certain period the routing tables are empty. The decay of routing table values is similar to AODV where the paths are valid for a certain duration of time. Thus, the paths obtained by using AODV or ANSI, ARA and Termite are identical. In addition, if a link breaks in ANSI, AntHocNet, Ant-AODV and ARA routing protocols inject route error packets similar to on-demand protocols. Indeed, the performance of ANSI, ARA and Ant-AODV is similar to AODV and DSR [7, 10, 12]. In our view, these algorithms are a variation of the on-demand protocols and deviate from the original idea of MABR algorithms in which routing tables are sufficient for routing data packets. We do not consider the performance of MABR algorithms that use on-demand flooding of mobile agents in this paper.



(a) Routing table: For any destination node i, $p_{xi}+p_{yi}+p_{zi}=1$ (b) Statistic Table: The mean μ_i and variance s_i represent the estimated mean and variance of the end-to-end delay to the node i. The moving observation window W_i is an array containing the end-to-end delay for node i.

Figure 1. The data structures: Routing table (T_k) and Statistics Table (M_k) for a node with neighbors x, y and z and a network with N nodes.

3. W_AntNet Algorithm

The W_AntNet algorithm is based on AntNet but uses neighbor discovery and added functionality to deal with node mobility. We summarize W_AntNet and refer to [4] for details.

3.1. Data Structures at a Node

Mobile agents communicate in an indirect way, through the information they concurrently read and write in two data structures stored at each network node k:

1. A routing table T_k, organized as a matrix with probabilistic entries as shown in Figure 1. Each row in the routing table corresponds to one destination in the network and each column corresponds to a neighbor of the current node. The routing table T_k defines the probabilistic routing policy currently adopted at node k: for each possible destination d and for each neighbor node n, T_k stores a probability value p_{nd} expressing the probability of choosing n as the next node when the destination is d such that:

$$\sum_{n \in N_k} p_{nd} = 1$$

   where $d \in [1, N]$ and $N_k = \{neighbors\,(k)\}$

2. A table $M_k(\mu_d, \sigma_d^2, W_d)$ containing statistics about the traffic distribution over the network as seen by the local

node k. For each destination d in the network, the table M_k contains a moving observation window W_d, an estimated mean μ_d and an estimated variance σ_d^2. The moving observation window W_d, of size W_{max}, represents an array containing the trip times of last W_{max} forward ants that travel from the node k to the destination d. The moving observation window W_d is used to compute the best trip time t_{best_d} i.e., the best trip time experienced by a forward ant travelling from the node k to the destination d among the last W_{max} forward ants that travel from the node k to the destination d. The mean μ_d and variance σ_d^2 represent the mean and variance of the trip times experienced by the forward ants to move from the node k to the destination node d and are calculated using the exponential model:

$$\mu_d \longleftarrow \mu_d + \eta\,(t_{k\longrightarrow d} - \mu_d) \tag{1}$$

$$\sigma_d^2 \longleftarrow \sigma_d^2 + \eta\left((t_{k\longrightarrow d} - \mu_d)^2 - \sigma_d^2\right) \tag{2}$$

In (1) and (2), $t_{k\longrightarrow d}$ represents the newly observed forward ant's trip time to travel from the node k to the destination node d and $\eta \in (0, 1]$ is a factor that weighs the number of recent samples that will affect the mean μ_d and the variance σ_d^2.

3.2. AntNet Algorithm

The AntNet algorithm [5] can be described as follows:

1. At regular intervals, from every network node s, a forward ant $F_{s\longrightarrow d}$ is launched with a randomly selected destination node d. While travelling towards their destination nodes, the identifier of every visited node k and the time elapsed since the launching time of the forward ant to arrive at this k-th node are pushed onto a memory stack $S_{s\longrightarrow d}$ stored in the data field of the forward ant. Forward ants share the same queues as data packets, so they experience the same traffic delays as data packets.

2. At each node k, each forward ant chooses the next node as follows:

   - If all the neighboring nodes have not been visited, then the next neighbor is chosen among the nodes that have not been visited as:

     $$p'_{nd} = \frac{p_{nd} + \alpha l_n}{1 + \alpha\ (|N_k| - 1)} \tag{3}$$

     In (3), N_k represents the set of neighbors of the current node k and $|N_k|$ the cardinality of that set, i.e., the number of neighbors while the heuristic correction l_n is a normalized value $[0, 1]$ such that $1 - l_n$ is proportional to the length q_n of the queue of the link connecting the node k with its neighbor n:

     $$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{|N_k|} q_{n'}} \tag{4}$$

     The value of α in (3) weighs the importance of the instantaneous state of the node's queue with respect to the probability values stored in the routing table.

   - If all the neighboring nodes have been visited previously, then the next node is chosen uniformly among all the neighbors. In this case, since all the neighbors have been visited previously the forward ant is in a loop (cycle).

   - With a small probability ε, the next node is chosen uniformly among all the neighboring nodes. The parameter ε ensures that the network is being constantly explored, though it introduces an element of inefficiency in the algorithm [4].

3. If a cycle is detected, the cycle's nodes are popped from the ant's stack and all memory about the cycle is destroyed. If the cycle lasted longer than the lifetime of the forward ant before entering the cycle, the ant is destroyed. The lifetime of a forward ant is defined as the total time since the forward ant was generated.

4. When the destination node d is reached, the forward ant $F_{s\longrightarrow d}$ generates a backward ant $B_{d\longrightarrow s}$. The forward ant transfers all the memory contained in the stack $S_{s\longrightarrow d}$ to the backward ant, and dies. The backward ant takes the same path as the corresponding forward ant, but in the opposite direction. At each node k, the backward ant pops the stack $S_{s\longrightarrow d}$ to move to the next node. Backward ants use high priority queues to quickly propagate to the routing tables the information collected by the forward ants.

5. Arriving at a node k coming from a neighbor node h, the backward ant updates the two main data structures of the node, the local model of the traffic M_k and the routing table T_k, for all the entries corresponding to the destination node d.

   - The mean μ_d and variance σ_d^2 entries in the local model of traffic M_k are modified using (1) and (2). If the newly observed forward ant's trip time $t_{k\longrightarrow d}$ from the node k to the destination d is less then t_{best_d}, then t_{best_d} is replaced by $t_{k\longrightarrow d}$.

   - The routing table T_k is changed by incrementing the probability $p_{hd'}$ (i.e., the probability of

choosing neighbor h when destination is d') and decrementing, by normalization, the other probabilities $p_{nd'}$.The probability $p_{hd'}$ is increased by the reinforcement value r as:

$$p_{hd'} \longleftarrow p_{hd'} + r\,(1 - p_{hd'}) \tag{5}$$

The probabilities $p_{nd'}$ of the other neighboring nodes n for destination d' are decreased by the negative reinforcement as:

$$p_{nd'} \longleftarrow p_{nd'} - rp_{nd'},\ \forall\, n \neq h,\ n \in N_k \tag{6}$$

Thus, in AntNet, every path found by the forward ants receives a positive reinforcement.

The reinforcement value r used in (5) and (6) is a dimensionless constant $(0, 1]$ and is calculated as:

$$r = c_1 \frac{t_{best_d}}{t_{k \longrightarrow d}} + c_2 \frac{t_{sup} - t_{best_d}}{(t_{sup} - t_{best_d}) \ + \ (t_{k \longrightarrow d} - t_{best_d})} \tag{7}$$

In (7), the parameter t_{sup} represents a better estimate of mean and is calculated as:

$$t_{sup} = \mu_d + \frac{\sigma_d}{\sqrt{1-\gamma}\sqrt{|W_{max}|}} \tag{8}$$

where γ is the confidence level [4]. The first term in (7) evaluates the ratio between the current trip time and the best trip time observed over the moving observation window. The second term is a correction factor and indicates how far the value of $t_{k \longrightarrow d}$ is from t_{best_d} in relation to the extension of the confidence interval [5]. The values of $c_1$and c_2 indicate the relative importance of each term.

The value r calculated in (7) is finally transformed by means of a squash function $s(x)$ defined by:

$$s(x) = \frac{1}{1 + \exp\left(\frac{a}{x|N_k|}\right)}\ ,\ \text{where } x \in (0, 1],\ a \in \mathbb{R}^+ \tag{9}$$

$$r \longleftarrow \frac{s(r)}{s(1)} \tag{10}$$

The squash function $s(x)$ is used so that small values of r would have negligible effect in updating the routing tables [5]. Due to the squash function $s(x)$, the low values of r are reduced further, and therefore do not contribute in the update of routing tables[1]. Figure 2 shows the effect of coefficient $\frac{a}{N_k}$ on the squash function $s(x)$. Figure 2 shows that if the value of coefficient $\frac{a}{N_k}$ is less than 1, then even low values of r get incremented due to the squash function $s(x)$. Thus, the value of parameter a should be chosen such that the coefficient $\frac{a}{N_k}$ is greater than 1.

[1] Low values of r indicate sub-optimal paths.

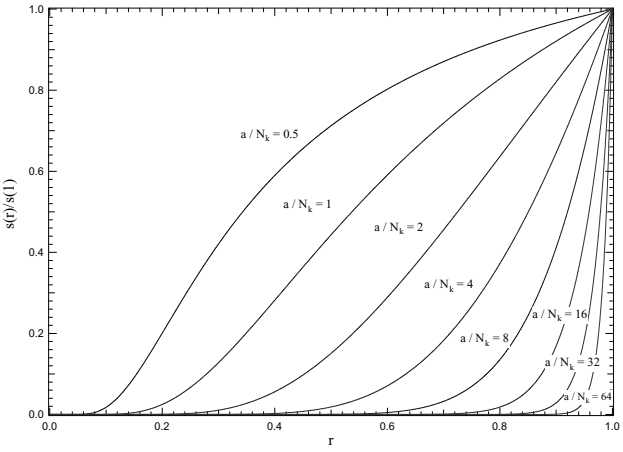


Figure 2. The squash function $s(x)$ for different values of the coefficient $\frac{a}{N_k}$.

Data packets use different routing tables than the forward ants for travelling from the source node to the destination node. The routing table values for data packets are obtained by re-mapping the routing table entries used by forward ants by means of a power function $g(v)$ and re-normalizing these entries.

$$g(v) = v^\beta, \beta > 1 \tag{11}$$

The power function $g(v)$ prevents the data packets from choosing links with very low probability. The data packets have a fixed time to live (TTL); if the data packets do not arrive at the destination within the TTL, they are dropped. Table 1 lists the optimized values of different parameters used in W_AntNet [4].

Table 1. The values of different parameters used in W_AntNet.

Name, symbol	Value
α used in (5)	0.2
ε	0.1
c_1 used in (9)	0.7
c_2 used in (9)	0.3
a used in (11)	$p * N$
β used in (13)	3
Confidence interval (γ) used in (10)	0.95

3.3. Neighbor Discovery Protocol

At regular intervals, *hello* messages are exchanged between neighboring nodes. When the neighbors are lost or new neighbors are added, the routing tables are updated and the probability values are re-normalized. The forward ant packets and data packets waiting in the buffer are rerouted if the next hop neighbor is lost. However, the backward ant packets are dropped if the next hop neighbor is lost. This ensures that only paths that are stable during a sufficiently long time interval, i.e. the time between the creation of forward ant and the receipt of the backward ant appear in the routing tables.

3.4. Routing Overhead for W_AntNet

The routing overhead for obtaining a path to any destination is $O(N)$ in AODV and DSR since both these protocols use controlled flooding or sequence numbers. To compute the complexity of W_AntNet, we use the fact that ad-hoc wireless networks can be modeled as a geometric random graph (or random graph) [8]. In W_AntNet, the forward ants perform random walks to search for the destination. Therefore, the worst-case routing overhead for a single update of the routing tables [2] is $O(N \log N)$. Moreover, the number of updates required for routing table convergence, i.e. probability for one of the entries in routing tables to be one, depends on the quality of different paths, network topology and the routing table update function. We show that multiple backward ants are needed for the routing tables to converge.

Consider a node k with N_k neighboring nodes. At $t = 0$, the probability of choosing neighbor n as the next hop for destination d, $p_{nd}(0) = \frac{1}{N_k}$. We assume that only the routing table values for neighbor n i.e., p_{nd} receive positive reinforcement. Under these assumptions, the probability $p_{nd}(t+1)$ is,

$$p_{nd}(t+1) = p_{nd}(t) + \delta p_{nd}(t) \tag{12}$$

where δ is a constant. After y updates,

$$p_{nd}(y) = p_{nd}(0)(1+\delta)^y$$

Since the routing table converges when $p_{nd}(y) \longrightarrow 1$, the value of y is :

$$y = \frac{\log(N_k)}{\log(1+\delta)} \tag{13}$$

where N_k is the number of neighbors of node k. Thus, the overhead for routing tables to converge in W_AntNet under the given assumptions is $O(N \log N_k \log N)$. The value of N_k varies from N for a complete mesh to $p(N-1)$ for a random graph or geometric random graph [8]. The geometric random graph is almost surely connected [8] if the link density $p > p_c \sim \frac{\log N}{N}$ when N is large and therefore, the minimum value of $N_k = \log N$.

4. Results

We performed extensive simulations of W_AntNet using our simulator [4] and NS-2 simulator[2]. Using NS-2, we compare the performance of W_AntNet with AODV and DSR. The simulations are performed for the *benchmark scenario* [3]: 50 nodes moving over an area of $1500m \times 300m$ for 900 s. In W_AntNet, there is training period (T.P.) at beginning of simulations during which data packets are not generated. The values of various parameters in W_AntNet are taken from [5, 4]. The mobility model in the simulations is random waypoint [3]. A pause time of 0s corresponds to continuous motion while a pause time of 900s corresponds to the static scenario. We assume a transmission range of 250m for each node in our simulator. Using these parameters, the average node degree is $\frac{N}{A} * \pi T^2 = \frac{50*\pi*(250)^2}{1500*300} \approx 22$ and the network diameter or worst case hopcount is $\frac{\sqrt{1500^2+300^2}}{250} \approx 6$.

4.1. Static versus Dynamic Topology

We compare the performance of W_AntNet for static and dynamic topologies. The MAC layer is ignored in these simulations. Moreover, we assume that each node is able to remove 1 packet/ms from the queue and no TTL is set for data packets. The routing tables are updated based on the hopcount between the source and destination nodes so that the performance is independent of the packet size, capacity of the links and the queueing delays. Thus, instead of (3), $p'_{nd} = p_{nd} + \varepsilon$ is used for choosing the next hop. Figure 3 shows the probability distribution of hopcount for W_AntNet with different pause times. To verify our analysis, we also plot the number of connectivity changes in our simulation model and the model used by Broch *et al.* [3] (insert in Figure 3).

The legend in Figure 3 shows expected hopcount for W_AntNet for different pause times. Figure 3 shows that W_AntNet performs similar to the Dijkstra's algorithm in a static topology. However, when mobility is introduced (pause time is less than 900s) the expected hopcount increases which points to a significant amount of packets in loops. Figure 4 shows the percentage of data packets with loops as a function of the pause time. Figure 4 also shows results for W_AntNet with look-ahead[3]. The increase of data packets in loops with increased mobility in W_AntNet is in sharp contrast to AODV and DSR that are shown to be loop-free [11, 9]. Sequence numbers in AODV excludes loops at all times while DSR uses source routing that is inherently loop-free. Figure 4 also shows that look-ahead re-

[2] NS-2 NETWORK SIMULATOR, *http://www.isi.edu/nsnam/ns*, 2005.

[3] Look-ahead means that if the destination is among the neighbors, then the destination is chosen as the next hop

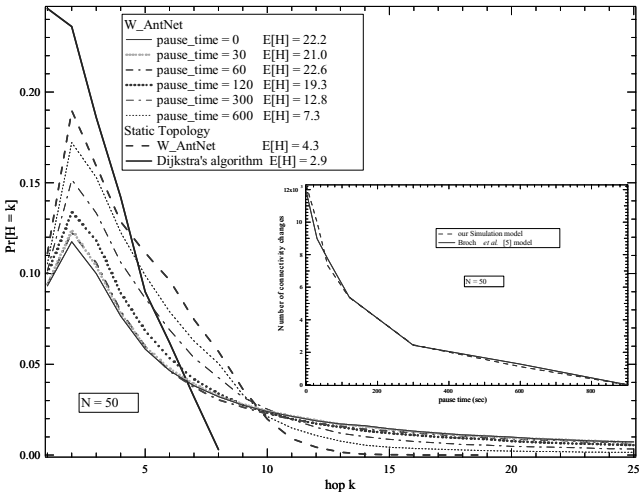


Figure 3. The probability distribution of hopcount for W_AntNet algorithm as a function of pause time. The legend shows the expected hopcount for different values of pause times.

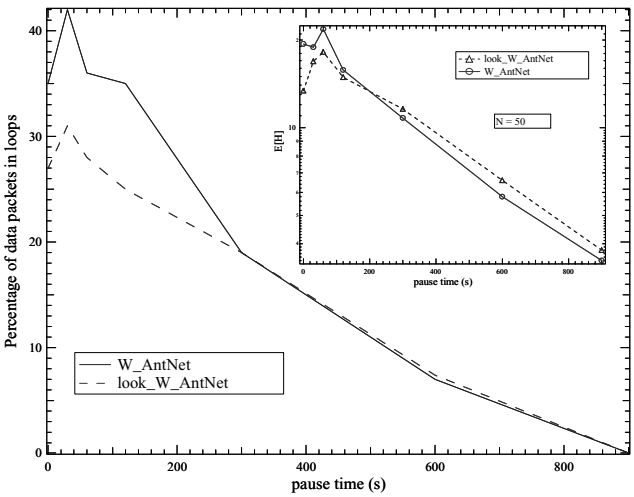


Figure 4. Percentage of data packets with a loop in their path as a function of pause time for W_AntNet (look_W_AntNet). The insert shows the expected hopcount in W_AntNet for different pause times.

duces the percentage of data packets in loops at increased mobility.

4.2. NS-2 Simulations

The default settings for all the experiments and the simulation code and parameters for AODV, DSR are taken from the CMU/Monarch extensions for NS-2 [3]. The MAC layer is 802.11 and the interface queue size is assumed to be 100 packets. The number of CBR sources is 10 and the data rate is 4 packets/sec. The capacity of links is 2 Mbps. In W_AntNet, we assume that each node can store 50 packets in the low and high priority queues.

4.2.1 Case Study 1: Static Network

We compare the packet delivery ratio (PDR) and the end-to-end delay for AODV, DSR and W_AntNet for a static scenario. The size of data packets is varied from 64 bytes to1024 bytes. We also show results for W_AntNet when the forward ants are generated only during the T.P. (W_AntNet_antsTP). This reduces the number of control packets in the network. Since the topology is static and the amount of data traffic is small, the generation of forward ants only during T.P. is sufficient for routing under these conditions.

The PDR and end-to-end delay in Figure 5 show that W_AntNet performs similar to AODV and DSR when forward ants are generated only during the T.P. However, as the forward ant generation rate is increased from 0 (W_AntNet_onlyTP) to 2 forward ants/s, the routing overhead causes congestion in the queues. Thus, W_AntNet causes a dual problem in ad hoc networks. A large number of forward ants cause congestion in the network. However, with increasing mobility, more forward ants need to be generated to account for frequent changes in the topology.

4.2.2 Case Study 2: Mobile Scenario

We study the performance of different routing protocols with mobility. The packet size is assumed to be 64 bytes. Figure 6 shows the PDR, end-to-end delay and the routing overhead (measured in number of bytes since the size of control packets varies in W_AntNet) for AODV, DSR, W_AntNet and W_AntNet with look-ahead (W_AntNet_look). We also reduce the routing overhead in W_AntNet by generating forward ants only from source-destination pairs that have data packets to send (W_AntNet_onlysrc). Thus, the remaining nodes in the network do not maintain up-to-date routing tables.

Figure 6 shows that with high node degree, AODV and DSR perform well in terms of PDR and end-to-end delay. Figure 6 shows that using optimizations such as look-ahead and generating ants only from source-destination pairs reduces the overhead in W_AntNet. However, the routing overhead is still more than in AODV and DSR. As a result, the PDR for W_AntNet_look and W_AntNet_onlysrc

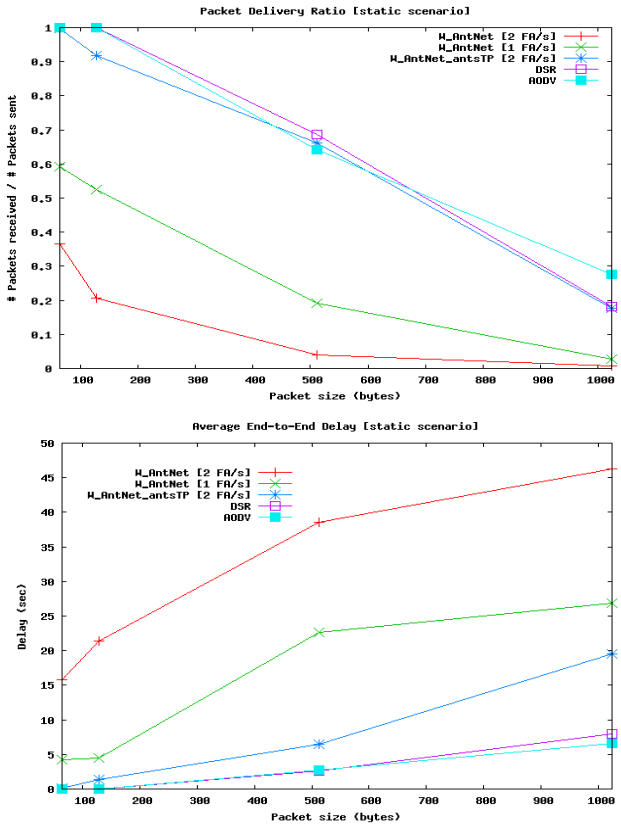


Figure 5. The PDR and end-to-end delay for W_AntNet, AODV and DSR for static topology.

is still lower than AODV and DSR.

4.2.3 Case Study 3: Large Area Network (Sparse Graph Topology)

In this case, we compare the performance of different protocols in a larger area with the same number of nodes ($N = 50$). The area over which the nodes move is $2000 \times 2000 m^2$. Under these conditions, the average node degree is 2.5 and the worst case hopcount is given by 11. The number of CBR sources is also increased to 20. Figure 7 shows the PDR and end-to-end delay for AODV, DSR and W_AntNet. The packet size is assumed to be 64 bytes.

This scenario leads to a sparse graph topology and an increase in network diameter. As a result, the performance of all three protocols degrades considerably. Thus, the simulations show that all three protocols have scalability problems. To improve the scalability of routing protocols, additional schemes such as clustering need to be implemented [15]. Figure 7 also shows that the use of look-ahead does not im-

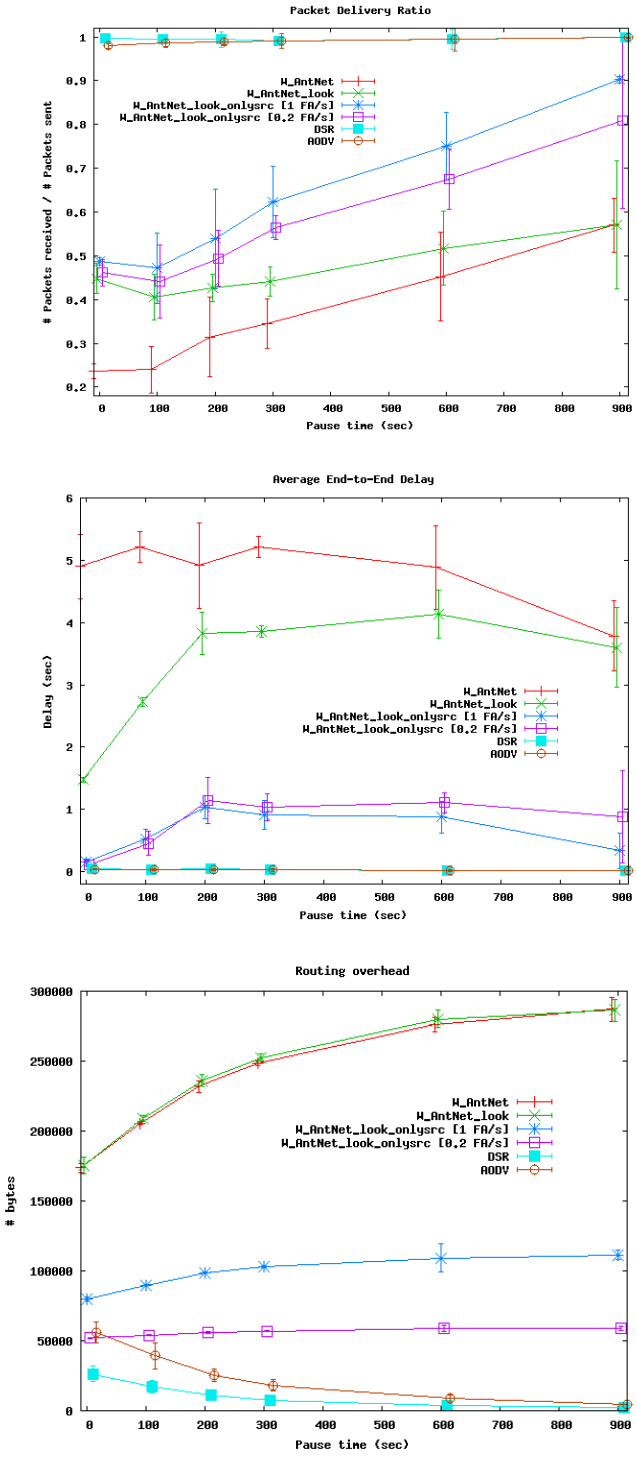


Figure 6. The PDR, end-to-end delay and routing overhead for W_AntNet, AODV and DSR for mobile network of 50 nodes moving in an area $1500 \times 300 m^2$.

prove the performance of W_AntNet since the average node degree is small.

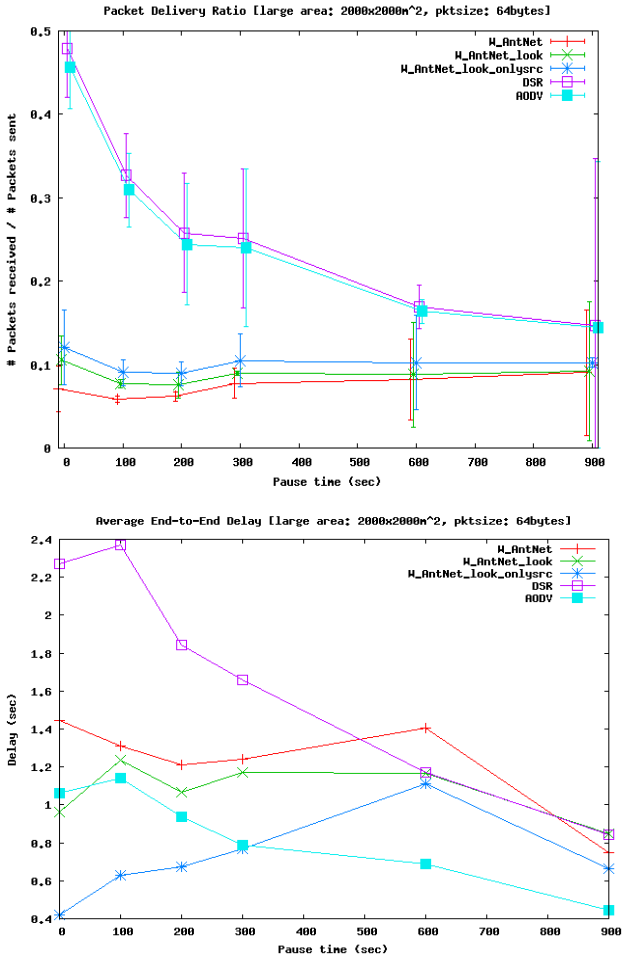


Figure 7. The PDR and end-to-end delay for W_AntNet, AODV and DSR for different values of the pause time. Nodes are moving in an area 2000m$\times$2000m.

5. Conclusion

The performance of W_AntNet is comparable to the shortest path algorithm for static topology but is dependent on the buffer size at the nodes. Since forward ants share the same queue as data packets in W_AntNet, a high ant generation rate leads to congestion in the network. This causes W_AntNet to perform poorly compared to AODV and DSR when the size of the buffer is small. In a dynamic topology, a significant amount of packets in W_AntNet have loops. This can be attributed to the non-convergence of routing tables in W_AntNet. Therefore, with mobility, the performance of W_AntNet deteriorates in comparison to AODV and DSR that are loop-free.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47:445–487, 2005.

[2] C. Avin and G. Ercal. On the cover time of random geometric graphs. *Lecture notes in computer science ISSN 0302-9743*, 2005.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. *Proc. IEEE/ACM MOBICOM*, 47:445–487, 1998.

[4] S. S. Dhillon and P. Van Mieghem. Performance analysis of the AntNet algorithm. *Computer Networks*, in print, 2007.

[5] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communication networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[6] G. Di Caro, F. Ducatelle, and L. M. Gambardella. AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. *Proc. 8th International Conf. on Parallel Problem Solving from Nature*, pages 461–470, 2004.

[7] M. Güneş and O. Spaniel. Routing algorithms for mobile multi-hop ad-hoc networks. *Proc. IFIP Conf. on Network Control and Engineering for QoS (Net-Con)*, pages 120–138, 2003.

[8] R. Hekmat and P. Van Mieghem. Study of connectivity in wireless ad-hoc networks with an improved radio model. *Proc. International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2004.

[9] D. Johnson, D. Maltz, Y.-C. Hu, and J. Jetcheva. The dynamic source routing protocol for mobile ad hoc networks. *Internet Draft, draft-ietf-manet-dsr09-11.txt, work in progress*, April 2003.

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

[11] C. Perkins, E. Royer, and S. Das. Ad hoc on demand distance vector routing. *Internet Draft, draft-ietf-manet-aodv-11.txt, work in progress*, August 2002.

[12] S. Rajagopalan and C. Shen. ANSI: A unicast routing protocol for mobile ad hoc networks using swarm intelligence. *Proc. International Conf. on Artificial Intelligence*, pages 24–27, 2005.

[13] M. Roth and S. Wicker. Termite: Emergent ad-hoc networking. *Proc. 2nd MedHocNet*, 2003.

[14] R. Schoonderwoerd, O. Holland, and J. Bruten. Ant-like agents for load balancing in telecommunication networks. *Proc. First International Conf. on Autonomous agents, 1997.*, 1997.

[15] P. Van Mieghem. *Data Communications Networking*. Techne Press, Amsterdam, The Netherlands, 2006.