Ant Agents for Hybrid Multipath Routing in Mobile Ad Hoc Networks

Frederick Ducatelle*, Gianni Di Caro and Luca Maria Gambardella
Istituto Dalle Molle sull'Intelligenza Artificiale (IDSIA)
Galleria 2, CH-6928 Manno-Lugano, Switzerland
{frederick,gianni,luca}@idsia.ch

Abstract

In this paper we describe AntHocNet, an algorithm for routing in mobile ad hoc networks based on ideas from the Nature-inspired Ant Colony Optimization framework. The algorithm consists of both reactive and proactive components. In a reactive path setup phase, multiple paths are built between the source and destination of a data session. Data are stochastically spread over the different paths, according to their estimated quality. During the course of the session, paths are continuously monitored and improved in a proactive way. Link failures are dealt with locally. The algorithm makes extensive use of ant-like mobile agents which sample full paths between source and destination nodes in a Monte Carlo fashion. We report results of simulation experiments in which we have studied the behavior of AntHocNet and AODV as a function of node mobility, terrain size and number of nodes. According to the observed results, AntHocNet outperforms AODV both in terms of end-to-end delay and delivery ratio.

1. Introduction

In this work we describe *AntHocNet*, a multipath routing algorithm for mobile ad hoc networks (MANETs) [31]. AntHocNet's design, which combines both proactive and reactive components, is based on the shortest path behavior observed in ant colonies and on the related optimization framework of *Ant Colony Optimization (ACO)* [12].

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 [16, 12]. 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 is called *stigmergy* [36].

These basic ingredients have been reverse-engineered in the framework of ACO, which exploits the ant behavior to define a Nature-inspired metaheuristic for combinatorial optimization. ACO has been applied with success to a variety of combinatorial problems (e.g., traveling salesman, vehicle routing, scheduling, etc., see [12, 13] for overviews), as well as to routing (e.g., [10, 32, 34]).

The first ACO routing algorithms were designed for wired networks (e.g., *AntNet* [10] for packet-switched networks and *ABC* [32] for circuit-switched networks). These algorithms exhibit a number of interesting properties which are also desirable for MANET routing: they can 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 [18] 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

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

can work efficiently in MANETs, while still 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 work

In this section we discuss the basic ideas behind ACO routing, and describe some of the ACO routing algorithms which were proposed for MANETs. We also point out some other algorithms which have elements in common with our algorithm. We also point out some other algorithms which have elements in common with our algorithm, such as *multipath routing*, *data load spreading*, and *proactive path maintenance*.

The basic idea behind ACO algorithms for routing [9, 12] is the use of *mobile agents*, called *ants*. These ants are generated by nodes in the network, with the task to sample a path between the node and an assigned destination. An ant going from source node s to destination node d collects information about the quality of the path (e.g. round trip time, 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 [35]. 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 toward 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. The pheromone information is used for the routing of both ants and data packets: all packets are routed *stochastically*, choosing with a higher probability those links associated with higher pheromone values. In this way data for a same destination are spread over *multiple 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.

The above description 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 done 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* [14, 26] 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* [25] 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)* [18] 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 [19, 30, 33] 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.

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 [27] 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 [15], whereas others look for link [24] or node [39] disjoint paths, or even paths which are outside each other's interference range [38]. 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 [15, 37]. *Proactively improving existing paths* is quite rare in MANET routing algorithms, although one possible approach is presented in [17] (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 [21, 28]. Some algorithms spread data over the multiple paths in a simple, even way [22], and in a few cases *adaptive data load spreading* depending on the estimated quality of paths, similar to the ACO ideas, is explored [15, 37]. 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 [7]. *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 [11].

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}:

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

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 1. In order to obtain a mesh of sufficiently disjoint multiple paths as shown in Figure 2, 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 [24].

Each forward ant keeps a list $\mathcal{P}$ of the nodes $[1, \ldots, n]$ 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

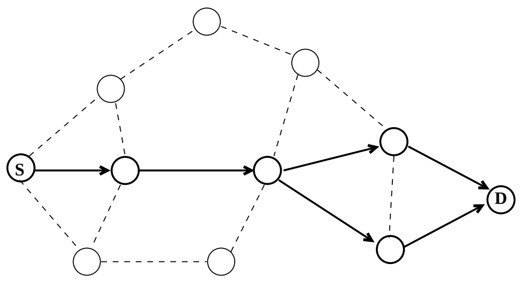

Figure 1. "Kite-shaped" multiple paths

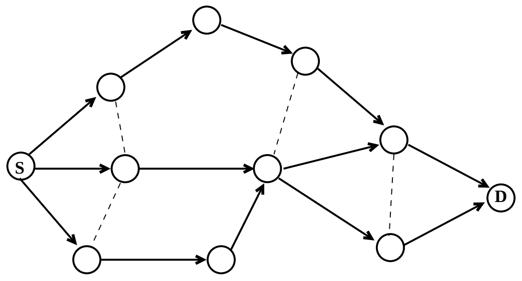

Figure 2. A mesh of multiple paths

travel over $\mathcal{P}$ towards the destination, which is used to update routing tables. $\hat{T}_{\mathcal{P}}$ is the sum of local estimates $\hat{T}^i_{i+1}$ in each node $i \in \mathcal{P}$ of the time to reach the next hop $i+1$:

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

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

$$\hat{T}^i_{i+1} = (Q^i_{mac} + 1)\hat{T}^i_{mac} \ .$$

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

$$\hat{T}^i_{mac} = \alpha\hat{T}^i_{mac} + (1-\alpha)t^i_{mac},$$

with $\alpha \in [0,1]$. Since $\hat{T}^i_{mac}$ is calculated at the MAC layer it includes channel access activities, so it takes into account local congestion of the shared medium. Forward ants calculate a similar time estimate $\hat{T}_{\mathcal{P}}$, 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 $\mathcal{T}^i_{nd}$ in $\mathcal{T}^i$. The pheromone value in $\mathcal{T}^i_{nd}$ 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 $\hat{T}^i_d$ is the travelling time estimated by the ant, and h is the number of hops, the value τ^i_d used to update the running average is defined as:

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

where T_{hop} is a fixed value representing the time to take one hop in unloaded conditions. Defining τ^i_d 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 $\mathcal{T}^i_{nd}$ is updated as follows:

$$\mathcal{T}^i_{nd} = \gamma\mathcal{T}^i_{nd} + (1-\gamma)\tau^i_d, \ \gamma \in [0,1],$$

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:

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

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, including some of the pheromone information of a node in the hello messages it sends to its neighbors. This will allow 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. Further actions depend on the event which was associated with the discovered disappearance. If the event was a failed transmission of a control packet, 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 event was the failed transmission of a data packet, the node sends the link failure notification only about the destinations for which it lost its best next hop if this was not the only next hop. For the destinations for which it lost its only next hop, the node starts a *local route repair*. 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 find an alternative 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 destinations.

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

We evaluated our algorithm in a number of simulation tests. We compare its performance to that of Ad-Hoc On-Demand Distance Vector Routing (AODV) [29] (with route repair), a state-of-the-art MANET routing algorithm and a de facto standard. The 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). In 4.1 we describe the simulation environment and the test scenarios, and in 4.2 we show and discuss the results. The parameters of the AntHocNet algorithm were set to the values mentioned in the previous section for all the different test settings. These values were obtained from empirical experience, without tuning them separately for each considered scenario (the algorithm is rather robust with respect to the setting of most parameters).

4.1. Simulation Environment

As simulation software we use Qualnet. We ran experiments with two different base settings. In the first setting, 100 nodes are randomly placed inside an area of $3000 \times 1000\ m^2$. Each experiment is run for 900 seconds. Data traffic is generated by 20 constant bit rate (CBR) sources using UDP at the transport layer. The sources send 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.[1] 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 popular 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* movement model [20], in which we varied the maximum speed and the pause time, and with the *Gauss-Markov* movement model [6], in which we again varied the maximum speed. The Gauss-Markov movement scenarios were generated with the BonnMotion software [8]. 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 [23]. 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.

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

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

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 [11]. 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.

4.2. Simulation results

In the first set of experiments we study the behavior of the algorithm in increasingly dynamic environments. All

1 We did not do any experiments with TCP at the transport layer, since TCP is known not to function well in MANET environments [1]. Also, TCP is not equipped to deal with multipath routing algorithms like AntHocNet (e.g. see [3] for a proposal of an adaptation of TCP to deal with multipath routing algorithms).

tests are done with the sparse network of 100 nodes in $3000 \times 1000\ m^2$. Figures 3 and 4 show the evolution of the delivery ratio and average delay for increasing node speed (from 10 to 50 m/s) in a random waypoint model. AntHocNet clearly outperforms AODV for all settings, while the performance difference between the algorithms remains more or less constant.

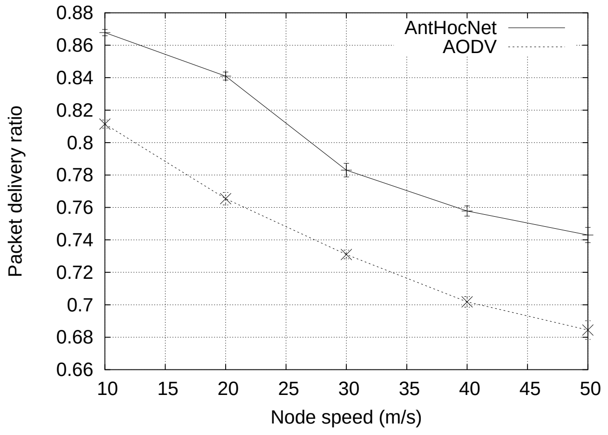

Figure 3. Delivery ratio under various speed values for random waypoint mobility.

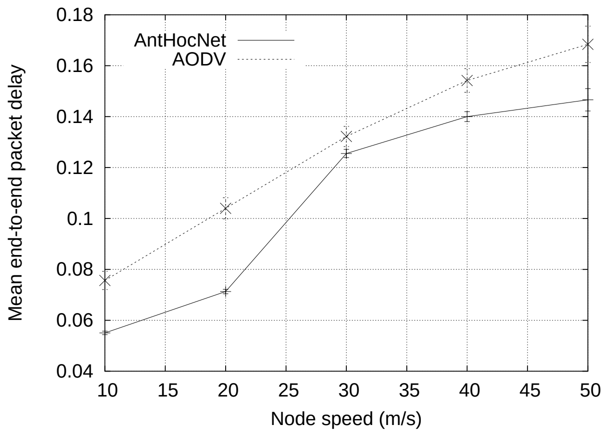

Figure 4. Average packet delay under various speed values for random waypoint mobility.

Figures 5 and 6 report results of an analogous study in function of node speed, but this time in a Gauss-Markov mobility model. Again AntHocNet performs better, and this time there is a significant increase in its advantage as the network becomes more mobile, especially for what concerns delay. This performance difference might be explained by the properties of the movement patterns created by the different mobility models. A node which moves according to the random waypoint model chooses its speed and direction after each turning point completely at random. In the Gauss-Markov model, on the other hand, there is a programmed correlation between subsequent movements. AntHocNet is an adaptive algorithm in which nodes use continuous sampling to *learn statistical estimates* about paths, and in turn use these estimates to guide the search for new and/or better paths. Such an approach can of course work better if there are exploitable regularities and correlations in the environment. AODV on the other hand is a reactive algorithm, which starts more or less from scratch after every route failure, and cannot benefit from the presence of such regularities.

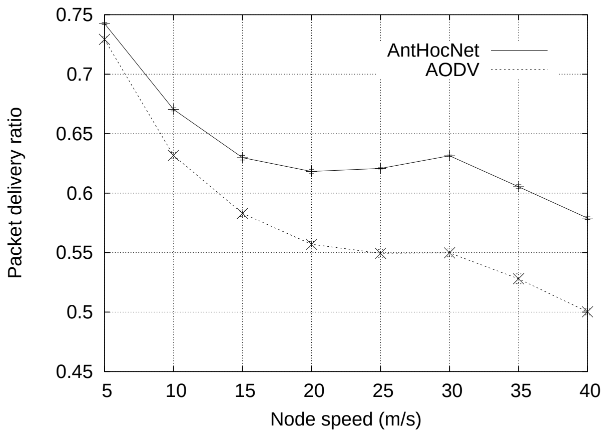

Figure 5. Delivery ratio under various speed values for Gauss-Markov mobility.

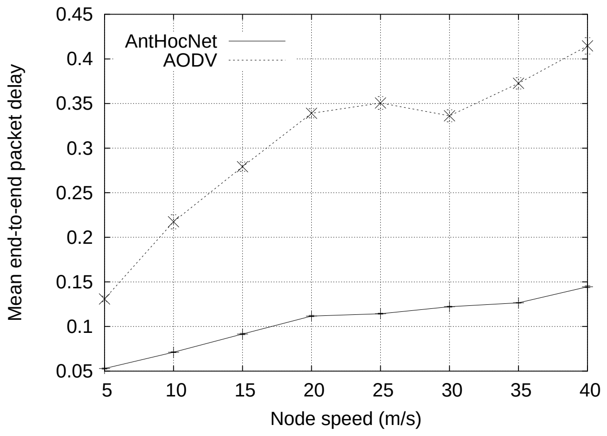

Figure 6. Average packet delay under various speed values for Gauss-Markov mobility.

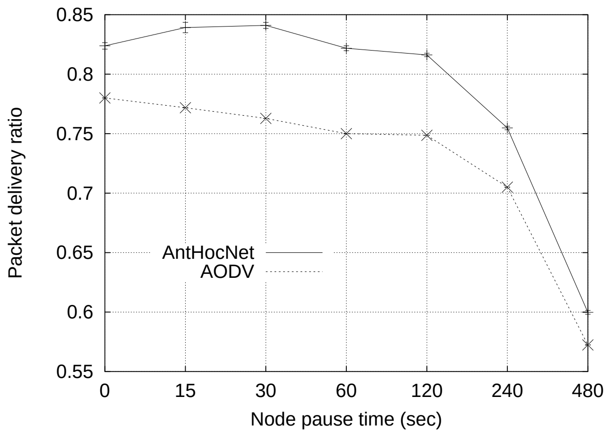

Figure 7. Delivery ratio under various pause times for random waypoint mobility.

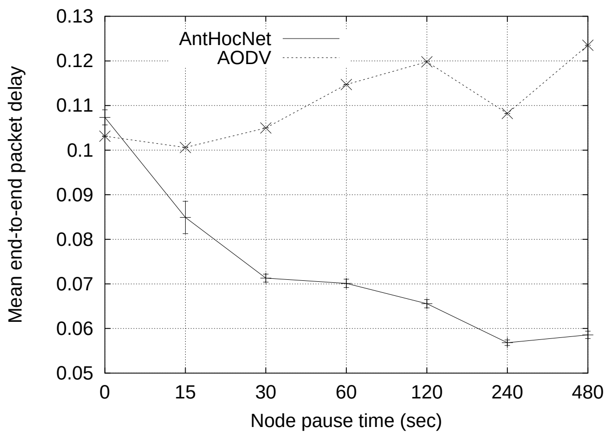

Figure 8. Average packet delay under various pause times for random waypoint mobility.

Figures 7 and 8 present the results for different node pause time values. This is another parameter which influences the node mobility in the random waypoint model: the lower the pause time, the higher the mobility. For the delivery ratio, the results are more or less in line with what we saw before: AntHocNet scores better than AODV. The dip in performance for the highest pause times (so for supposedly easier scenarios) is likely caused by the fact that the network is quite sparse, and in the same time quite static: this means that nodes can become unreachable for long periods. Also in terms of delay AntHocNet outperforms AODV, but not in the scenario with lowest pause time. This is probably again due to the fact that nodes are moving constantly with uncorrelated changes in directions and speeds. The very bad relative performance in delay of AODV in the case of high pause times can probably be attributed to an artefact in the algorithms: while AntHocNet deletes packets which have been buffered for a long time, AODV keeps them, and can still send them much later. This means that packets for nodes which are temporary unreachable can be delivered much after their generation, experiencing very high delays.

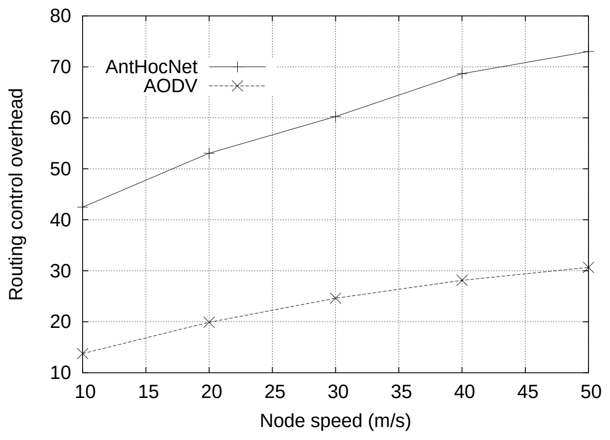

Figure 9. Routing control overhead in number of control packets per successfully delivered data packet under various speed values for random waypoint mobility.

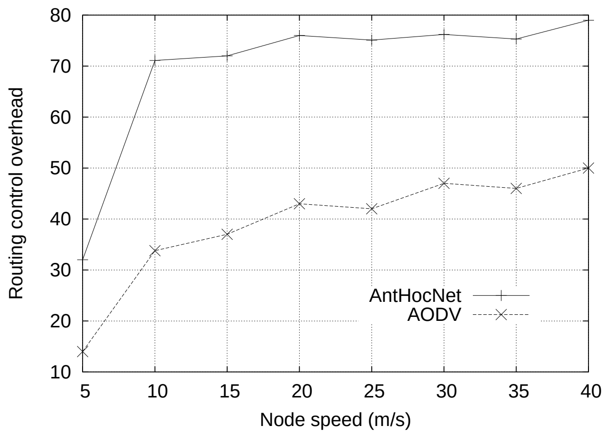

Figure 10. Routing control overhead in number of control packets per successfully delivered data packet under various speed values for Gauss-Markov mobility.

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

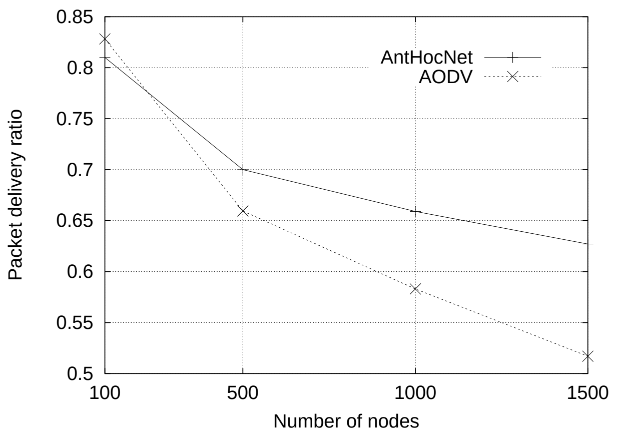

Figure 11. Delivery ratio under increasing network sizes.

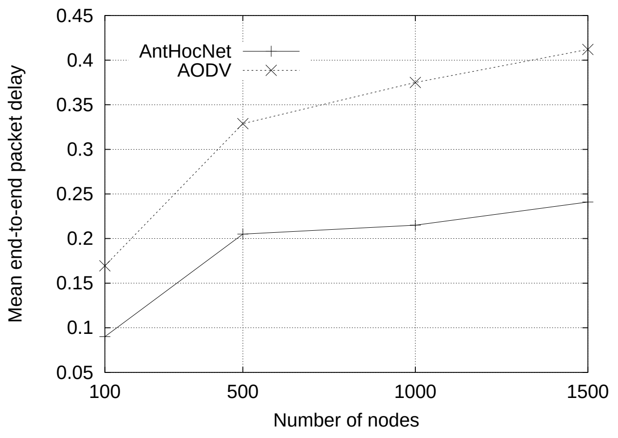

Figure 12. Average packet delay under increasing network sizes.

to adapt to the ever changing MANET environment and be able to provide a high delivery ratio and low delays. Figures 9 and 10 show the overhead in the considered test scenarios with variable speed for the two mobility models. Overhead is expressed as the number of control packets forwarded divided by the number of data packets delivered correctly. AntHocNet generates substantially (1.5 to 3 times) more overhead than AODV, although the differences remain relatively stable for different speed values. We are currently working to improve on the created overhead, in the first place using the technique of pheromone diffusion mentioned in 3.3. This gives proactive ants specific guidelines about the paths to follow, so that we can avoid to let them proliferate blindly. According to preliminary results, this can lead to better results with less ants.

In the last set of experiments, of which the results are visualized in figures 11 and 12, we study the two algorithms in large network simulations. Both in terms of delivery ratio and delay AntHocNet clearly outperforms AODV. Probably the mechanisms of multipath routing and local repair pay off more when paths are longer. The good performances of the algorithm in these studies give an indication of its scalability.

5. Conclusions and future work

We have described AntHocNet, an ACO algorithm for routing in MANETs. It is a hybrid algorithm, combining reactive route setup with proactive route probing and exploration. In simulation experiments we show that AntHocNet can outperform AODV in terms of delivery ratio and average delay, especially in more mobile and larger networks. The algorithm seems to benefit a lot from situations in which there are some regularities and correlations which can be learned and exploited for data transport and path discovery. The algorithm also shows good scalability.

In future work we will improve the exploratory working of proactive ants. By extending the concept of pheromone diffusion, more information about possible path improvements will be available in the nodes, and this information can guide proactive ants. This should lead to better results with less overhead. We also want to make the generation and forwarding of proactive ants adaptive to the network situation.

References

[1] V. Anantharaman, S.-J. Park, K. Sundaresan, and R. Sivakumar. TCP performance over mobile ad hoc networks: a quantitative study. *Wireless Communications and Mobile Computing*, 4:203–222, 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] S. Bohacek, J. Hespanha, J. Lee, C. Lim, and K. Obraczka. A new TCP for persistent packet reordering-TCP-PR. *Accepted for Publication in Transactions on Networking*, 2004.

[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. Loureiro. Gps/ant-like routing in ad hoc networks. *Telecommunication Systems*, 18(1–3):85–100, 2001.

[6] 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.

[7] D. De Couto, D. Aguayo, B. 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.

[8] C. de Waal. Bonnmotion: A mobility scenario generation and analysis tool, 2002. `http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion/`.

[9] 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, Forthcoming, 2004.

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

[11] G. Di Caro, F. Ducatelle, and L. Gambardella. AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks. In *Proceedings of Parallel Problem Solving from Nature (PPSN VIII)*, LNCS. Springer-Verlag, 2004.

[12] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for distributed discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[13] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[14] K. Fujita, A. Saito, T. Matsui, and H. Matsuo. An adaptive ant-based routing algorithm used routing history in dynamic networks. In *4th Asia-Pacific Conf. on Simulated Evolution and Learning*, 2002.

[15] 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.

[16] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the Argentine ant. *Naturwissenschaften*, 76:579–581, 1989.

[17] C. Gui and P. Mohapatra. SHORT: Self-healing and optimizing routing techniques for mobile adhoc networks. In *Proceedings of MobiHoc*, 2003.

[18] 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.

[19] M. Heissenbüttel and T. Braun. Ants-based routing in large scale mobile ad-hoc networks. In *Kommunikation in verteilten Systemen (KiVS03)*, March 2003.

[20] D. Johnson and D. Maltz. *Mobile Computing*, chapter Dynamic Source Routing in Ad Hoc Wireless Networks, pages 153–181. Kluwer, 1996.

[21] 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.

[22] S.-J. Lee and M. Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of IEEE ICC*, 2001.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[30] M. Roth and S. Wicker. Termite: Emergent ad-hoc networking. In *The Second Mediterranean Workshop on Ad-Hoc Networks*, 2003.

[31] E. Royer and C.-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 1999.

[32] R. Schoonderwoerd, O. Holland, J. Bruten, and L. Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207, 1996.

[33] 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. To appear.

[34] K. Sim and W. 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.

[35] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[36] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life,* Special Issue on Stigmergy, 5:97–116, 1999.

[37] L. Wang, Y. Shu, O. Yang, M. Dong, and L. Zhang. Adaptive multipath source routing in wireless ad hoc networks. In *Proc. of the IEEE Int. Conf. on Communications*, 2001.

[38] K. Wu and J. Harms. On-demand multipath routing for mobile ad hoc networks. In *Proceedings of EPMCC*, 2001.

[39] Z. Ye, S. Krishnamurthy, and S. Tripathi. A framework for reliable routing in mobile ad hoc networks. In *Proc. of IEEE INFOCOM*, 2003.