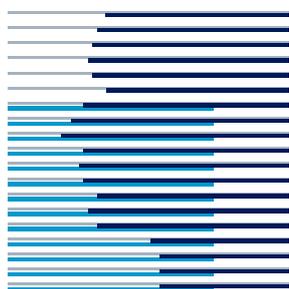


AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks

Gianni Di Caro, Frederick Ducatelle and Luca Maria Gambardella



Technical Report No. IDSIA-27-04-2004
September 2004

IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

A revised version of this report has been accepted for publication on the *European Transactions on Telecommunications, Special Issue on Self-organization in Mobile Networking, to appear, 2005*.

IDSIA is a joint institute of both University of Lugano (USI) and University of Applied Sciences of Southern Switzerland (SUPSI), and was founded in 1988 by the Dalle Molle Foundation which promoted quality of life.

AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in 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

Tel. +41 (0)91 6108568 Fax. +41 (0)91 6108661

Abstract

In this paper we describe AntHocNet, an algorithm for routing in mobile ad hoc networks. It is a hybrid algorithm, which combines reactive route setup with proactive route probing, maintenance and improvement. The algorithm is based on the Nature-inspired Ant Colony Optimization framework. Paths are learned by guided Monte Carlo sampling using ant-like agents communicating in a stigmergic way. In an extensive set of simulation experiments, we compare AntHocNet with AODV, a reference algorithm in this research area. We show that our algorithm can outperform AODV on different evaluation criteria. AntHocNet's performance advantage is visible over a broad range of possible network scenarios, and increases for larger, sparser and more mobile networks. AntHocNet is also more scalable than AODV.

1. Introduction

Mobile Ad Hoc Networks (MANETs) [38] are networks in which all nodes are mobile and communicate with each other via wireless connections. Nodes can join or leave the network at any time. 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.

Since a few years research interest in MANETs has been growing, and especially the design of MANET routing protocols has received a lot of attention. One of the reasons is that routing in MANETs is a particularly challenging task due to the fact that the topology of the network changes constantly, and paths which were initially efficient can quickly become inefficient or even infeasible. Moreover, control information flow in the network is very restricted. This is because 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 radio neighborhood. The access to the shared channel is controlled by protocols at the Medium Access Control layer (MAC), such as ANSI/IEEE 802.11 DCF [24] (the most commonly used in MANETs), which in their turn create extra overhead. It is therefore 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 like *insect societies*

*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 Swiss Hasler Foundation through grant DICS-1830.

show precisely these desirable properties. Making use of a number of relatively simple biological agents (e.g., the ants) a variety of different organized behaviors are generated at the system-level from the local interactions among the agents and with the environment. The robustness and efficacy of the collective behaviors of insect societies with respect to variations of environment conditions is a key-aspect of their biological success. Because of these same properties, they have recently become a source of inspiration for the design of routing algorithms for dynamic networks (as well as for the solution of several other classes of problems).

In this paper we describe *AntHocNet*, a hybrid routing algorithm for MANETs. AntHocNet's design 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)* [14].

It has been experimentally observed that ants in a colony can converge on moving over the shortest among different paths connecting their nest to a source of food [19, 14]. The main catalyst of this colony-level shortest path behavior is the use of a volatile chemical substance called *pheromone*: ants moving between the nest and a food source deposit pheromone, and preferentially move in the direction of areas of higher pheromone intensity. Shorter paths can be completed quicker and more frequently by the ants, and will therefore be marked with higher pheromone intensity. These paths will therefore attract more ants, which will in turn increase the pheromone level, until there is convergence of the majority of the ants onto the shortest path. The local intensity of the pheromone field, which is the overall result of the repeated and concurrent *path sampling* experiences of the ants, encodes a spatially distributed *measure of goodness* associated with each possible move. This form of distributed control based on indirect communication among agents which locally modify the environment and react to these modifications leading to a phase of global coordination of the agent actions is called *stigmergy* [45]. Stigmergic coordination is one of the keys to obtain self-organized behaviors not only in ant colonies but more generically across *social systems*, from insects to humans (e.g., [6, 16]). When stigmergy is at work, system's *protocols* (interfaces) play a prominent role with respect to *modules* (agents) [8]. Protocols are the rules that prescribe the characteristics of the allowed interfaces and of the information exchanged between modules, permitting system functions that could not be achieved by isolated modules. A good stigmergic model supplies global robustness, scalability, evolvability, and allows to fully exploit the potentialities of the modules and of modularity.

All these ingredients have been reverse-engineered in the framework of ACO, which exploits the mechanisms behind the ant colony shortest path behavior to define a Nature-inspired meta-heuristic for combinatorial optimization. ACO features a multi-agent organization, stigmergic communication among the agents, distributed operations, use of a stochastic decision policy to construct solutions, stigmergic learning of the parameters of the decision policy, and so on. It has been applied with success to a variety of combinatorial problems (e.g., travelling salesman, vehicle routing, scheduling, etc., see [14, 15] for overviews), as well as to routing (e.g., [12, 41, 43]). The first ACO routing algorithms were designed for wired networks (e.g., *AntNet* [12] for packet-switched networks and *ABC* [41] for circuit-switched networks). These algorithms exhibit a number of interesting properties which are also desirable for MANET routing: they work in a fully distributed way, are highly adaptive to network and traffic changes, use mobile agents for active path sampling, are robust to agent failures, provide multipath routing, and automatically 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 already been some attempts to design ACO routing algorithms for MANETs. Examples are ARA [21] and PERA [2]. 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.

With AntHocNet we aim to design an algorithm which can work efficiently in MANETs while

maintaining those properties which make ACO routing algorithms so appealing. In particular, while most of the previous algorithms for wired networks were mainly adopting a proactive scheme by periodically generating ant-like agents for all possible destinations, AntHocNet follows a hybrid approach. Ants are generated according to both *proactive* and *reactive* schemes.

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

2 Related literature

In this section we describe related literature. In 2.1 we give a short introduction to MANET routing algorithms, and in 2.2 we describe the basic elements of ACO routing. Then in 2.3 we give an overview of existing implementations of ACO routing for MANETs, and finally in 2.4 we indicate other MANET routing algorithms which contain ACO routing elements.

2.1 Routing in MANETs

In recent years a large number of MANET routing algorithms have been proposed (see [1] for an overview). These algorithms all deal with the dynamic aspects of MANETs in their own way, using reactive or proactive behavior, or a combination of both. *Reactive behavior* means that an algorithm only gathers routing information in response to an event, usually an event which triggers the need for new routes, such as the start of a data session or the failure of an existing route. *Proactive behavior* means that the algorithm also gathers information at other times, so that routing information is readily available when the event happens.

In the MANET literature, the classical distinction is between purely proactive, purely reactive, and hybrid algorithms. In *purely proactive* algorithms (e.g., DSDV [35]) 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. In *purely reactive* algorithms (e.g., AODV [34] and DSR [25]), nodes only gather routing information on demand: when a data session to a new destination starts, or when a route which is in use fails. Reactive algorithms are in general more scalable [4] since they greatly reduce the routing overhead, but they can suffer from oscillations in performance because they are never prepared for disruptive events. In practice, many algorithms are *hybrid algorithms* (e.g. ZRP [22]), using both proactive and reactive components in order to try to combine the best of both worlds.

2.2 ACO routing algorithms

The basic idea behind ACO algorithms for routing [11, 14] is the acquisition of routing information through sampling of paths using small control packets, which are called ants. The ants are generated concurrently 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 (e.g. end-to-end delay, number of hops, etc.), and uses this on its way back from d to s to update the routing information at the intermediate nodes. Ants always sample complete paths, so that routing information can be updated in a pure *Monte Carlo* way, without relying on bootstrapping information from one node to the other [44].

The routing tables contain for each destination a vector of real-valued entries, one for each known neighbor node. These entries are a measure of the goodness of going over that neighbor on the way to the destination. They are termed *pheromone* variables, and are continually updated according to path quality values calculated by the ants. The repeated and concurrent generation of path-sampling ants results in the availability at each node of a bundle of paths, each with an estimated measure of quality. In turn, the ants use the routing tables to define which path to

their destination they sample: at each node they stochastically choose a next hop, giving higher probability to those links which are associated with higher pheromone values.

This process is quite similar to the pheromone laying and following behavior of real ant colonies. Like their natural counterparts, the artificial ants are in practice autonomous agents, and through the updating and stochastic following of pheromone tables they participate in a stigmergic communication process. The result is a collective learning behavior, in which individual ants have low complexity and little importance, while the whole swarm together can collect and maintain up-to-date routing information.

This pheromone information is used for routing data packets, more or less in the same way as for the routing of ants: all packets are routed *stochastically*, choosing with a higher probability those links associated with higher pheromone values. In this way data for a same destination are spread over *multiple paths* (but with more packets going over the best paths), resulting in *load balancing*. For data packets, mechanisms are usually adopted to avoid low quality paths, while ants are more explorative, so that also less good paths are occasionally sampled and maintained as *backup paths* in case of failure or sudden congestion. In this way *path exploration* is kept separate from the use of paths by data. If enough ants are sent to the different destinations, nodes can keep up-to-date information about the best paths, and automatically adapt their data load spreading to this.

2.3 ACO routing in MANETs

The description of the previous subsection highlights a number of key ingredients of ACO routing: routing tables are adapted and maintained via continuous and concurrent Monte Carlo sampling of paths, data are stochastically spread over multiple paths, leading to automatic load balancing based on path qualities, all routing and control decisions are taken locally, and the system is robust with respect to agent failures. A number of attempts have been made to incorporate these features into a MANET routing algorithm. Important challenges hereby are the high change rate and in particular the limited bandwidth which conflicts with the continuous generation of ant packets.

Accelerated Ants Routing [17, 31] uses ant-like agents which go through the network randomly, without a specific destination, updating pheromone entries pointing to their source. In [5] the authors describe a location-based algorithm which makes use of ant agents to disseminate routing information; here the ants serve as an efficient form of flooding. *Ant-AODV* [30] 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)* [21] 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)* [2] 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).

Also other ACO routing algorithms [23, 37, 42] have been proposed for MANETs. In general, however, most of all these algorithms move quite far away from the original ACO routing ideas trying to obtain the efficiency needed in MANETs, and many of them are not very different from single-path on-demand algorithms.

2.4 Elements of ACO routing in other MANET routing algorithms

Some of the ingredients of ACO routing appear separately in other MANET routing algorithms. Especially the idea of *multipath routing* has received a lot of attention recently, both in order to improve reliability and end-to-end delay (see [32] for an overview). The algorithms differ in the way multiple paths are set up, maintained and used. At route setup time, the algorithm selects a number of paths. Some algorithms allow braided multiple paths [18], whereas others look for link [29] or node [48] disjoint paths, or even paths which are outside each other's interference range [47]. Once the paths are set up, they need to be maintained. Most algorithms manage the paths in a reactive way: they remove paths when a link break occurs, and only take action when no valid path to the destination is left. The idea of *proactively probing paths* to obtain up-to-date information about them and to detect failures can be found in few algorithms [18, 46]. *Proactively improving existing paths* is quite rare in MANET routing algorithms, although one possible approach is presented in [20] (in the context of single-path routing). The use of the multiple paths differs strongly among algorithms. In many of them, only one of the paths is used for data transport, while the others are only used in case of a failure in the primary path [26, 33]. Some algorithms spread data over the multiple paths in a simple, even way [27], and in a few cases *adaptive data load spreading* depending on the estimated quality of paths, similar to the ACO ideas, is explored [18, 46]. The quality of paths is usually assessed in terms of hop count or round trip time; *combining different metrics* is less common but can be important [9]. *Stochastic data spreading* is to the best of our knowledge unexplored outside the area of ACO routing algorithms.

3 AntHocNet

AntHocNet is a hybrid multipath algorithm, designed along the principles of ACO routing. It consists of both reactive and proactive components. It does not maintain routes to all possible destinations at all times (like the original ACO algorithms for wired networks), but only sets up paths when they are needed at the start of a data session. This is done in a *reactive route setup* phase, where ant agents called *reactive forward ants* are launched by the source in order to find multiple paths to the destination, and *backward ants* return to the source to set up the paths. According to the common practice in ACO algorithms, the paths are set up in the form of pheromone tables indicating their respective quality. After the route setup, *data packets are routed stochastically* over the different paths following these pheromone tables. While the data session is going on, the *paths are monitored, maintained and improved proactively* using different agents, called *proactive forward ants*. The algorithm reacts to *link failures* with either a local route repair or by warning preceding nodes on the paths. An earlier version of the algorithm described in the following appeared in [13].

3.1. Reactive path setup

When a source node s starts a communication session with a destination node d , and it does not have routing information for d available, it broadcasts a reactive forward ant F_d^s . Due to this initial broadcasting, each neighbor of s receives a replica $F_d^s(k)$ of F_d^s . In what follows, we will also refer to the set of replicas which originated from the same original ant as an *ant generation*. The task of each ant $F_d^s(k)$ is to find a path connecting s and d . At each node, an ant is either unicast or broadcast, according to whether or not the current node has routing information for d . The routing information of a node i is represented in its pheromone table \mathcal{T}^i . The entry $\mathcal{T}_{nd}^i \in \mathbb{R}$ of this table is the pheromone value indicating the estimated goodness of

going from i over neighbor n to reach destination d . If pheromone information is available, the ant will choose its next hop n with the probability P_{nd} :

$$P_{nd} = \frac{(T_{nd}^i)^{\beta_1}}{\sum_{j \in \mathcal{N}_d^i} (T_{jd}^i)^{\beta_1}}, \quad \beta_1 \geq 1,$$

where \mathcal{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 (although in current experiments β_1 is kept to 1).

If no pheromone information is available for d , the ant is broadcast. Due to this broadcasting, ants can proliferate quickly over the network, following different paths to the destination (although ants which have reached a maximum number of hops, related to the network diameter, are deleted). When a node receives several ants of the same generation, it will compare the path travelled by each ant to that of the previously received ants of this generation: only if its number of hops and travel time are both within an acceptance factor a_1 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 there is still the possibility to find multiple good paths. However, it does have as an effect that the ant which arrives first in a node is let through, while subsequent ants meet with selection criteria set by the best of the ants preceding them, which means that they have higher chances of being rejected. This can lead to “kite-shaped” paths, as shown in Figure 1a. In order to obtain a mesh of sufficiently disjoint multiple paths as shown in Figure 1b, which provides much better protection in case of link failures, we also consider in the selection policy the first hop taken by the ant. If this first hop is different from those taken by previously accepted ants, we apply a higher (less restrictive) acceptance factor a_2 than in the case the first hop was already seen before (in the experiments a_2 was set to 2 as opposed to $a_1 = 0.9$). A similar strategy can be found in [29].

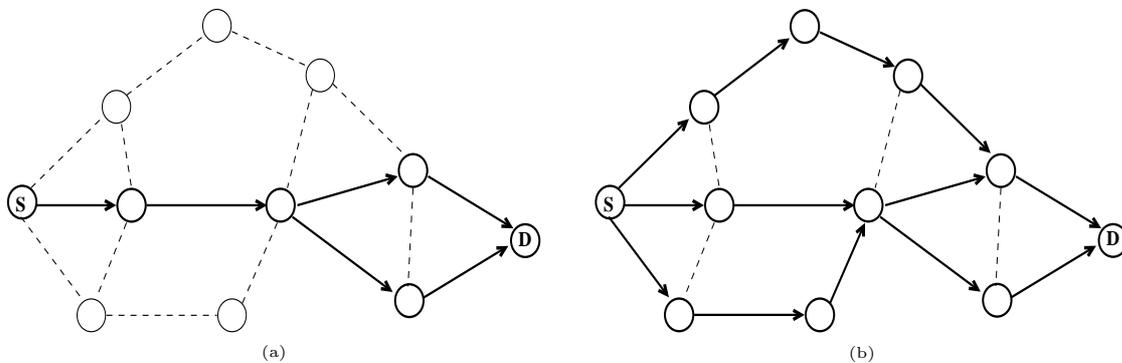


Figure 1. (a) “Kite-shaped” multiple paths, (b) a mesh of multiple paths

Each forward ant keeps a list \mathcal{P} of the nodes $[1, \dots, n]$ it has visited. Upon arrival at the destination d , it is converted into a *backward ant*, which travels back to the source retracing \mathcal{P} (if this is not possible because the next hop is not there, for instance due to node movements, the backward ant is discarded). 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+1}^i in each node $i \in \mathcal{P}$ of the time to reach the next hop $i + 1$:

$$\hat{T}_{\mathcal{P}} = \sum_{i=1}^{n-1} \hat{T}_{i+1}^i.$$

The value of \hat{T}_{i+1}^i is defined as the product of the estimate of the average time to send one

packet, \hat{T}_{mac}^i , times the current number of packets in queue (plus one) to be sent at the MAC layer, Q_{mac}^i :

$$\hat{T}_{i+1}^i = (Q_{mac}^i + 1)\hat{T}_{mac}^i .$$

\hat{T}_{mac}^i 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_{mac}^i is the time it took to send a packet from node i , then node i updates its estimate as:

$$\hat{T}_{mac}^i = \alpha \hat{T}_{mac}^i + (1 - \alpha)t_{mac}^i,$$

with $\alpha \in [0, 1]$. Since \hat{T}_{mac}^i 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 $\hat{T}_{\mathcal{P}}$, which is used for the filtering of 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 the pheromone table entry \mathcal{T}_{nd}^i in \mathcal{T}^i . The pheromone value in \mathcal{T}_{nd}^i represents a running 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}_d^i is the travelling time estimated by the ant, and h is the number of hops, the value τ_d^i used to update the running average is defined as:

$$\tau_d^i = \left(\frac{\hat{T}_d^i + hT_{hop}}{2} \right)^{-1},$$

where T_{hop} is a fixed value representing the time to take one hop in unloaded conditions. Defining τ_d^i like this 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. The value of \mathcal{T}_{nd}^i is updated as follows:

$$\mathcal{T}_{nd}^i = \gamma \mathcal{T}_{nd}^i + (1 - \gamma)\tau_d^i, \quad \gamma \in [0, 1],$$

where γ and α were both 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 probability P_{nd} . P_{nd} is calculated in the same way as for the reactive forward ants, but with a higher exponent (in the experiments set to 2), in order to be more greedy with respect to the better paths:

$$P_{nd} = \frac{(\mathcal{T}_{nd}^i)^{\beta_2}}{\sum_{j \in \mathcal{N}_d^i} (\mathcal{T}_{jd}^i)^{\beta_2}}, \quad \beta_2 \geq \beta_1 .$$

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 according to the estimated quality of the paths. If the estimates are kept up-to-date (which is done using the proactive ants described in Subsection 3.3), this leads to *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.

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 data packets). They are normally unicast, choosing the next hop according to the pheromone values using the same formula as the reactive forward ants, but also have a small probability at each node of being broadcast (this probability was set to 0.1 in the experiments). 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 the backward ant updates the pheromone values along the path, just like the reactive backward ants do. If on the other hand the ant got broadcast at any point, it will leave the currently known paths, and explore new ones.

After a broadcast the ant will arrive in all the neighbors of the broadcasting node. It is possible that in these neighbors it does not find pheromone pointing towards the destination, so that it needs to be broadcast again. The ant will then quickly proliferate and flood the network, like reactive forward ants do. In order to avoid this, we limit the number of broadcasts to n_b (n_b was set to 2 in the experiments). If the proactive ant does not find routing information within n_b hops, it is 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 better, we use *hello messages*. These are short messages (in our case containing just the address of the sender) broadcast 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 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 (*allowed-hello-loss* = 2 in our case), n will be removed. 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 allows nodes to clean up stale pheromone entries from their routing tables.

3.4. Link failures

In AntHocNet, each node tries to maintain an updated view of its immediate neighbors at each moment, in order to detect link failures as quickly as possible, before they can lead to transmission errors and packet loss. The presence of a neighbor node can be confirmed when a hello message is received, or after any other successful interception or exchange of signals. The disappearance of a neighbor is assumed when such an event has not taken place for a certain amount of time, defined by $t_{hello} \times allowed-hello-loss$, or when a unicast transmission to this neighbor fails.

When a neighbor is assumed to have disappeared, the node takes a number of actions. In the first place, it removes the neighbor from its neighbor list and all the associated entries from its routing table. Then the node broadcasts a *link failure notification* message. Such a message contains a list of the destinations to which the node lost its best path, 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 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 concerned nodes are notified of the new situation.

If the link failure was discovered due to the failed transmission of a data packet, and there is no other path available for this packet, the node will try to *locally repair the path* (and it will not include this path in the link failure notification message). 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 broadcast 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 delay of the lost path), and if no backward repair ant is received by then, it concludes that it was not possible to repair the path to the destination. Packets which were in the meantime buffered for this destination are discarded, and the node sends a link failure notification about the lost destination.

Link failure notifications keep routing tables on paths up-to-date about upstream link failures. However, they can sometimes get lost and leave dangling links. A data packet following such a link arrives in a node where no further pheromone is available. The node will then discard the data packet and unicast a warning back to the packet's previous hop, which can remove the wrong routing information.

4 Simulation experiments

In a number of simulation experiments we compare AntHocNet's performance to that of AODV [34] (with local repair), a state-of-the-art MANET routing algorithm and a de facto standard. In 4.1 we describe the simulation environment and in 4.2 we present and analyze the results.

4.1. Simulation Environment

As simulation software we use Qualnet [40]. We ran experiments with two different base settings. In the first setting, 100 nodes are randomly placed inside an area of $3000 \times 1000 \text{ m}^2$. Each experiment is run for 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 a two-ray signal propagation model is used. The radio propagation range of the nodes is 300 meters, and the data rate is 2 Mbit/s. At the MAC layer we use the 802.11b DCF protocol. For the different experiments in this setting, we varied the movement patterns of the nodes. We did tests with the *random waypoint* mobility model [25], in which we varied the maximum speed and the pause time, and with the *Gauss-Markov* mobility model [7], in which we again varied the maximum speed. The Gauss-Markov movement scenarios were generated with the BonnMotion software [10]. Parameter values were kept as follows: the update frequency was 2.5, the angle standard deviation 0.4, and the speed standard deviation 0.5.

For the second setting, we used the same setup as was used in the scalability study of AODV performed by Lee, Belding-Royer and Perkins in [28]. In this study, the number of nodes and the size of the simulation area are varied, while keeping the average node density constant (≈ 7.5). The authors do experiments with up to 10000 nodes, but due to computational constraints we limited our tests to maximum 1500 nodes. The exact values used for the number of nodes and the size of the area are given in table 1. Other properties of the simulation setup are kept constant over the different test scenarios. The data traffic consists of 20 CBR sources sending four 512-byte packets per second. The nodes move according to the random waypoint model, with a minimum speed of 0 m/s, a maximum speed of 10 m/s, and a pause time of 30 seconds. The radio propagation range of the nodes is 250 meters, and the channel capacity is 2 Mb/s.

The radio propagation model is a free space model. At the MAC layer the 802.11b DCF model is used. Each simulation is run for 500 seconds.

Table 1. Number of nodes and area sizes for the scalability experiments.

Number of nodes	Area size
100	1500 × 1500
500	3500 × 3500
1000	5000 × 5000
1500	6000 × 6000

For each of the different settings of the parameter values, 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 (3 for the scalability tests of 1000 and 1500 nodes tests due to computational limitations) on each of these 5 problems, to account for stochastic elements both in the algorithms and in the physical and MAC layers.

The choice of the above described scenarios is based on the results obtained for an earlier version of AntHocNet, which are described in [13]. In that paper we investigated the behavior of AntHocNet in the basic scenario used in the influential comparative study of [4]. This scenario is very densely packed, with 50 nodes with a radio range of 300 meters in an area of $1500 \times 300 m^2$. In such an environment, with high interference and very short paths, it is clear that the advantages of maintaining multiple paths, stochastically spreading data, using local repair, etc., do not outweigh their costs. A simple, reactive approach as AODV is expected to be much more effective. In the tests we ran, it became clear that as the environment became more difficult (more mobility, more sparseness, longer paths), the characteristics of AntHocNet became an advantage over those of AODV, resulting in an increasing performance gap in favor of AntHocNet. In this paper we start from a larger and sparser network, and investigate again the effect of increasing the mobility and the size. The study on large networks is necessary to validate the scalability of our approach.

In the following, algorithms are evaluated in terms of *average end-to-end delay* per packet and *delivery ratio* (i.e., the fraction of successfully delivered data packets). These are two important measures of effectiveness for MANET routing algorithms. Apart from that, we also consider the *delay jitter* and the *routing overhead*. Delay jitter measures packet delay variation. It is a metric used in QoS applications and also provides a measure of the stability of the algorithm’s response to changes in the network topology. Delay jitter is calculated as the average of the difference of the interarrival time between subsequently received packets: if the last three packets are received respectively at t_3, t_2, t_1 , the session’s jitter is calculated as the arithmetic average of the values $(t_3 - t_2) - (t_2 - t_1)$ for all triplets of received packets. This definition of jitter was proposed in [36] and is used in Qualnet and in a number of real-world routing devices. The routing overhead measures the algorithm’s internal efficiency and is calculated as the total number of control packets sent divided by the number of data packets delivered successfully.

4.2. Simulation results

We first study the behavior of AntHocNet and AODV in increasingly dynamic environments under the random waypoint mobility model. We use the sparse network of 100 nodes in $3000 \times 1000 m^2$. Node mobility is increased by either increasing the maximum node speed or decreasing the node pause time (the lower the pause time, the higher the node mobility). Figures 2 and 4a show the delivery ratio, average delay and average jitter of AntHocNet and

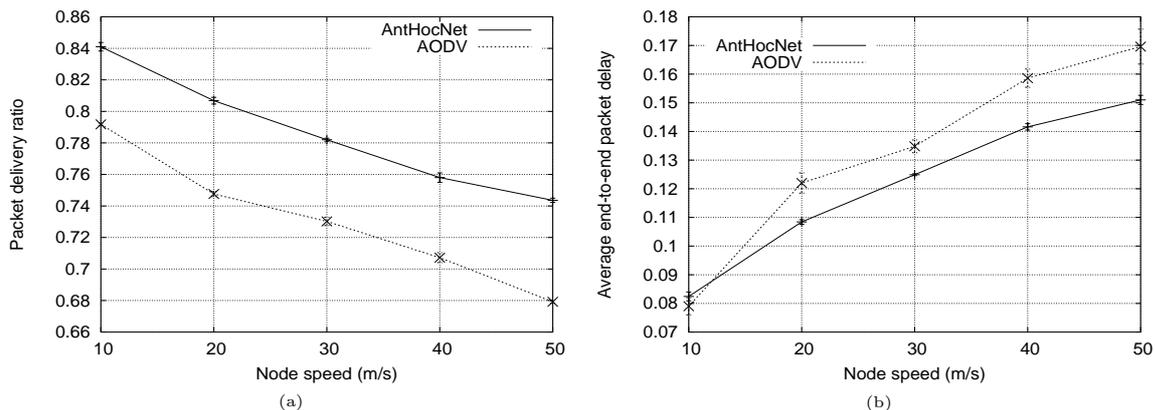


Figure 2. (a) Delivery ratio and (b) average packet delay under various speed values for random waypoint node mobility.

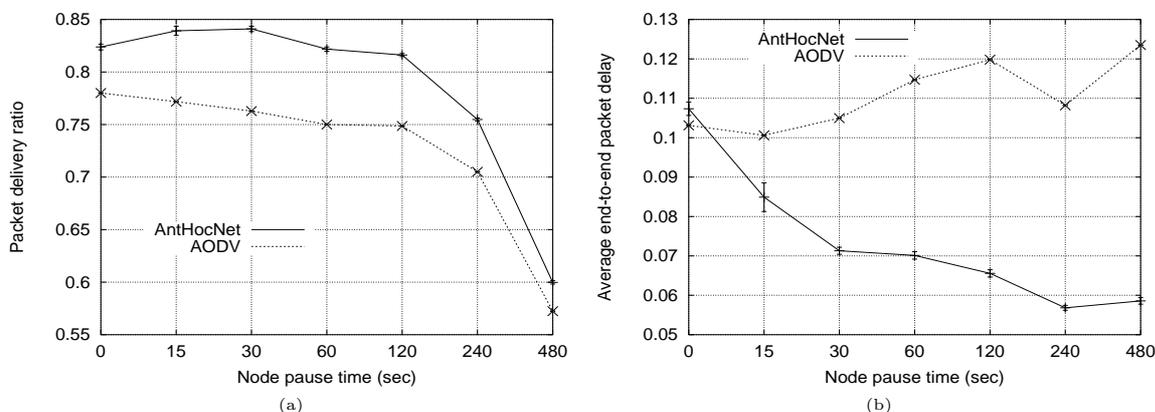


Figure 3. (a) Delivery ratio and (b) average packet delay under various pause times for random waypoint node mobility.

AODV under different node speeds. AntHocNet outperforms AODV clearly for delivery ratio and jitter, and the differences increase for higher speeds. The performance differences for average delay are smaller, but again they increase for higher speeds. Figures 3 and 4b show the same performance measures for both algorithms under different node pause times. AntHocNet again outperforms AODV in terms of delivery ratio, delay and jitter. The relation between mobility and performance is more difficult to establish than for the node speed experiments. Apparently the pause time influences the mobility in a different way than the maximum node speed. Also, the pause time does not only influence the mobility, but also the connectivity: since the network under investigation is sparse, it is possible at high pause times that some nodes remain out of reach of the rest of the network for a long time, and no packets can be delivered to them, resulting in a low delivery ratio. This explains the dip in the delivery ratio and the rise of the jitter for both algorithms.

In order to validate the good results for the random waypoint model, we carried out a similar study with the Gauss-Markov model, where we again increased the maximum node speed. We again use the sparse network scenario of 100 nodes in $3000 \times 1000 m^2$. Figure 5 shows the delivery ratio and average delay for AntHocNet and AODV. Compared to the speed experiments under the random waypoint model, there seem to be two differences: delivery ratios are lower and

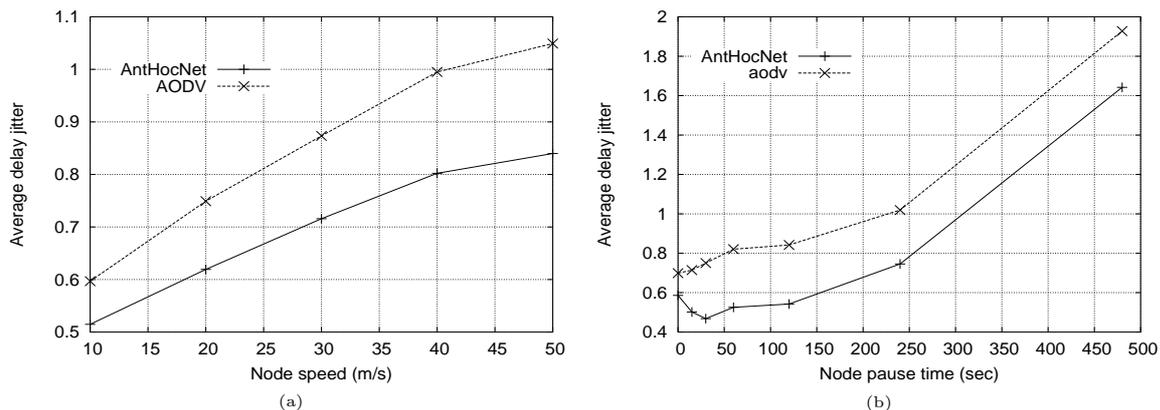


Figure 4. Average delay jitter under (a) various speed values and (b) pause times for random waypoint node mobility.

delays are higher, and the performance differences between AntHocNet and AODV for both measures increase more clearly for higher speeds.

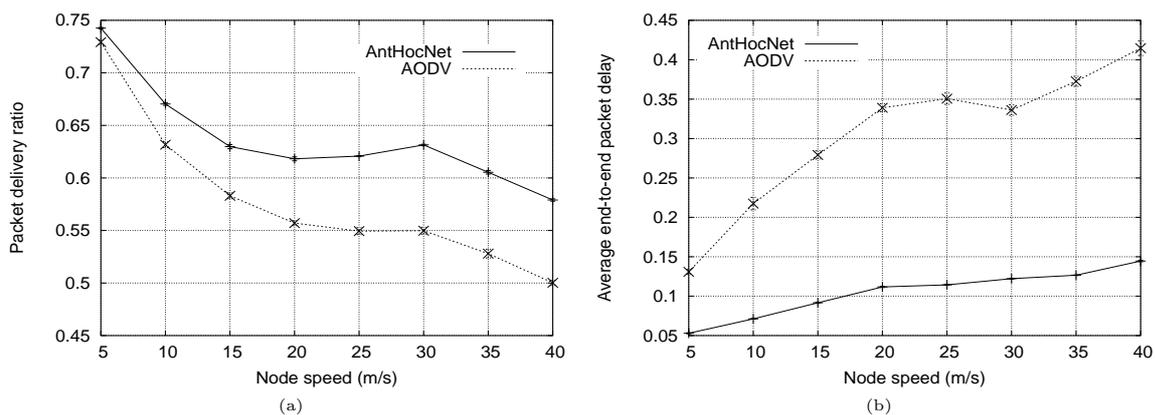


Figure 5. (a) Delivery ratio and (b) average packet delay under various speed values for Gauss-Markov node mobility.

In order to be able to compare results for both mobility models better, we plot the algorithm's performances under both mobility models together in the same graph, against *the average link duration*. The average link duration has been proposed as a measure for the difficulty of a node mobility scenario which is more general than the maximum node speed [39]. The graphs are given in figures 6a and 6b. In these plots, the same observations seem to hold: both algorithms perform better under the random waypoint model for the same average link duration, while under the Gauss-Markov model there is a stronger tendency for the performance advantage of AntHocNet over AODV to grow as the mobility increases. Clearly the differences between the movement patterns generated according to the random waypoint model and the Gauss-Markov model go beyond what can be measured with the average link duration. One difference which might make random waypoint movement patterns easier to deal with, is that nodes tend to cluster together in the middle of the area, resulting in shorter paths (see [3]), something which is maybe not the case for Gauss-Markov models (for this mobility model the average node distribution has not been studied, to the best of our knowledge). Another difference between both models is

that subsequent node movements in the Gauss-Markov model are always correlated, while in the random waypoint model nodes make sudden, uncorrelated changes in direction at the pause points. Possibly an adaptive learning algorithm like AntHocNet can take more advantage out of these correlations than a purely reactive algorithm like AODV, explaining the increasing difference in performance.

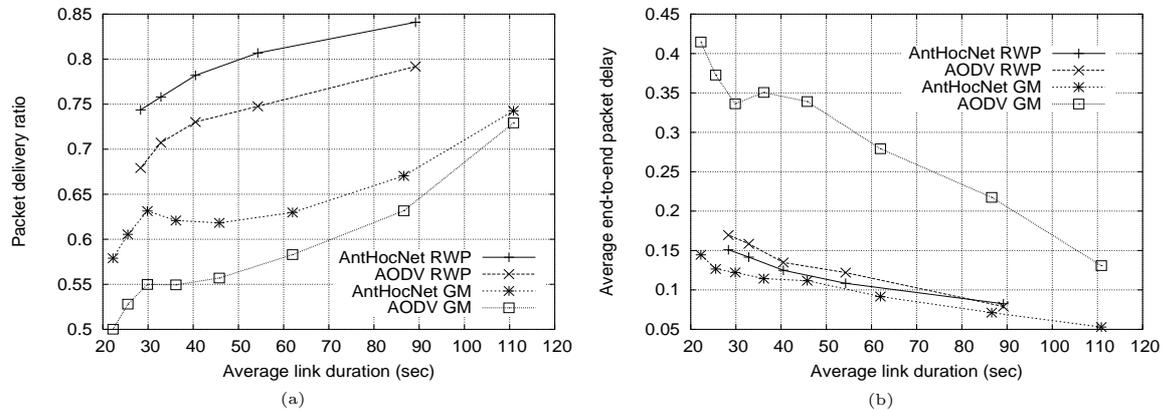


Figure 6. (a) Delivery ratio and (b) average packet delay under different speed values for random waypoint (RWP) and Gauss-Markov (GM) mobility, plotted against average link duration.

The good performances shown above come at a cost though. It is clear from the description of section 3 that AntHocNet uses a lot of different kinds of ant packets in order to adapt to the ever changing MANET environment and be able to provide a high delivery ratio and low delays. Figures 7a and 7b show that AntHocNet generates substantially more control overhead

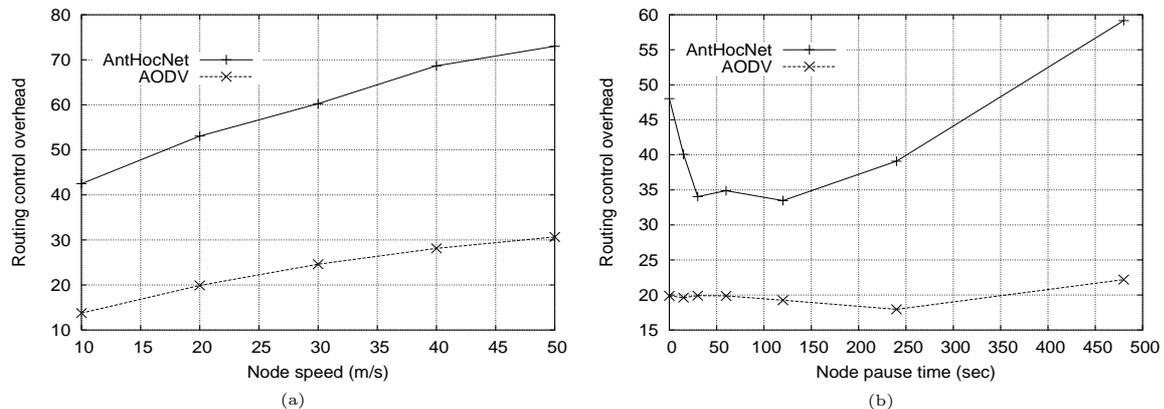


Figure 7. Routing control overhead in number of control packets per successfully delivered data packet under various (a) speed values and (b) pause times for random waypoint node mobility.

(measured in number of control packets per successfully delivered data packet) than AODV. This is clearly an aspect of the algorithm which can be improved. In future work we plan to do this in the first place using the technique of pheromone diffusion mentioned in 3.3: this will allow to limit the blind proliferation of proactive ants so that better results can be obtained with less ants. Another point of improvement can be the behavior of nodes at route setup time:

when a source node fails to establish a connection to the destination, it retries with very short intervals to send reactive forward ants. This can lead to high overhead in case of unreachable nodes. This is clearly visible in figure 7b: for the highest values of the pause time, where some nodes can be cut off from the other nodes for extended periods of time, the overhead is very high.

An important question is how the performance changes as the scale of the problem grows. In order to investigate this, we ran a set of experiments of increasing size, using the second setting described in 4.1. The results are shown in figures 8a and 8b. We can see that again AntHocNet outperforms AODV in terms of delivery ratio and delay, and this difference grows with the scale of the problem. The mechanisms of multipath routing and local repair seem to pay off more when paths are longer. The good performance of the algorithm in these studies gives an indication of its scalability.

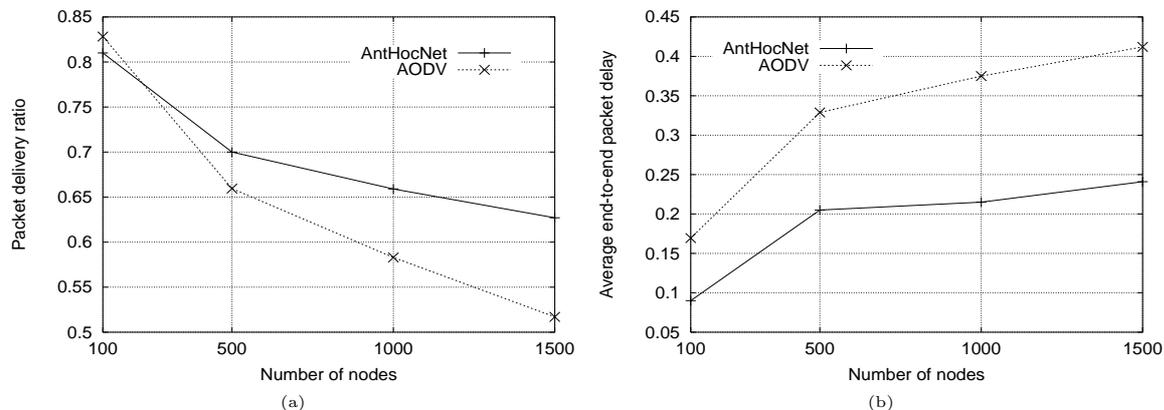


Figure 8. (a) Delivery ratio and (b) average packet delay under increasing network sizes.

5 Conclusions and future work

In this paper we have described AntHocNet, an ACO routing algorithm for MANETs. It is a hybrid algorithm, combining both proactive and reactive elements: after a reactive route setup phase, the algorithm probes, maintains and improves paths in a proactive way. AntHocNet is inspired by the stigmergy-driven shortest path following behavior of biological ants and the related ACO optimization framework. In a series of simulation tests, we show that AntHocNet has a performance advantage over AODV, a reference algorithm in this research area. The advantage exists in terms of packet delivery ratio, average end-to-end delay and average jitter, and increases for larger, sparser and more dynamic environments. However, AntHocNet is less efficient in terms of routing overhead.

Throughout the paper we have indicated some elements which can be improved. These will be the starting points for future work. First of all, we want to improve the behavior of the proactive ants. In 3.3, we proposed the term pheromone diffusion. The idea is to include some limited routing information in hello messages, so that information about existing paths can spread over the network, propagating from node to node in hello messages. This routing information would be too unreliable (due to the slow spreading via subsequent hello messages) for data packets to use, but they could be a good guideline for proactive ants, making their search for new and better paths less blind. The extra information could also be used to regulate the generation rate of proactive ants, which is another important issue to deal with. The improvement of the

proactive ant behavior is also expected to help reduce the overhead created by the algorithm: better guided proactive ants generated in an intelligent and adaptive way could provide better results at lower cost. Other reductions of the overhead could be obtained by improving the generation rate and routing behavior of reactive ants. As was pointed out in 4.2, the retry interval for reactive forward ants in the case of a failed path setup is rather low, which can lead to a lot overhead when a destination is unreachable. Moreover, ants are allowed to spread over the network until a maximum number of hops is reached. This maximum number is quite high at the moment, and setting it more intelligently (e.g. using an expanding ring search like in AODV or using the information from pheromone diffusion) could limit useless and counterproductive propagation.

References

- [1] M. Abolhasan, T. Wysocki, and E. Dutkiewicz. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2:1–22, 2004.
- [2] 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.
- [3] C. Bettstetter and C. Wagner. The spatial node distribution of the random waypoint mobility model. In *Proc. German Workshop on Mobile Ad Hoc Networks (WMAN)*, 2002.
- [4] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom98)*, 1998.
- [5] D. Câmara and A.A.F. Loureiro. Gps/ant-like routing in ad hoc networks. *Telecommunication Systems*, 18(1–3):85–100, 2001.
- [6] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2001.
- [7] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications & Mobile Computing: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2002.
- [8] M.E. Csete and J.C. Doyle. Reverse engineering of biological complexity. *Science*, 295(1):1664–11669, 2002.
- [9] D.S.J. De Couto, D. Aguayo, B.A. Chambers, and R. Morris. Performance of multihop wireless networks: Shortest path is not enough. In *Proceedings of the First Workshop on Hot Topics in Networks (HotNets-I)*. ACM SIGCOMM, 2002.
- [10] Christian de Waal. BonnMotion: A mobility scenario generation and analysis tool, 2002. <http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/>.
- [11] G. Di Caro. *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, 2004.
- [12] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)*, 9:317–365, 1998.
- [13] G. Di Caro, F. Ducatelle, and L.M. Gambardella. AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. In *Proceedings of Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *LNCS*, pages 461–470. Springer-Verlag, 2004.
- [14] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5(2):137–172, 1999.
- [15] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

- [16] J. H. Fewell. Social insect networks. *Science*, 301(26):1867–1869, September 2003.
- [17] 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.
- [18] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1(2), 2002.
- [19] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.
- [20] C. Gui and P. Mohapatra. SHORT: Self-healing and optimizing routing techniques for mobile adhoc networks. In *Proceedings of MobiHoc*, 2003.
- [21] M. Günes, U. Sorges, and I. Bouazizi. ARA - The ant-colony based routing algorithm for MANETs. In *Proceedings of the ICPP International Workshop on Ad Hoc Networks (IWAHN)*, 2002.
- [22] Zygmunt J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications*, 1997.
- [23] M. Heissenbüttel and T. Braun. Ants-based routing in large scale mobile ad-hoc networks. In *Kommunikation in verteilten Systemen (KiVS03)*, March 2003.
- [24] 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.
- [25] D.B. Johnson and D.A. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer, 1996.
- [26] S.-J. Lee and M. Gerla. AODV-BR: Backup routing in ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2000.
- [27] S.-J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of IEEE ICC*, 2001.
- [28] S.-J. Lee, E. M. Royer, and C. E. Perkins. Scalability study of the ad hoc on-demand distance vector routing protocol. *ACM/Wiley International Journal of Network Management*, 13(2):97–114, 2003.
- [29] M. Marina and S. Das. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, pages 14–23, 2001.
- [30] S. Marwaha, C. K. Tham, and D. Srinivasan. Mobile agents based routing protocol for mobile ad hoc networks. In *Proc. of IEEE Globecom*, 2002.
- [31] H. Matsuo and K. Mori. Accelerated ants routing in dynamic networks. In *2nd Int. Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2001.
- [32] 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 *LNCS*. Springer-Verlag, 2004.
- [33] A. Nasipuri, R. Castaneda, and S. R. Das. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *Mobile Networks and Applications*, August 2001.
- [34] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.
- [35] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.
- [36] J. Roberts and F. Guillemin. Jitter in ATM networks and its impact on peak rate enforcement. *Performance Evaluation*, 16(1–3), November 1992.

- [37] M. Roth and S. Wicker. Termite: Emergent ad-hoc networking. In *Proceedings of the Second Mediterranean Workshop on Ad-Hoc Networks*, 2003.
- [38] E.M. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999.
- [39] N. Sadagopan, F. Bai, B. Krishnamachari, and A. Helmy. PATHS: analysis of PATH duration statistics and their impact on reactive MANET routing protocols. In *Proceedings of MobiHoc'03*, pages 245–256, 2003.
- [40] Scalable Network Technologies, Inc., Culver City, CA, USA. *Qualnet Simulator, Version 3.6*, 2003. <http://stargate.ornl.gov/trb/tft.html>.
- [41] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.
- [42] C.-C. Shen, C. Jaikaeo, C. Srisathapornphat, Z. Huang, and S. Rajagopalan. Ad hoc networking with swarm intelligence. In *Ants Algorithms - Proceedings of ANTS 2004, Fourth International Workshop on Ant Algorithms*, LNCS. Springer-Verlag, 2004.
- [43] K.M. Sim and W.H. Sun. Ant colony optimization for routing and load-balancing: Survey and new directions. *IEEE Transactions on Systems, Man, and Cybernetics—Part A*, 33(5):560–572, 2003.
- [44] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [45] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, Special Issue on Stigmergy, 5:97–116, 1999.
- [46] L. Wang, Y.T. Shu, O.W.W. Yang, M. Dong, and L.F. Zhang. Adaptive multipath source routing in wireless ad hoc networks. In *Proc. of the IEEE Int. Conf. on Communications*, 2001.
- [47] K. Wu and J. Harms. On-demand multipath routing for mobile ad hoc networks. In *Proceedings of EPMCC*, 2001.
- [48] Z. Ye, S.V. Krishnamurthy, and S.K. Tripathi. A framework for reliable routing in mobile ad hoc networks. In *Proc. of IEEE INFOCOM*, 2003.