AntHocNet: an Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks

Gianni Di Caro*, Frederick Ducatelle and Luca Maria Gambardella

Istituto Dalle Molle sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH-6928 Manno-Lugano, Switzerland
{gianni, frederick, luca}@idsia.ch

Abstract. In this paper we present AntHocNet, a new algorithm for routing in mobile ad hoc networks. Due to the ever changing topology and limited bandwidth it is very hard to establish and maintain good routes in such networks. Especially reliability and efficiency are important concerns. AntHocNet is based on ideas from Ant Colony Optimization. It consists of both reactive and proactive components. In a reactive path setup phase, multiple paths are set up between the source and destination of a data session, and during the course of the communication session, ants proactively test existing paths and explore new ones. In simulation tests we show that AntHocNet can outperform AODV, one of the most important current state-of-the-art algorithms, both in terms of end-to-end delay and packet delivery ratio.

1 Introduction

In recent years there has been an increasing interest in Mobile Ad Hoc Networks (MANETs) [13]. In this kind of networks, all nodes are mobile, and they communicate with each other via wireless connections. There is no fixed infrastructure. All nodes are equal and there is no centralized control or overview. There are no designated routers: all nodes can serve as routers for each other, and data packets are forwarded from node to node in a multi-hop fashion.

Routing is the task of directing data flow from source to destination maximizing network performance. This is particularly difficult in MANETs. Due to the mobility of the nodes, the topology of the network changes constantly, and paths which were initially efficient can quickly become inefficient or even infeasible. This means that routing information should be updated more regularly than in wired networks, so that in principle more routing control packets are needed. However, this is a problem in MANETs, since the bandwidth of the wireless medium is very limited, and the medium is shared: nodes can only send or receive data if no other node is sending in their neighborhood. The access

* Corresponding author. This work was partially supported by the Future & Emerging Technologies unit of the European Commission through project "BISON: Biology-Inspired techniques for Self Organization in dynamic Networks" (IST-2001-38923) and by the Hasler Foundation through grant DICS-1830.

to the shared channel is controlled by protocols at the Medium Access Control layer (MAC), such as ANSI/IEEE 802.11 DCF [7] (the most commonly used in MANETs), which in their turn create extra overhead.

In this work we propose *AntHocNet*, a new MANET routing algorithm based on ideas from ant-based routing. For wired networks, a number of successful ant-based routing algorithms exist (eg. ABC [14] and AntNet [3]). They are based on the pheromone trail laying-following behavior of real ants and the related framework of ant colony optimization (ACO) [4]. The main idea is to continuously sample possible paths with ant-like agents, and to indicate the quality of paths by means of artificial pheromone variables. Multiple paths are made available this way, and data packets are stochastically spread over them following the pheromone values. Ant-based routing algorithms exhibit a number of desirable properties for MANET routing: they work in a distributed way, are highly adaptive, are robust, and provide automatic load balancing.

In this paper, we aim to propose an algorithm which can work efficiently in MANETs, while still maintaining those properties which make ant-based algorithms so appealing. The rest of this paper is organized as follows. In section 2 we describe related work in MANET and ant-based routing. Section 3 contains the description of our algorithm and in section 4 we present simulation results.

2 Related work: MANET routing and ant-based routing

The specific challenges and possible applications of MANETs have made this a very popular research area, and a lot of routing algorithms have been proposed. People traditionally classify these algorithms as either *proactive* or *reactive*. In purely proactive protocols (e.g., DSDV [11]) nodes try to maintain at all times routes to all other nodes. This means that they need to keep track of all topology changes, which can become difficult if there are a lot of nodes or if they are very mobile. Therefore, reactive protocols (e.g., AODV [12] or DSR [8]) are in general more scalable (see [2]). In these protocols, nodes only gather routing information on demand: only when they have data for a certain destination they construct a path, and only when the path becomes infeasible they search a new path. In this way they greatly reduce the routing overhead, but they can suffer from oscillations in performance since they are never prepared for disruptive events. *Hybrid algorithms* like ZRP [6] have both a proactive and a reactive component, in order to try to combine the best of both worlds.

Most of the algorithms are single path: at any time, they use only one path between source and destination. Multipath routing (see [10] for an overview) offers an interesting alternative in terms of link failure robustness and load balancing. Some algorithms create multiple paths at path setup time, and use the best of these until it fails, after which they switch to the second best and so on (e.g., AODV-BR [9]). A problem with this way of working is that alternative paths are often infeasible by the time they need to be used. Moreover, when only the best path is used, one looses the opportunity to spread data packets over the different paths, a practice which can improve the network throughput.

The first ant-based routing algorithms were ABC [14] and AntNet [3]. Both algorithms follow a similar general strategy. Nodes send ant agents out at regular intervals to randomly chosen destinations. The main aim of the ants is to sample the paths, assign a quality to them, and use this information to update the routing tables in the nodes they pass. These routing tables contain an entry for each destination and each neighbor, indicating the goodness of going over this neighbor on the way to the destination. This goodness value is called pheromone. This pheromone information is used for the routing of both ants and data packets: all packets are routed stochastically, choosing with a higher probability those links with higher pheromone values. If enough ants are sent to the different destinations, nodes keep up-to-date information about the best paths, and automatically adapt their data load spreading to this.

Ant-based routing algorithms have a number of properties which are desirable in MANETs: they are highly adaptive to network changes, use active path sampling, are robust to agent failures, provide multipath routing, and take care of data load spreading. However, the fact that they crucially rely on repeated path sampling can cause significant overhead if not dealt with carefully. There have been a number of attempts to design ant-based routing algorithms for MANETs. Examples are ARA [5] and PERA [1]. However, these algorithms loose much of the proactive sampling and exploratory behavior of the original ant-based algorithms in their attempt to limit the overhead caused by the ants.

3 AntHocNet

AntHocNet is a hybrid multipath algorithm. When a data session is started at node s with destination d, s checks whether it has up-to-date routing information for d. If not, it reactively sends out ant-like agents, called *reactive forward ants*, to look for paths to d. These ants gather information about the quality of the path they followed, and at their arrival in d they become *backward ants* which trace back the path and update routing tables. The routing table $\mathcal{T}^i$ in node i contains for each destination d and each possible next hop n a value $\mathcal{T}^i_{nd} \in \mathbb{R}$. $\mathcal{T}^i_{nd}$ is an estimate of the goodness of the path over n to d, which we call pheromone. In this way, pheromone tables in different nodes indicate multiple paths between s and d, and data packets can be routed from node to node as datagrams. They are stochastically spread over the paths: in each node they select the next hop with a probability proportional to its pheromone value. Once paths are set up and the data session is running, s starts to send *proactive forward ants* to d. These ants follow the pheromone values similarly to data packets. In this way they can monitor the quality of the paths in use. Moreover, they have a small probability of being broadcasted, so that they can also explore new paths. In case of link failures, nodes either try to locally repair paths, or send a warning to their neighbors such that these can update their routing tables. In the rest of this section we describe each of these functions in detail.

3.1 Reactive path setup

Reactive forward ants looking for a destination d are either broadcasted or unicasted, according to whether or not the node they are currently in has routing information for d. Due to the broadcasting, ants can proliferate quickly over the network, following different paths to the destination. When a node receives several ants of the same generation (i.e., they started as the same original forward ant at the source), it will compare the path travelled by the ant to that of the previously received ants of this generation: only if its number of hops and travel time are both within a certain factor (a parameter which we empirically set to 1.5) of that of the best ant of the generation, it will forward the ant. Using this policy, overhead is limited by removing ants which follow bad paths, while the possibility to find multiple good paths is not hindered.

The main task of the reactive forward ant is to find a path connecting s and d. It keeps a list $\mathcal{P}$ of the nodes $[1, \ldots, n]$ it has visited. Upon arrival at the destination d, the forward ant is converted into a *backward ant*, which travels back to the source retracing $\mathcal{P}$. The backward ant incrementally computes an estimate $\hat{T}_{\mathcal{P}}$ of the time it would take a data packet to travel over $\mathcal{P}$ towards the destination, which is used to update routing tables. $\hat{T}_{\mathcal{P}}$ is the sum of local estimates $\hat{T}_{i \to i+1}$ in each node $i \in \mathcal{P}$ of the time to reach the next hop $i+1$: $\hat{T}_{\mathcal{P}} = \sum_{i=1}^{n-1} \hat{T}_{i \to i+1}$. The value of $\hat{T}_{i \to i+1}$ is defined as $(Q^i_{mac} + 1)\hat{T}^i_{mac}$: the product of the estimate of the average time to send one packet, $\hat{T}^i_{mac}$, times the current number of packets in queue (plus one) to be sent at the MAC layer, Q^i_{mac}. $\hat{T}^i_{mac}$ is calculated as a running average of the time elapsed between the arrival of a packet at the MAC layer and the end of a successful transmission. So if t^i_{mac} is the time it took to send a packet from node i, then node i updates its estimate as follows: $\hat{T}^i_{mac} = \alpha \hat{T}^i_{mac} + (1-\alpha) t^i_{mac}$ with $\alpha \in [0, 1]$. Since $\hat{T}^i_{mac}$ is calculated at the MAC layer it includes channel access activities, so it takes into account local congestion of the shared medium. Forward ants calculate a similar time estimate, which is used for filtering the ants, as mentioned above.

At each intermediate node $i \in \mathcal{P}$, the backward ant virtually sets up a path towards the destination d, creating or updating routing table entries $\mathcal{T}^i_{nd}$. Upon arrival in a node i from its neighbor n, the ant creates an entry in the routing table $\mathcal{T}^i$, indicating n as next hop to take from this node in order to reach d. The entry will contain a pheromone value $\mathcal{T}^i_{nd}$, which is an indication of the goodness of the path going to destination d over next hop n. The pheromone value represents an average of the inverse of the cost, in terms of both estimated time and number of hops, to travel to d through n. If $\hat{T}_{i \to d}$ is the travelling time estimated by the ant, and h is the number of hops, the pheromone value is defined as: $\tau_{id} = \left((\hat{T}_{s \to d} + h T_{hop})/2\right)^{-1}$, where T_{hop} is a fixed value representing the time of taking one hop in unloaded conditions. Taking this average is a way to avoid possibly large oscillations in the time estimates gathered by the ants (e.g., due to local bursts of traffic) and to take into account both end-to-end delay and number of hops. If there was already an entry $\mathcal{T}^i_{nd}$ in $\mathcal{T}^i$, its value is updated using a weighted average: $\mathcal{T}^i_{nd} = \gamma \mathcal{T}^i_{nd} + (1-\gamma)\tau_{id},\ \gamma \in [0, 1]$ (γ and α were set to 0.7 in the experiments).

3.2 Stochastic data routing

The path setup phase described above creates a number of good paths between source and destination, indicated in the routing tables of the nodes. Data can then be forwarded between nodes according to the values of the pheromone entries. Nodes in AntHocNet forward data stochastically. When a node has multiple next hops for the destination d of the data, it will randomly select one of them, with the probability P_{nd} of a next hop n assigned as the square of its pheromone: $P_{nd} = \frac{\mathcal{T}_{nd}^2}{\sum_{i \in \mathcal{N}_d} \mathcal{T}_{id}^2}$. We take the square in order to be more greedy with respect to the better paths. According to this strategy, we do not have to choose a priori how many paths to use: their number will be automatically selected in function of their quality.

The probabilistic routing strategy leads to data load spreading with consequent *automatic load balancing*. When a path is clearly worse than others, it will be avoided, and its congestion will be relieved. Other paths will get more traffic, leading to higher congestion, which will make their end-to-end delay increase. By continuously adapting the data traffic, the nodes try to spread the data load evenly over the network. This is quite important in MANETs, because the bandwidth of the wireless channel is very limited. Of course, to do this properly, it is important to frequently monitor the quality of the different paths. To this end we use the proactive ants.

3.3 Proactive path maintenance and exploration

While a data session is running, the source node sends out proactive forward ants according to the data sending rate (one ant every n^{th} data packet). They follow the pheromone values in the same way as the data (although the pheromone values are not squared, so that they sample the paths more evenly), but have a small probability at each node of being broadcasted. In this way they serve two purposes. If a forward ant reaches the destination without a single broadcast it simply samples an existing path. It gathers up-to-date quality estimates of this path, and updates the pheromone values along the path from source to destination. A backward ant does the same for the direction from the destination back to the source. If on the other hand the ant got broadcasted at any point, it will leave the currently known pheromone trails, and explore new paths.

After a broadcast the ant will arrive in all the neighbors of the broadcasting node. It is possible that in this neighbor it does not find pheromone pointing towards the destination, so that it will need to be broadcasted again. The ant will then quickly proliferate and flood the network, like a reactive forward ant does. In order to avoid this, we limit the number of broadcasts to two. If the proactive ant does not find routing information within two hops, it will be deleted. The effect of this mechanism is that the search for new paths is concentrated around the current paths, so that we are looking for path improvements and variations.

In order to guide the forward ants a bit better, we use *hello messages*[1]: using these messages, nodes know about their immediate neighbors and have pheromone information about them in their routing table. So when an ant arrives in a neighbor of the destination, it can go straight to its goal. Looking back at the ant colony inspiration of our model, this can be seen as pheromone diffusion: pheromone deposited on the ground diffuses, and can be detected also by ants further away. In future work we will extend this concept, to give better guidance to the exploration by the proactive ants. Hello messages also serve another purpose: they allow to detect broken links. This allow nodes to clean up stale pheromone entries from their routing tables.

3.4 Link failures

Nodes can detect link failures (e.g., a neighbor has moved far away) when unicast transmissions (of data packets or ants) fail, or when expected hello messages were not received. When a link fails, a node might loose a path to one or more destinations. If the node has other next hop alternatives to the same destination, or if the lost destination was not used regularly by data, this loss is not so important, and the node will just update its routing table and send a notification of the update to its neighbors. On the other hand, if the destination was regularly used for data traffic, and it was the node's only alternative for this destination, the loss is important and the node should try to *repair the path*. This is the strategy followed in AntHocNet, with the restriction that a node only repairs the path if the link loss was discovered with a failed data packet transmission.

After the link failure, the node broadcasts a *route repair ant* that travels to the involved destination like a reactive forward ant: it follows available routing information when it can, and is broadcasted otherwise. One important difference is that it has a maximum number of broadcasts (which we set to 2 in our experiments), so that its proliferation is limited. The node waits for a certain time (empirically set to 5 times the estimated end-to-end of the lost path), and if no backward repair ant is received, it concludes that it was not possible to find an alternative path to the destination which is removed from the routing table.

In the case the node still has other entries for the destination(s) involved in a link failure, but the lost next hop was its best alternative for the destination, or if the link failure was due to an ant packet, the node will only send a *notification* to its neighbors. Also in the case of a failed path repair it will send a similar notification. The notification contains a list of the destinations it lost a path to, and the new best estimated end-to-end delay and number of hops to this destination (if it still has entries for the destination). All its neighbors receive the notification and update their pheromone table using the new estimates. If

[1] Hello messages are short messages broadcasted every t_{hello} seconds (e.g., $t_{hello} = 1sec$) by the nodes. If a node receives a hello message from a new node n, it will add n as a new destination in its routing table. After that it expects to receive a hello from n every t_{hello} seconds. After missing a certain number of expected hello's (2 in our case), n will be removed.

they in turn lost their best or their only path to a destination due to the failure, they will broadcast the notification further, until all nodes along the different paths are notified of the new situation.

4 Simulation Experiments

We evaluate our algorithm in a number of simulation tests. We compare its performance with AODV [12] (with route repair), a state-of-the-art MANET routing algorithm and a de facto standard. In 4.1 we describe the simulation environment and the test scenarios, and in 4.2 we show and discuss the results.

4.1 Simulation Environment

As simulation software we used Qualnet, a discrete-event simulator developed by Scalable Networks as a follow-up of GloMoSim, which was designed by UCLA. Qualnet is specifically optimized to simulate large-scale MANETs, and comes with correct implementations of the most important routing protocols.

All our simulation scenarios are derived from the base scenario used in [2], which is an important reference. In this base scenario 50 nodes are randomly placed in an area of 1500×300 m^2. The area is rectangular in order to have more long paths. Within this area, the nodes move according to the random waypoint model [8]: each node randomly chooses a destination point and a speed, and moves to this point with the chosen speed. After that it stops for a certain pause time and then chooses a new destination and speed. The maximum speed in the scenario is $20m/s$ and the pause time is 30 seconds. The total length of the simulation is 900 seconds. Data traffic is generated by 20 constant bit rate (CBR) sources sending one 64-byte packet per second. Each source starts sending at a random time between 0 and 180 seconds after the start of the simulation, and keeps sending until the end. At the physical layer we use a two-ray signal propagation model. The transmission range is 300 meters, and the data rate is $2Mbit/s$. At the MAC layer we use the popular 802.11 DCF protocol.

The different test scenarios used below were derived from the base scenario by changing some of the parameters. In particular, we varied the pause time, the area dimensions and the number of nodes. For each new scenario, 5 different problems were created, by choosing different initial placements of the nodes and different movement patterns. The reported results are averaged over 5 different runs (to account for stochastic elements, both in the algorithms and in the physical and MAC layers) on each of the 5 problems.

4.2 Simulation Results

In a first set of experiments we progressively extended the long side of the simulation area. This has a double effect: paths become longer and the network becomes sparser. The results are shown in figure 1. In the base scenario, AntHocNet has a better delivery ratio than AODV, but a higher average delay.

For the longer areas, the difference in delivery ratio becomes bigger, and AODV also looses its advantage in delay. If we take a look at the 99^{th} percentile of the delay, we can see that the decrease in performance of AODV is mainly due to a small number of packets with very high delay. This means that AODV delivers packets with a very high delay jitter, a crucial problem in terms of quality of service (QoS). The jitter could be reduced by removing these packets with very high delay, but that would mean an even worse delivery ratio for AODV. Next

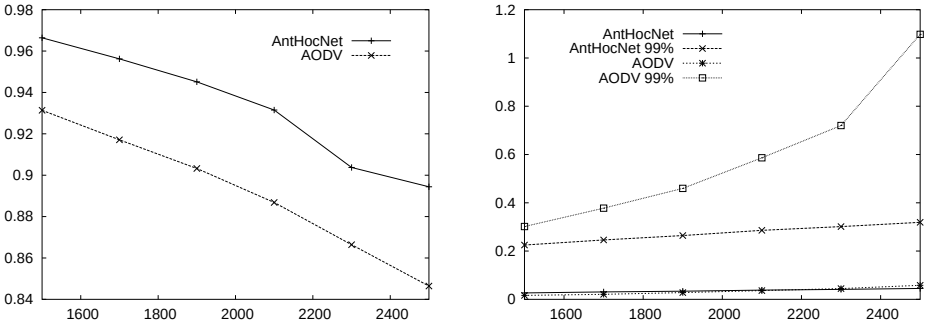

Fig. 1. On the left the delivery ratio (the fraction of sent packets which actually arrives at their destination) and on the right the average and the 99^{th} percentile of the delay per packet. On x-axis the long edge of the area: starting from the base scenario of $1500 \times 300\ m^2$, and ending at $2500 \times 300\ m^2$.

we changed the mobility of the nodes, varying the pause time between 0 seconds (all nodes move constantly) and 900 seconds (all nodes are static). The area dimensions were kept on $2500 \times 300\ m^2$, like at the end of the previous experiment (results for $1500 \times 300\ m^2$ were similar but less pronounced). In figure 2 we can see a similar trend as in the previous experiment. For easy situations (long pause times, hardly any mobility), AntHocNet has a higher delivery ratio, while AODV has lower delay. As the environment becomes more difficult (high mobility), the difference in delivery ratio becomes bigger, while the average delay of AntHocNet becomes better than that of AODV. Again, the 99^{th} percentile of AODV shows that this algorithm delivers some packets with a very high delay. Also AntHocNet has some packets with a high delay (since the average is above the 99^{th} percentile), but this number is less than 1% of the packets. In a last experiment we increased the scale of the problem. Starting from 50 nodes in a $1500 \times 500\ m^2$ area, we multiply both terrain edges by a scaling factor and the number of nodes by the square of this factor, up to 200 nodes in a $3000 \times 1000\ m^2$ area. The results, presented in figure 3, show again the same trend: as the problem gets more difficult, the advantage of AntHocNet in terms of delivery ratio increases, while the advantage of AODV in terms of average delay becomes a disadvantage. Again this is due to a number of packets with a very high delay.

The experiments described above show that AntHocNet has some clear advantages over AODV. First of all, AntHocNet gave a better delivery ratio than AODV in all scenarios. The construction of multiple paths at route setup, and

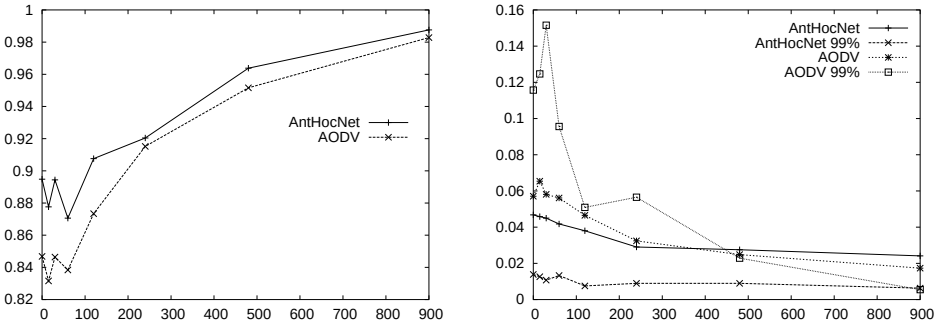

Fig. 2. On the left the delivery ratio and on the right the average and 99^{th} percentile of the delay. On the x-axis the node pause time in seconds.

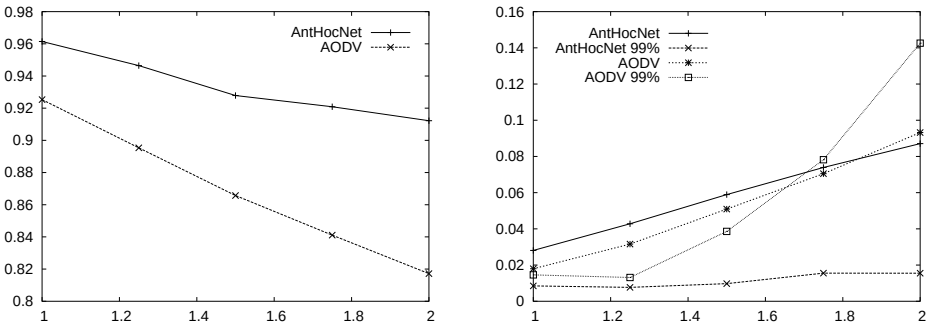

Fig. 3. On the left the delivery ratio and on the right the average and 99^{th} percentile of the delay. On the x-axis the scaling factor for the problem.

the continuous search for new paths with proactive ants ensures that there are often alternative paths available in case of route failures, resulting in less packet loss. Second, AntHocNet has a higher average delay than AODV for the simpler scenarios, but a lower average delay for the more difficult ones. The average delay of AODV increases sharply in each of the difficult scenarios, and the 99^{th} percentile figures indicate that this is mainly due to a fraction of packets which is delivered with an abnormally high delay. Moreover, the 95^{th} percentile (not shown in the figures) is usually lower for AODV than for AntHocNet, indicating that AODV still delivers most of its packets faster than AntHocNet. This is in line with the multipath nature of AntHocNet: since it uses different paths simultaneously, not all packets are sent over the shortest path, and so the average delay will be slightly higher. On the other hand, since AODV relies on just one path, delays can become very bad when this path becomes inefficient or invalid. This is especially likely to happen in difficult scenarios, with longer paths, lower node density or higher mobility, rather than in the dense and relatively easy base scenario. Delivering packets with low variability and low maximum delay is an important factor in QoS routing.

5 Conclusions and future work

We have presented AntHocNet, a new ant-based algorithm for routing in MANETs. It is a hybrid algorithm, combining reactive route setup with proactive route probing and exploration. In simulation experiments we show that AntHocNet can outperform AODV in terms of delivery ratio and average delay, especially in difficult scenarios. Also in terms of delay jitter, AntHocNet shows better results.

In future work we want to improve the exploratory working of proactive ants. By extending the concept of pheromone diffusion, more information about possible path improvements will be available in the nodes, and this information can guide proactive ants. This should lead to better results with less overhead. Also, we would like to try out a virtual circuit based approach. This could result in better control over paths, so that data delivery can be made more reliable.

References

1. 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.
2. J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. of 4th Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking*, 1998.
3. G. Di Caro and M. Dorigo. AntNet: Distributed stigmergic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.
4. M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
5. M. Günes, U. Sorges, and I. Bouazizi. ARA - the ant-colony based routing algorithm for manets. In *Proc. of 2002 ICPP Workshop on Ad Hoc Networks*, 2002.
6. Z. J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proc. of the IEEE Int. Conf. on Universal Personal Communications*, 1997.
7. IEEE 802.11 working group. ANSI/IEEE std. 802.11, 1999 edition: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. Technical report, ANSI/IEEE, 1999.
8. D. B. Johnson and D. A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer, 1996.
9. S.-J. Lee and M. Gerla. Aodv-br: Backup routing in ad hoc networks. In *Proceedings of IEEE WCNC 2000*, 2000.
10. S. Mueller, R. Tsang, and D. Ghosal. Multipath routing in mobile ad hoc networks: Issues and challenges. In *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.
11. C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994.
12. C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.
13. E. M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999.
14. R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, (2):169–207, 1996.