

## *Special Issue on Self-organisation in Mobile Networking*

# 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*

### SUMMARY

In this paper, we describe AntHocNet, an algorithm for routing in mobile ad hoc networks. It is a hybrid algorithm, which combines reactive path setup with proactive path probing, maintenance and improvement. The algorithm is based on the nature-inspired ant colony optimisation 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 the field. 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. Copyright © 2005 AEIT.

### 1. INTRODUCTION

Mobile ad hoc networks (MANETs) [1] are networks in which all nodes are mobile and communicate with each other via wireless connections. Nodes can join or leave at any time. There is no fixed infrastructure, all nodes are equal and there is no centralised 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 limited, and the medium is shared. The access to the shared channel is controlled by protocols at the medium access control (MAC) layer, such as ANSI/IEEE 802.11 DCF [2] (which is 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 *localised way*, due to the lack of central control or infrastructure in the network. *Nature's self-organising* systems like *insect societies* show precisely these desirable properties. Making use of a number of relatively simple biological agents (e.g., ants) a variety of different organised behaviours are generated at the system-level from the local interactions among the agents and with the environment. The robustness and efficiency of the collective behaviours 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

---

\* Correspondence to: Gianni Di Caro, Istituto Dalle Molle sull'Intelligenza Artificiale (IDSIA), Galleria 2, CH-6928 Manno-Lugano, Switzerland.  
E-mail: gianni@idsia.ch

Contract/grant sponsor: Future & Emerging Technologies unit of the European Commission; contract/grant number: IST-2001-38923.  
Contract/grant sponsor: Swiss Hasler Foundation; contract/grant number: DICS-1830.

algorithms for dynamic networks (as well as for the solution of several other classes of problems, e.g. Reference [3]).

In this paper, we describe *AntHocNet*, a new routing algorithm for MANETs. AntHocNet's design is based on a specific self-organising behaviour of ant colonies, the *shortest paths* discovery, and on the related framework of *ant colony optimization* (ACO) [4]. It has been observed that ants in a colony can converge on moving over the shortest among different paths connecting their nest to a food source [5, 4]. The main catalyst of this colony-level shortest path behaviour is the use of a volatile chemical substance called *pheromone*: ants moving between the nest and a food source deposit pheromone, and preferentially move towards 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 then 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 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* [6]. Stigmergic coordination is one of the keys to obtain self-organised behaviours not only in ant colonies but more generally across *social systems*, from insects to humans (e.g. [7, 8]). When stigmergy is at work, system's *protocols* (interfaces) play a prominent role with respect to *modules* (agents) [9]. 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 behaviour to define a nature-inspired metaheuristic for combinatorial optimisation. ACO features multi-agent organisation, 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, vehi-

cle routing etc., see [4, 3] for overviews), as well as to routing (e.g. [10–12]). The first ACO routing algorithms were designed for wired networks (e.g. *AntNet* [10] for packet-switched networks and *ABC* [11] for circuit-switched networks). These algorithms exhibit interesting properties which are also desirable for MANET routing: they work in a fully distributed way, are highly adaptive, 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 [13] and PERA [14]. However, these algorithms lose much of the proactive sampling and exploratory behaviour 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 works efficiently in MANETs while maintaining the properties which make ACO routing algorithms so appealing. While most of the previous algorithms for wired networks were 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.

This paper is organised 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 Subsection 2.1, we give an introduction to MANET routing algorithms, and in Subsection 2.2 we describe the basic elements of ACO for routing. Then in Subsection 2.3, we give an overview of existing implementations of ACO routing for MANETs, and in Subsection 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 Reference [15] for an overview). These algorithms deal with the dynamic aspects of MANETs in their own way, using reactive or proactive behaviour, or a combination of both. *Reactive behaviour*

means that an algorithm only gathers routing information in response to an event, usually an event which triggers the need for new paths, such as the start of a data session or the failure of an existing paths. *Proactive behaviour* means that the algorithm also gathers routing information at other times, so that it is readily available when needed.

In the MANET literature, the classical distinction is between purely proactive, purely reactive and hybrid algorithms. In *purely proactive* algorithms (e.g. DSDV [16]), nodes try to maintain paths to all other nodes at all times. 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 [17] and DSR [18]), nodes only gather routing information on demand: when a data session to a new destination starts, or when a path which is in use fails. Reactive algorithms are in general more scalable [19] 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 [20]), 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 [21, 4] is the acquisition of routing information through the sampling of paths using small control packets, which are called *ants*. The ants are generated concurrently and independently at the nodes, with the task to test a path from a source node  $s$  to an assigned destination node  $d$ . The ant collects information about the quality of its 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 and at  $s$ . 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 next [22].

The routing tables contain for each destination a vector of real-valued entries, one for each known neighbour node. These entries are a measure of goodness of going over that neighbour on the way to a certain 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 links with higher pheromone values. In the following, we also call routing tables *pheromone tables*.

This process is quite similar to the pheromone laying and following behaviour 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 behaviour*, in which individual ants have low complexity and little importance, while the whole swarm together can collect and maintain up-to-date routing information.

The pheromone information is used for routing data packets, more or less in the same way as for routing ants: packets are routed *stochastically*, giving higher probability to links with higher pheromone values. Like this, 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. This way *path exploration* is kept separate from the use of paths by data. If enough ants are sent to the different destinations, nodes have up-to-date information about the best paths and automatically adapt their data load spreading.

## 2.3. ACO routing in MANETs

The description of Subsection 2.2 highlights a number of key ingredients of ACO routing: routing tables are adapted and maintained via repeated and concurrent Monte Carlo path sampling; data are stochastically spread over multiple paths, leading to automatic load balancing; routing and control decisions are taken locally and the system is robust to agent failures. Some attempts have been made to incorporate these features into a MANET routing algorithm. 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* [23] uses ant-like agents which go through the network randomly, without a specific destination, updating pheromone entries pointing to their source. In Reference [24], 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* [25] is a hybrid algorithm combining ants with the basic AODV behaviour: a fixed number of ants keep going around the

network in a more or less random manner, proactively updating the AODV routing tables in the nodes they visit whenever possible. *ant-colony-based routing algorithm* (ARA) [13] works 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) [14] 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 [26, 27] 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 Reference [28] for an overview). The algorithms differ in the way multiple paths are set up, maintained and used. At path setup time, a number of paths are selected. Some algorithms allow braided multiple paths [29], whereas others look for link [30] or node [31] disjoint paths, or even paths which are outside each other's interference range [32]. 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 [29, 33]. *Proactively improving existing paths* is quite rare in MANET routing algorithms, although one possible approach is presented in Reference [34] (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 [35, 36]. Some algorithms spread data over the multiple paths in a

simple, even way [37], and in a few cases *adaptive data load spreading* depending on the estimated quality of paths, similar to the ACO ideas, is explored [29, 33]. 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 [38]. *Stochastic data spreading* is according to our knowledge unexplored outside the area of ACO routing algorithms (although stochastic elements have been used otherwise in MANET algorithms, for example to improve flooding [39]).

### 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 paths to all destinations at all times (like the ACO algorithms for wired networks), but sets up paths when they are needed at the start of a session. This is done in a *reactive path 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 set up the paths. The paths are represented in pheromone tables indicating their respective quality. After path setup, *data packets are routed stochastically* as datagrams over the different paths using these pheromone tables. While a data session is going on, the *paths are probed, maintained and improved proactively* using different agents, called *proactive forward ants*. The algorithm reacts to *link failures* with either local path repair or by warning preceding nodes on the paths. An earlier version of the algorithm described here appeared in Reference [40].

#### 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 neighbour of  $s$  receives a replica of  $F_d^s$ . We refer to the set of replicas which originated from the same original ant as an *ant generation*. The task of each ant of the generation 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 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 the table is the pheromone value indicating the estimated goodness of going from  $i$  over neighbour  $n$  to

reach destination  $d$ . If pheromone information is available, the ant chooses its next hop  $n$  with probability  $P_{nd}$ :

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

where  $\mathcal{N}_d^i$  is the set of neighbours of  $i$  over which a path to  $d$  is known, and  $\beta$  is a parameter value which can control the exploratory behaviour of the ants (although in current experiments  $\beta$  is kept to 1).

If no pheromone 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 killed). When a node receives several ants of the same generation, it compares 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. 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, so they have higher chances of being killed. Duplicate ants which result from a broadcast of the best ant just before it reaches the destination are close in performance to the best ant and have higher chances of being accepted. The result is a set of ‘kite-shaped’ paths, as shown by the solid line arrows in Figure 1. In order to obtain a mesh of sufficiently disjoint multiple paths, 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  $a_2$  (in the experiments  $a_2$  was set to 2 as opposed to  $a_1 = 0.9$ ). A similar strategy is used in Reference [30]. The result is a uniformly

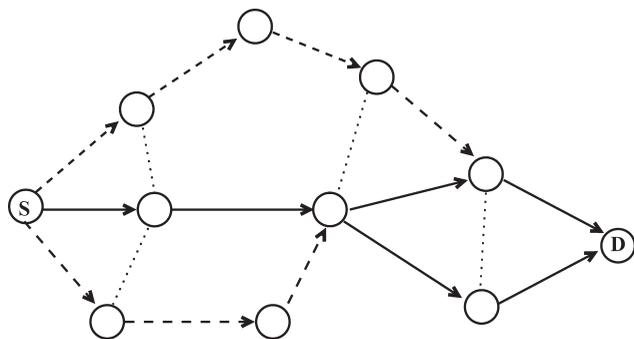


Figure 1. Example of ‘kite-shaped’ and meshed multiple paths.

spread set of paths, as shown by the combination of solid and dashed line arrows in Figure 1.

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  $i + 1$ :

$$\hat{T}_{\mathcal{P}} = \sum_{i=1}^{n-1} \hat{T}_{i+1}^i. \quad (2)$$

$\hat{T}_{i+1}^i$  is defined as the product of the estimate of the average time to send one packet,  $\hat{T}_{\text{mac}}^i$ , times the current number of packets in queue (plus one) at the MAC layer,  $Q_{\text{mac}}^i$ :

$$\hat{T}_{i+1}^i = (Q_{\text{mac}}^i + 1) \hat{T}_{\text{mac}}^i, \quad (3)$$

$\hat{T}_{\text{mac}}^i$  is calculated as running average of time elapsed between the arrival of a packet at the MAC layer and the end of a successful transmission. So if  $t_{\text{mac}}^i$  is the time it took to send a packet from node  $i$ , then node  $i$  updates its estimate as:

$$\hat{T}_{\text{mac}}^i = \alpha \hat{T}_{\text{mac}}^i + (1 - \alpha) t_{\text{mac}}^i, \quad (4)$$

with  $\alpha \in [0, 1]$ . Since  $\hat{T}_{\text{mac}}^i$  is calculated at the MAC layer it includes channel access activities, so it accounts for local congestion of the shared medium. Forward ants calculate a similar time estimate  $\hat{T}_{\mathcal{P}}$ , which is used for filtering the ants, as mentioned before.

At each node  $i \in \mathcal{P}$ , the backward ant 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  $\tau_d^i$  used to update the running average is defined as:

$$\tau_d^i = \left( \frac{\hat{T}_d^i + hT_{\text{hop}}}{2} \right)^{-1}, \quad (5)$$

where  $T_{\text{hop}}$  is a parameter (set to  $3 \times 10^{-3}$  s) representing the time to take one hop in unloaded conditions. Defining  $\tau_d^i$  like this is a way to avoid 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:

$$\mathcal{T}_{nd}^i = \gamma \mathcal{T}_{nd}^i + (1 - \gamma) \tau_d^i, \quad \gamma \in [0, 1]. \quad (6)$$

$\gamma$  and  $\alpha$  (Equation 4) were both set to 0.7 in the experiments.

If the path setup process is successful, a number of good paths between source and destination are made available. If, on the other hand, no backward ant has come back to the source after a certain amount of time (in the experiments set to 1 s), data are temporarily buffered and the whole process is restarted. This is repeated for a maximum number of times (set to 3), after which the buffered data are discarded.

### 3.2. Stochastic data routing

Nodes in AntHocNet forward data *stochastically*. When a node has multiple next hops for the destination  $d$  of the data, it randomly selects one of them with probability  $P_{nd}$ .  $P_{nd}$  is calculated in the same way as for reactive forward ants (Equation 1), but with a higher  $\beta$  exponent (set to 2), 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 is selected automatically 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 probing, 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, where  $n$  was 5 in the experiments). They are normally unicast, choosing the next hop according to the pheromone values using the same formula as reactive forward ants (Equation 1), but also have a small probability at each node of being broadcast (this probability was set to 0.1 in the experiments). This way they serve

two purposes. If a forward ant reaches the destination without a single broadcast it *probes* an existing path. It gathers up-to-date quality estimates of this path, and the backward ant updates the pheromone values of intermediate nodes, just like reactive backward ants do. If, on the other hand, the ant got broadcast at any point, it leaves the currently known paths and *explores* new ones.

After a broadcast, the ant arrives in all neighbours of the broadcasting node. It is possible that in these neighbours it does not find pheromone for its destination, so that it needs to be broadcast again. The ant will then quickly proliferate and flood the network, like reactive forward ants do. To avoid this, we limit the number of broadcasts to  $n_b$  (set to 2 in the tests). If the proactive ant does not find routing information within  $n_b$  hops, it is killed. The effect of this is that the search for new paths is concentrated around the current paths, so that we are looking for *path improvements and variations*.

To guide the forward ants better, we use *hello messages*. These are short messages (in our case containing just the sender's address) broadcast every  $t_{\text{hello}}$  seconds by the nodes (e.g.  $t_{\text{hello}} = 1$  sec). If a node receives a hello from a new node  $n$ , it adds  $n$  in its routing table. After that it expects a hello from  $n$  every  $t_{\text{hello}}$  seconds. After missing a certain number of hello's (*allowed-hello-loss* = 2 here),  $n$  is removed. Using these messages, nodes have pheromone information about their immediate neighbours in their routing table. So when an ant arrives in a neighbour of its 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 proactive ants. Hello messages also serve another purpose: they allow to detect broken links. This allows nodes to clean up stale entries from their routing tables.

### 3.4. Link failures

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

When a neighbour is assumed to have disappeared, the node takes a number of actions. First, it removes the neighbour from its neighbour list and all associated entries from its routing table. Then it broadcasts a *link failure notification* message. This message contains a list of 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 neighbours receive the notification and update their pheromone using the new estimates. If they in turn lost their best or their only path to a destination due to the failure, they also broadcast a notification, until all concerned nodes are notified.

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 tries to *locally repair the path* (and does not include this path in the link failure notification). The node broadcasts a *path repair ant* that travels to the involved destination like a reactive forward ant: it follows available pheromone when it can and is broadcast otherwise. One difference is that it has a maximum number of broadcasts (two in our tests) so that proliferation is limited. The node waits for some time (empirically set to five times the estimated delay of the lost path), and if no backward repair ant is received, it concludes that it was not possible to repair the path. 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 range of simulation experiments, we compare AntHocNet to AODV [17] (with local repair), a state-of-the-art MANET routing algorithm and de facto standard. In Subsection 4.1, we describe the simulation environment, and in Subsection 4.2 we present and analyse the results.

### 4.1. Simulation environment

As simulation software we use QualNet [41]. We ran experiments with two different base settings. In the first

setting, 100 nodes are randomly placed in an area of  $3000 \times 1000 \text{ m}^2$ . Each experiment is run for 900 s. 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 s after the start of the simulation, and keeps sending until the end. A two-ray pathloss model is used in the radio propagation model. The radio range of the nodes is 300 m, and the data rate is 2 Mbit/s. At the MAC layer, we use the 802.11b DCF protocol as is common practice in MANET research. We did tests with the *random waypoint* (RWP) mobility model [18], in which we varied the maximum speed and the pause time, and with the *Gauss–Markov* (GM) mobility model [42], in which we again varied the maximum speed. The update frequency was set to 2.5, the angle standard deviation to 0.4 and the speed standard deviation to 0.5. The GM movement scenarios were generated with the BonnMotion software [43].

For the second setting, we used the same setup as in the scalability study of AODV performed by Lee, Belding-Royer and Perkins in Reference [44]. In this paper, the number of nodes and the size of the simulation area are varied, while keeping the average node density constant ( $\approx 7.5$ ). We did experiments with 100, 500, 1000 and 1500 nodes in square areas with sides of, respectively, 1500, 3500, 5000 and 6000 m. In Reference [44], the experiments go up to 10000 nodes, but we had to limit our tests due to computational constraints. Data traffic consists of 20 CBR sources sending four 512-byte packets per second. Nodes move according to the RWP model, with a minimum speed of 0 m/s, a maximum speed of 10 m/s and a pause time of 30 s. The radio propagation range of the nodes is 250 m, and the data rate is 2 Mbit/s. The pathloss model is a free space model. At the MAC layer, the 802.11b DCF model is used. Each simulation is run for 500 s.

For each of the settings of the parameter values, five different problems were created, by choosing different initial placements of the nodes and different movement patterns. The reported results are averaged over five different runs (three for the scalability tests of 1000 and 1500 nodes due to computational limitations) on each of these five problems, to account for stochastic elements both in the algorithms and in the physical and MAC layers.

The choice of the described scenarios is based on results obtained for an earlier version of AntHocNet, described in Reference [40]. In that paper, we investigated the behaviour of AntHocNet in scenarios based on the influential comparative study of [19]. The considered base scenario was very densely packed, with 50 nodes with a 300 m radio

range in an area of  $1500 \times 300 \text{ m}^2$ . In such an environment, with high interference and short paths (the average path length is about 2.5 hops), the advantages of maintaining multiple paths, stochastically spreading data, using local repair etc., might not outweigh their costs. A simple, reactive approach as AODV is expected to be equally effective. In our tests the performances of AntHocNet and AODV were comparable, but when the environment became more difficult (more mobility, more sparseness, longer paths), there was an increasing performance gap in favour of AntHocNet. In this paper, we start from a larger and sparser network, and investigate the effect of increasing mobility and size. The study on large networks is necessary to validate the scalability of our approach. The combined sets of tests reported here and in Reference [40] cover a wide range of possible MANET scenarios.

In the following, algorithms are evaluated in terms of *average end-to-end delay* per packet and *delivery ratio* (the fraction of successfully delivered data packets), which are two important measures of routing effectiveness. We also consider *delay jitter* and *routing overhead*. Delay jitter measures packet delay variation. It is calculated as the average of the difference of interarrival time between subsequently received packets: the session's jitter is the arithmetic average of the values  $(t_3 - t_2) - (t_2 - t_1)$  for all triplets of subsequently received packets, where  $t_1$  is arrival time of the first packet and  $t_3$  of the last. Routing overhead measures the algorithm's 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 behaviour of AntHocNet and AODV in increasingly dynamic environments under RWP mobility. We use the sparse scenario of 100 nodes in  $3000 \times 1000 \text{ m}^2$ . 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 2–4 show the delivery ratio, average delay and average jitter of AntHocNet and AODV under different node speeds. AntHocNet outperforms AODV clearly for delivery ratio and jitter, and the differences increase for higher speeds. Performance differences for average delay are smaller, but again they increase for higher speeds. Figures 5–7 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

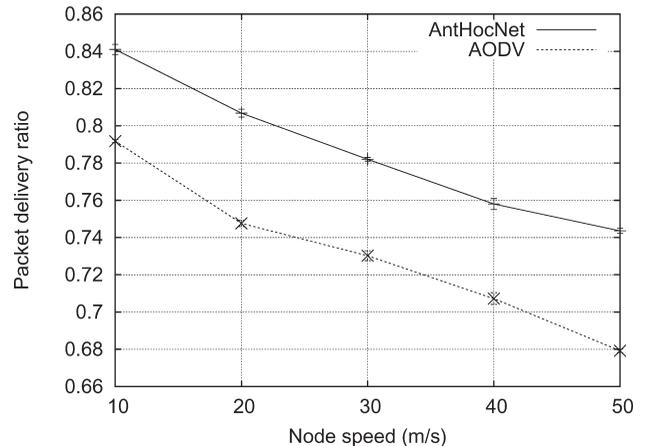


Figure 2. Delivery ratio under various speed values for RWP mobility.

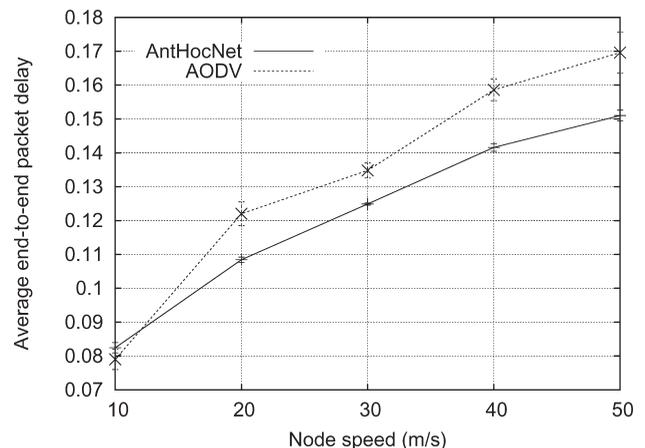


Figure 3. Average packet delay under various speed values for RWP mobility.

the node speed experiments. Apparently the pause time influences mobility in a different way than the maximum node speed. Also, the pause time does not only influence mobility, but also 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 delivery ratio and the rise of jitter for both algorithms.

In order to validate the good results for RWP mobility, we carried out a similar study with GM mobility, where we again increased the maximum node speed. We again use the sparse network scenario of 100 nodes in  $3000 \times 1000 \text{ m}^2$ . Figures 8 and 9 show the delivery ratio

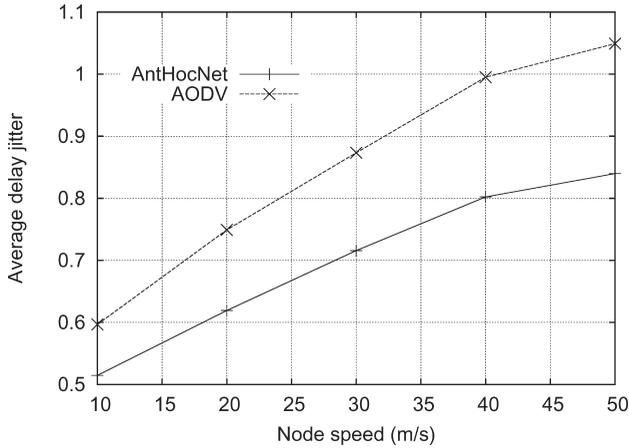


Figure 4. Average delay jitter under various speed values for RWP mobility.

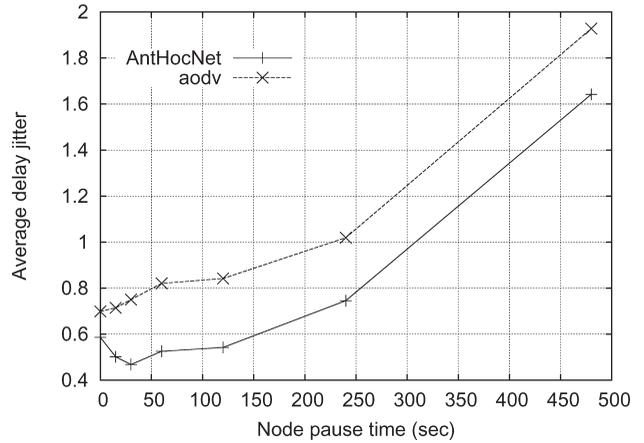


Figure 7. Average delay jitter under various pause times for RWP mobility.

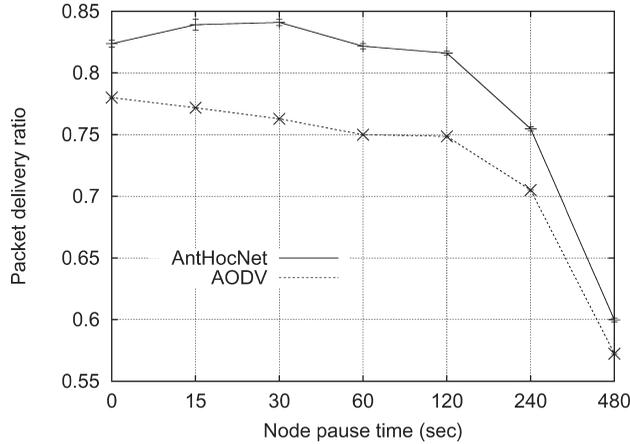


Figure 5. Delivery ratio under various pause times for RWP mobility.

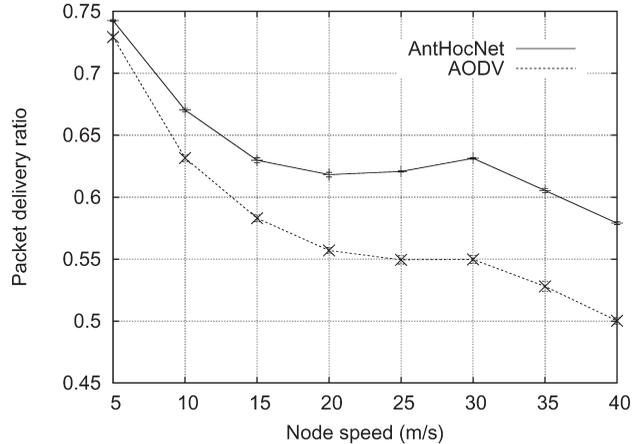


Figure 8. Delivery ratio under various speed values for GM mobility.

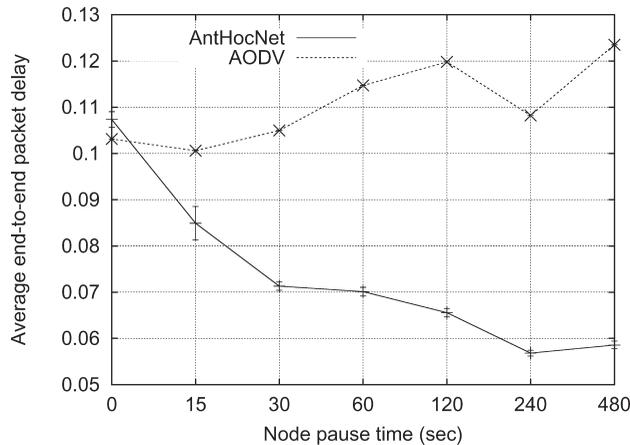


Figure 6. Average packet delay under various pause times for RWP mobility.

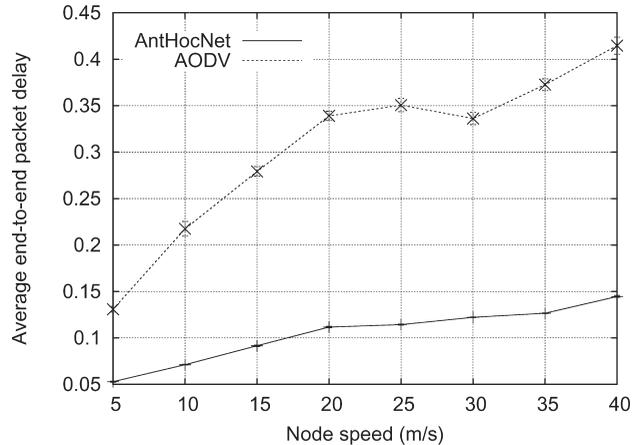


Figure 9. Average packet delay under various speed values for GM mobility.

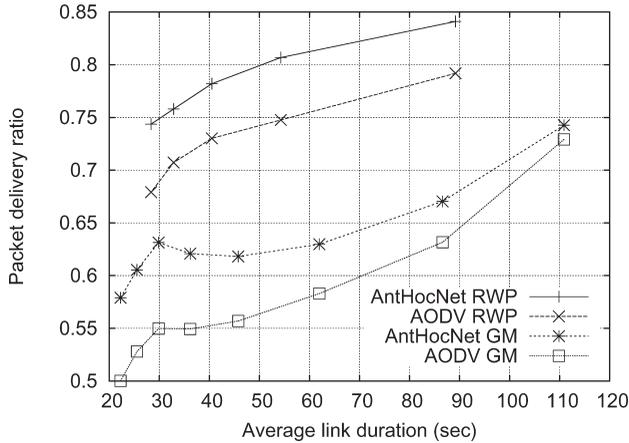


Figure 10. Delivery ratio under various speed values for RWP and GM mobility, plotted against average link duration.

and average delay for AntHocNet and AODV. Compared to the speed experiments under RWP mobility, there are two differences: delivery ratios are lower and delays are higher, and the performance differences between AntHocNet and AODV for both measures increase more clearly for higher speeds.

In order to compare the results for both mobility models better, we plot the algorithms' performances under both mobility models together in the same graph, against the *average link duration*. 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 [45]. The graphs are given in Figures 10 and 11. The previous observations seem to hold: both algorithms

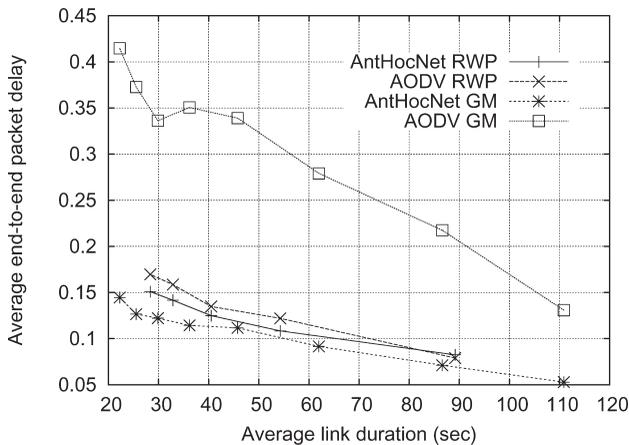


Figure 11. Average packet delay under various speed values for RWP and GM mobility, plotted against average link duration.

perform better under RWP mobility for the same average link duration, while under GM mobility the performance advantage of AntHocNet over AODV grows stronger as the mobility increases. Clearly the differences between the movement patterns generated according to the RWP model and the GM model go beyond what can be measured with average link duration. One difference which might make RWP movement patterns easier to deal with, is that nodes tend to cluster together in the middle of the area, resulting in shorter paths (see reference [46]), something which is not the case for GM patterns. Another difference between both models is that subsequent node movements in the GM model are always correlated, while in the RWP model nodes make sudden, uncorrelated changes in direction at the pause points. Possibly an adaptive learning algorithm like AntHocNet can take more advantage of these correlations than a purely reactive algorithm like AODV, explaining the increasing difference in performance.

The good performances shown above come at a cost though. AntHocNet uses a lot of different kinds of ants to adapt to the changing environment and be able to provide a high delivery ratio and low delays. Figures 12 and 13 show that AntHocNet generates substantially more control overhead 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 pheromone diffusion mentioned in Subsection 3.3: this will allow to limit the blind proliferation of proactive ants so that better results can be obtained with less ants. A different point worth mentioning here is the behaviour of nodes at path setup time: when a source fails to establish a connection to its

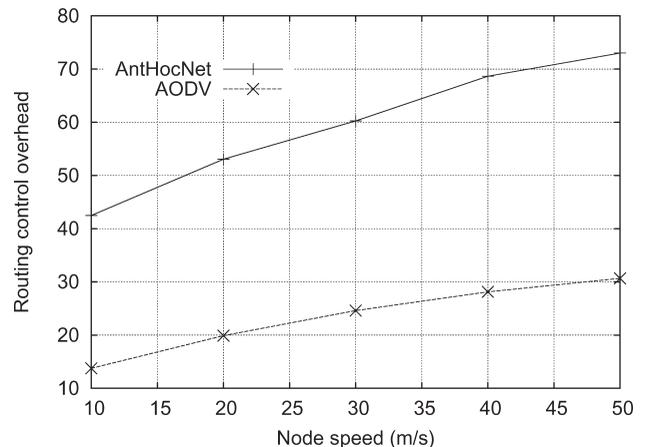


Figure 12. Routing control overhead in number of control packets per successfully delivered data packet under various speed values for RWP mobility.

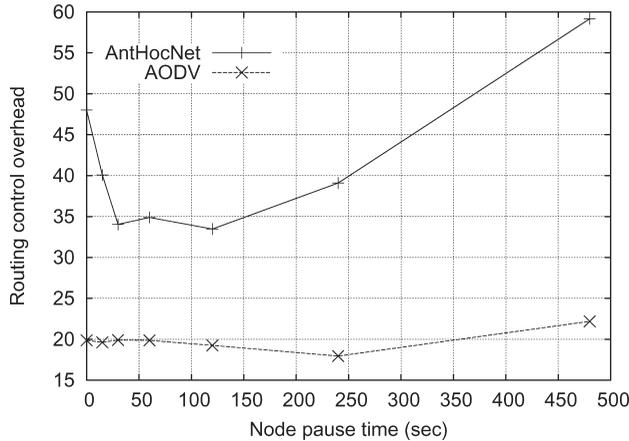


Figure 13. Routing control overhead in number of control packets per successfully delivered data packet under various pause times for RWP mobility.

destination, it retries with short intervals to send reactive forward ants. This can lead to high overhead in case of unreachable nodes, which is clearly visible in Figure 13: 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 Subsection 4.1. The results are shown in Figures 14 and 15. 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.

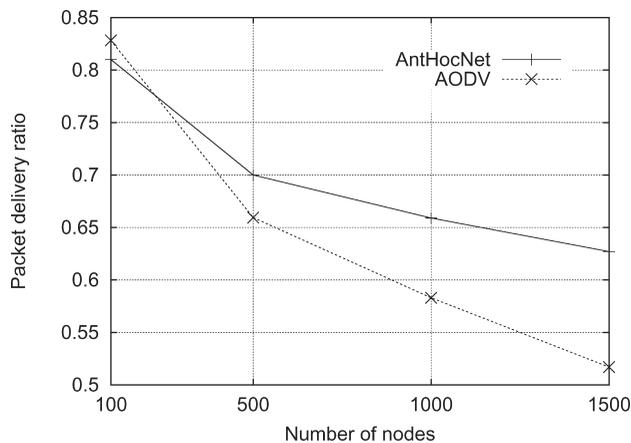


Figure 14. Delivery ratio under increasing network sizes.

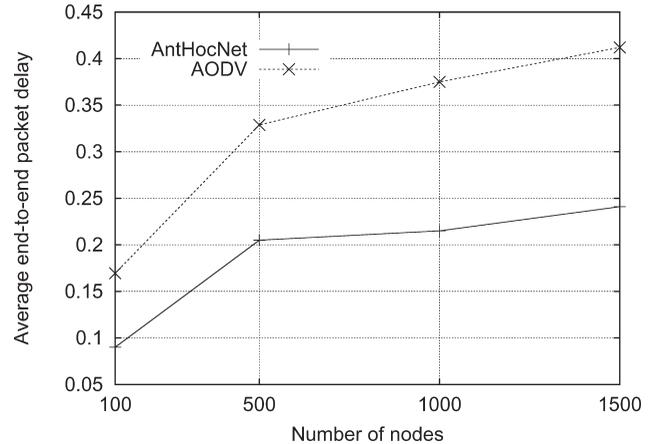


Figure 15. Average packet delay under increasing network sizes.

## 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 path setup phase, the algorithm probes, maintains and improves path in a proactive way. AntHocNet is inspired by the stigmergy-driven shortest paths following behaviour of ant colonies and the related ACO optimisation 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.

To further improve the working of AntHocNet, we consider the behaviour of the proactive ants as a crucial point. In future work we want to investigate the use of pheromone diffusion, which we proposed in Subsection 3.3. 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 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 behaviour is 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 behaviour of reactive ants.

#### ACKNOWLEDGEMENT

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 Net-works” (IST-2001-38923) and by the Swiss Hasler Foundation through grant DICS-1830.

#### REFERENCES

- Royer EM, Toh C-K. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999; **4**(2):46–55.
- 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.
- Dorigo M, Stützle T. *Ant Colony Optimization*. MIT Press: Cambridge, MA, 2004.
- Dorigo M, Di Caro G, Gambardella LM. Ant algorithms for discrete optimization. *Artificial Life* 1999; **5**(2):137–172.
- Goss S, Aron S, Deneubourg JL, Pasteels JM. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften* 1989; **76**:579–581.
- Theraulaz G, Bonabeau E. A brief history of stigmergy. *Artificial Life*, Special Issue on Stigmergy, 1999; **5**:97–116.
- Camazine S, Deneubourg J-L, Franks NR, Sneyd J, Theraulaz G, Bonabeau E. *Self-Organization in Biological Systems*. Princeton University Press: Princeton, NJ, 2001.
- Fewell JH. Social insect networks. *Science* 2003; **301**(26):1867–1870.
- Csete ME, Doyle JC. Reverse engineering of biological complexity. *Science* 2002; **295**(1):1664–1669.
- Di Caro G, Dorigo M. AntNet: distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research (JAIR)* 1998; **9**:317–365.
- Schoonderwoerd R, Holland O, Bruten J, Rothkrantz L. Ant-based load balancing in telecommunications networks. *Adaptive Behavior* 1996; **5**(2):169–207.
- Sim KM, Sun WH. Ant colony optimization for routing and load-balancing: survey and new directions. *IEEE Transactions on Systems, Man, and Cybernetics—Part A* 2003; **33**(5):560–572.
- Günes M, Sorges U, Bouazizi I. ARA—the ant-colony based routing algorithm for MANETs. In *Proceedings of the ICPP International Workshop on Ad Hoc Networks (IWAHN)*, 2002.
- Baras JS, Mehta H. A probabilistic emergent routing algorithm for mobile ad hoc networks. In *Proceedings of WiOpt03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.
- Abolhasan M, Wysocki T, Dutkiewicz E. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks* 2004; **2**:1–22.
- Perkins C, Bhagwat P. Highly dynamic destination-sequenced distance-vector (DSDV) for mobile computers. *ACM SIGCOMM’94: Computer Communications Review* 1994; **24**(4):234–244.
- Perkins CE, Royer EM. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.
- Johnson DB, Maltz DA. *Mobile Computing*, Chapter Dynamic Source Routing in Ad Hoc Wireless Networks. Kluwer: Norwell, MA, 1996; 153–181.
- Broch J, Maltz DA, Johnson DB, Hu Y-C, Jetcheva J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom98)*, 1998.
- Haas ZJ. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the IEEE International Conference on Universal Personal Communications*, 1997.
- Di Caro G. *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. PhD thesis. Université Libre de Bruxelles: Brussels, Belgium, 2004.
- Sutton RS, Barto AG. *Reinforcement Learning: An Introduction*. MIT Press: Cambridge, MA, 1998.
- Fujita K, Saito A, Matsui T, Matsuo H. An adaptive ant-based routing algorithm used routing history in dynamic networks. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, 2002.
- Câmara D, Loureiro AAF. GPS/ant-like routing in ad hoc networks. *Telecommunication Systems* 2001; **18**(1–3):85–100.
- Marwaha S, Tham CK, Srinivasan D. Mobile agents based routing protocol for mobile ad hoc networks. In *Proceedings of the IEEE Global Communications Conference (GlobeCom)*, 2002.
- Roth M, Wicker S. Termite: emergent ad-hoc networking. In *Proceedings of the Second Mediterranean Workshop on Ad-Hoc Networks*, 2003.
- Shen C-C, Jaikao C, Srisathapornphat C, Huang Z, Rajagopalan S. Ad hoc networking with swarm intelligence. In *Ants Algorithms—Proceedings of ANTS 2004, Fourth International Workshop on Ant Algorithms*, volume 3172 of LNCS. Springer-Verlag, 2004.
- Mueller S, Tsang R, Ghosal D. 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.
- Ganesan D, Govindan R, Shenker S, Estrin D. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review* 2001; **5**(4):11–25.
- Marina M, Das S. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, 2001; pp. 14–23.
- Ye Z, Krishnamurthy SV, Tripathi SK. A framework for reliable routing in mobile ad hoc networks. In *Proceedings of IEEE INFOCOM*, 2003.
- Wu K, Harms J. On-demand multipath routing for mobile ad hoc networks. In *Proceedings of the European Personal and Mobile Communications Conference (EPMCC)*, 2001.
- Wang L, Shu YT, Yang OWW, Dong M, Zhang LF. Adaptive multipath source routing in wireless ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2001.
- Gui C, Mohapatra P. SHORT: self-healing and optimizing routing techniques for mobile ad hoc networks. In *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.
- Lee S-J, Gerla M. AODV-BR: backup routing in ad hoc networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2000.
- Nasipuri A, Castaneda R, Das SR. Performance of multipath routing for on-demand protocols in mobile ad hoc networks. *Mobile Networks and Applications (MONET)* 2001; **6**(4):339–349.
- Lee S-J, Gerla M. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of the IEEE International Conference on Communications (ICC)*, 2001.
- De Couto DSJ, Aguayo D, Chambers BA, Morris R. 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.

39. Sasson Y, Cavin D, Schiper A. Probabilistic broadcast for flooding in wireless mobile ad hoc networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, 2003.
40. Di Caro G, Ducatelle F, Gambardella LM. 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*, Springer-Verlag, 2004; pp. 461–470.
41. Scalable Network Technologies, Inc., Culver City, CA, USA. *Qualnet Simulator, Version 3.6*, 2003. [www.scalable-networks.com](http://www.scalable-networks.com)
42. Camp T, Boleng J, Davies V. 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; **2**(5):483–502.
43. de Waal C. BonnMotion: a mobility scenario generation and analysis tool, 2002. <http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/>.
44. Lee S-J, Royer EM, Perkins CE. Scalability study of the ad hoc on-demand distance vector routing protocol. *ACM/Wiley International Journal of Network Management* 2003; **13**(2):97–114.
45. Sadagopan N, Bai F, Krishnamachari B, Helmy A. PATHS: analysis of PATH duration statistics and their impact on reactive MANET routing protocols. In *Proceedings of MobiHoc*, 2003.
46. Bettstetter C, Wagner C. The spatial node distribution of the random waypoint mobility model. In *Proceedings of the German Workshop on Mobile Ad Hoc Networks (WMAN)*, 2002.

## AUTHORS' BIOGRAPHIES

**Gianni Di Caro** received his 'Laurea' degree in Physics with full honors from the Università di Bologna, Italy, in 1992 and his Ph.D. in Applied Sciences from the Université Libre de Bruxelles, Belgium in 2004. He has worked in several leading scientific laboratories worldwide. His interests are in adaptive network routing, Ant Colony Optimisation and other metaheuristics for combinatorial optimisation, reinforcement learning, immune system, autonomous robotics and parallel computing. He is currently at IDSIA, Switzerland, where he works on biologically-inspired algorithms for routing in mobile ad hoc networks.

**Frederick Ducatelle** obtained his M.Sc. in Commercial Engineering in Management Informatics from the Katholieke Universiteit Leuven in Belgium in 2000 and his M.Sc. in Artificial Intelligence from the University of Edinburgh in Scotland in 2001. He is currently a Ph.D. student at the IDSIA in Lugano, Switzerland, where his work is mainly focused on ant colony Optimisation and its application to network problems.

**Luca Maria Gambardella** is the director at IDSIA and professor of Artificial Intelligence at SUPSI in Lugano, Switzerland. His major research interests are in the area of optimisation, simulation, multi-agent learning and adaptation, applied to both academic and real-world problems. He is a co-author of several state-of-the-art algorithms for combinatorial optimisation problems. He leads several research and industrial projects both at Swiss and European level.