SWARM INTELLIGENCE FOR ROUTING IN MOBILE AD HOC NETWORKS

Gianni Di Caro, Frederick Ducatelle and Luca Maria Gambardella

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

ABSTRACT

Mobile Ad Hoc Networks are communication networks built up of a collection of mobile devices which can communicate through wireless connections. Routing is the task of directing data packets from a source node to a given destination. This task is particulary hard in Mobile Ad Hoc Networks: due to the mobility of the network elements and the lack of central control, routing algorithms should be robust and adaptive and work in a decentralized and self-organizing way. In this paper, we describe an algorithm which draws inspiration from Swarm Intelligence to obtain these characteristics. More specifically, we borrow ideas from ant colonies and from the Ant Colony Optimization framework. In an extensive set of simulation tests, we compare our routing algorithm with a state-of-the-art algorithm, and show that it gets better performance over a wide range of different scenarios and for a number of different evaluation measures. In particular, we show that it scales better with the number of nodes in the network.

1. INTRODUCTION

In communications network research, there is currently an increasing interest for the paradigm of *autonomic computing* [14]. The idea is that networks are becoming more and more complex and that it is desirable that they can self-organize and self-configure, adapting to new situations in terms of traffic, services, network connectivity, etc.. To support this new paradigm, future network algorithms should be robust, work in a distributed way, be able to observe changes in the network, and adapt to them.

Nature's self-organizing systems like *insect societies* show precisely these desirable properties. Making use of a number of relatively simple biological agents (e.g., the ants) a variety of different organized behaviors are generated at the system-level from the local interactions among the agents and with the environment. The robustness and effectiveness of such collective behaviors with respect to variations of environment conditions are key-aspects of their biological success. This kind of systems are often referred to with the term *Swarm Intelligence*. Swarm systems have recently become a source of inspiration for the design of distributed and adaptive algorithms, and in particular of routing algorithms. *Routing* is the task of directing data flows from sources to destinations maximizing network performance. It is at the core of all network activities. Several successful routing algorithms have been proposed taking inspiration from ant colony behavior and the related framework of *Ant Colony Optimization* (ACO) [8]. Examples of *ACO routing algorithms* are AntNet [6] and ABC [19].

One type of networks where the need for autonomic control is intrinsically necessary are *Mobile Ad Hoc Networks* (MANETs) [17]. These 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 centralized control or overview. There are no designated routers: nodes serve as routers for each other, and data packets are forwarded from node to node in a multi-hop fashion.

The ACO routing algorithms mentioned before were developed for wired networks. They work in a distributed and localized way, and are able to observe and adapt to changes in traffic patterns. However, changes in MANETs are much more drastic: in addition to variations in traffic, both topology and number of nodes can change continuously. Further difficulties are posed by the limited practical bandwidth of the shared wireless channel: although the data rate of wireless communication can be quite high, algorithms used for medium access control, such as IEEE 802.11 DCF[12] (the most commonly used in MANETs), create a lot of overhead both in terms of control packets and delay, lowering the effectively available bandwidth. The challenges of autonomic control are therefore much bigger, and new designs are necessary to guarantee even the basic network functions.

In the following, we describe *AntHocNet*, an ant inspired algorithm for routing in MANETs. Building on ideas from previous work on ACO routing, in combination with tech-

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

niques from dynamic programming, it is tailored to deal with the challenges posed by the extreme dynamics of MANET environments. We compare the algorithm with traditional approaches and show its superiority especially under those conditions where the difference with wired networks is more evident. We also present results indicating that the algorithm is remarkably scalable. The rest of this article is organized as follows. First we introduce some necessary background, next we describe the algorithm, and finally we present the results of an extensive simulation study.

2. SCIENTIFIC BACKGROUND

In this section we discuss some issues related to the work presented in this paper, and introduce some notions which will be used in the rest of this paper.

2.1. MANET routing algorithms

Many MANET routing algorithms have been proposed. In the literature, the classical distinction is between table-driven and demand-driven algorithms [17]. *Table-driven* algorithms, such as DSDV [15], are purely *proactive*: all nodes try to maintain routes to all other nodes at all times. This means that they need to keep track of all topology changes, which can be difficult if there are a lot of nodes or if they are very mobile. *Demand-driven* algorithms, such as AODV [16], are purely *reactive*: nodes only gather routing information when a data session to a new destination starts, or when a route which is in use fails. Reactive algorithms are in general more scalable [4] since they reduce routing overhead, but they can suffer from oscillations in performance because they are never prepared for disruptive events. In practice, many algorithms are *hybrid* (e.g. ZRP [11]), using both proactive and reactive components. Also AntHocNet can be described as a hybrid algorithm.

2.2. Stigmergic learning and ACO routing

Stigmergy is a 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 [21]. The local environment's variables whose values determine in turn the characteristics of the agents' response, are called *stigmergic variables*. An example of a stigmergic process is the mechanism used by ant colonies to find the shortest path between their nest and a food source [10, 8]. The main catalyst of this colony-level shortest path behavior is the use of a volatile chemical substance called *pheromone*, which acts as a stigmergic variable: ants moving back and forth between the nest and a food source lay pheromone, and preferentially move towards areas of higher pheromone intensity. Shorter paths can be completed quicker and more frequently by the ants, and are therefore marked with higher pheromone intensity. These paths can then attract more ants, which in turn increases the pheromone level. The overall effect is a *distributed reinforcement learning* [20] process which eventually allows the majority of the ants to converge onto the shortest path.

The ant colony shortest path behavior has attracted a lot of attention, and has been reverse-engineered in the context of ACO [8]. For this paper, we are interested in the application of *ACO for routing* [5]. In ACO routing algorithms, routing information is gathered through a stigmergic learning process using ant agents. These are lightweight agents which are generated concurrently and independently by the nodes, with the task to sample path to an assigned destination. An ant going from its source s to a destination d collects information about the quality of the path it follows (e.g. end-to-end delay), and, retracing its way back from d to s, uses this to update the routing information at intermediate nodes. Following the datagram model of IP networks, routing information is expressed in the form of tables kept locally at each node. The routing table $\mathcal{T}^i$ at node i is a matrix, where each entry $\mathcal{T}^i_{nd} \in \mathbb{R}$ of the table is a value indicating the estimated goodness of going from i over neighbor n to reach destination d. These table entries play the role of stigmergic variables in the learning process: an ant agent sampling a path to its destination d makes a *stochastic routing decision* at each node, giving higher probability to decisions with high goodness, while an ant retracing its way back from d to s updates the table entries influencing the routing of other ants. The routing tables are therefore also called *pheromone tables* and the goodness values *pheromone values*. Data packets are routed more or less in the same way as ants: packets are routed *stochastically*, choosing with a higher probability those links associated with higher pheromone values. The result is automatic balancing of the data load over the network.

2.3. Information bootstrapping

The way ACO routing algorithms gather information could be described as a *Monte Carlo learning* process: ants try out complete paths from source to destination, and gather information about it. Hence, the estimated goodness values for routing decisions recorded in the pheromone tables are the result of the direct experiences of the ants. In the *reinforcement learning* [20] literature, Monte Carlo learning is often contrasted with a different learning paradigm, namely *information bootstrapping*. Bootstrapping is a characteristic of dynamic programming. Nodes estimate the cost-to-go of a path by combining the cost estimates made by neighboring nodes and the cost to go to those neighboring nodes, rather than by the direct sampling of a full path. Information bootstrapping is the typical mode of operation in classical Bellman-Ford routing algorithms [1], as well as in derived

table-driven algorithms like DSDV, and some reinforcement learning inspired approaches to routing, like Q-routing [3].

Compared to Monte Carlo sampling, bootstrapping is more efficient in a stationary environment. However, in a dynamic environment it can be unreliable, because wrong information can arise from combining new and old estimates, and because errors can easily propagate since estimates are built up using other estimates. In AntHocNet, we attempt to combine the best of both learning methods: the Monte Carlo sampling of paths by ants is supported by a lightweight information bootstrapping process which provides a second way of building up goodness estimates. This bootstrapping process, which we call *pheromone diffusion*, and its interaction with the Monte Carlo sampling process, is described in detail in subsection 3.3.

3. DESCRIPTION OF THE ALGORITHM

AntHocNet's design is inspired by ACO routing algorithms for wired networks. It uses ant agents which follow and update pheromone tables in a *stigmergic learning* process. Data packets are *routed stochastically* according to the learned tables. An important difference with other ACO routing algorithms is that AntHocNet is a *hybrid algorithm*, in order to deal better with the specific challenges of MANET environments. It is *reactive* in the sense that nodes only gather routing information for destinations which they are currently communicating with, while it is *proactive* because nodes try to maintain and improve routing information as long as communication is going on. We make a distinction between the *path setup*, which is the reactive mechanism to obtain initial routing information about a destination at the start of a session, and *path maintenance and improvement*, which is the normal mode of operation during the course of a session to proactively adapt to network changes. Path maintenance and improvement is supported by the *pheromone diffusion* process mentioned in 2.3: the routing information obtained via stigmergic learning is spread between the nodes of the MANET in an information bootstrapping process to provide secondary guidance for the learning agents. *Link failures* are dealt with using a local path repair process or via notification messages. In the following we provide a concise description of each of these components. A more detailed description of the AntHocNet algorithm can be found in [7, 9].

3.1. Pheromone tables

Paths are implicitly defined by the pheromone tables which are kept locally at each node. An entry $\mathcal{T}^i_{nd} \in \mathbb{R}$ of the pheromone table $\mathcal{T}^i$ at node i contains a value indicating the estimated goodness of going from i over neighbor n to reach destination d. This goodness is a combined measure of path end-to-end delay and number of hops. These are two commonly used quality measures in MANETs. Combining them is also a way to smooth out possibly large oscillations in the time estimates gathered by the ants (e.g., due to local bursts of traffic). Since AntHocNet only maintains information about destinations which are active in a communication session, and the neighbors of a node change continually, the filling of the pheromone tables is sparse and dynamic.

3.2. 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*. At each node, the ant is either unicast or broadcast, according to whether or not the current node has pheromone information for d. If information is available, the ant chooses its next hop n with the probability P_{nd} which depends on the relative goodness of n as a next hop, expressed in the pheromone variable $\mathcal{T}^i_{nd}$:

$$P_{nd} = \frac{(\mathcal{T}^i_{nd})^\beta}{\sum_{j \in \mathcal{N}^i_d} (\mathcal{T}^i_{jd})^\beta}, \ \beta \geq 1, \tag{1}$$

where $\mathcal{N}^i_d$ is the set of neighbors of i over which a path to d is known, and β is a parameter which controls the exploratory behavior of the ants. If no pheromone is available, the ant is broadcast. Due to subsequent broadcasts, many duplicate copies of the same ant travel to the destination. A node which receives multiple copies of the same ant only accepts the first and discards the other. This way, only one path is set up initially. During the course of the communication session, more paths are added via the proactive path exploration and maintenance mechanism (see subsection 3.3) to provide a *mesh of multiple paths* for data forwarding.

Each forward ant keeps a list $\mathcal{P} = [1, 2, \ldots, d]$ of the nodes it has visited. Upon arrival at the destination d, it is converted into a *backward ant*, which travels back to the source retracing $\mathcal{P}$. At each intermediate node $i \in \mathcal{P}$ ($i < d$), the backward ant reads a locally maintained estimate $\hat{T}^i_{i+1}$ of the time it takes to reach the neighbor $i+1$ the ant is coming from. The time $\hat{T}^i_d$ it would take a data packet to reach d from i over $\mathcal{P}$ is calculated incrementally as the sum of the local estimates $\hat{T}^j_{j+1}$ gathered by the ant between i and d. This time estimate $\hat{T}^i_d$ is combined with the number of hops h between i and d over $\mathcal{P}$ to calculate the pheromone update value τ^i_d. A pheromone value is a goodness measure expressed as an inverted cost. It has the dimension of an inverted time. Therefore, we use the following formula to calculate τ^i_d:

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

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

$$\mathcal{T}_{nd}^i = \gamma \mathcal{T}_{nd}^i + (1-\gamma)\tau_d^i, \quad \gamma \in [0,1]. \tag{3}$$

Once the backward ant makes it back to the source, a full path is set up and the source can start sending data. If the backward ant does not arrive for some reason, a timer runs out at the source, and the whole process is started again.

3.3. Proactive path maintenance and exploration

During the course of a communication session, source nodes send out *proactive forward ants* to update the information about currently used paths and to try to find new and better paths. They follow pheromone and update routing tables in the same way as reactive forward ants. Such continuous sampling of paths and pheromone updating by ants is the typical mode of operation in ant inspired routing algorithms. However, in MANET environments, characterized by constant changes, the needed ant sending frequency would be quite high, so that the process would get in conflict with the typically limited bandwidth in such networks. Moreover, to find entirely new paths, too much blind exploration through random walks or broadcasts would be needed, again leading to excessive bandwidth consumption. Therefore, we introduce at this point the supporting *pheromone diffusion* function which allows to spread pheromone information over the network. This process provides a second way of updating pheromone information about existing paths, and can give information to guide exploratory behavior.

The pheromone diffusion is implemented using short *messages*, broadcast periodically and asynchronously by the nodes to all their neighbors. In these messages, the sending node n places a list of destinations it has information about, including for each of these destinations d the best pheromone value $\mathcal{T}_{m^*d}^n, m^* \in \mathcal{N}_d^n$, which n has available for d. A node i receiving the message from n first of all registers that n is its neighbor. Then, for each destination d listed in the message, it derives an estimate of the goodness of going from i to d over n, combining the cost of hopping from i to n with the reported pheromone value $\mathcal{T}_{m^*d}^n$. We call the obtained estimate the *bootstrapped pheromone* variable $\mathcal{B}_{nd}^i$, since it is built up using an estimate which is non-local to i (see subsection 2.3). This bootstrapped pheromone variable can in turn be forwarded in the next message sent out by n, giving rise to a bootstrapped pheromone field over the MANET. This field is complementary to the field of regular pheromone, learned via ant-based Monte Carlo sampling.

Bootstrapped pheromone is used directly for the *maintenance* of existing paths. If i already has a regular pheromone entry $\mathcal{T}_{nd}^i$ in its routing table for destination d going over neighbor n, $\mathcal{B}_{nd}^i$ is treated as an update of the goodness estimate of this path, and is used directly to replace $\mathcal{T}_{nd}^i$. Due to the slow multi-step forwarding of bootstrapped pheromone, this information does not provide the most accurate view of the current situation. However, it is obtained via a lightweight, efficient process, and is complemented by the explicit path updating done by the ants. In this way we have two updating frequencies in the path maintenance process.

For path *exploration*, bootstrapped pheromone is used indirectly. If i does not yet have a value for $\mathcal{T}_{nd}^i$ in its routing table, $\mathcal{B}_{nd}^i$ could indicate a possible new path from i to d over n. However, this path has never been sampled explicitly by an ant, and due to the slow multi-step pheromone bootstrapping process it could contain undetected loops or dangling links. It is therefore not used directly for data forwarding. It is seen as a sort of *virtual pheromone*, which needs to be tested. Proactive forward ants will use both the regular and the virtual pheromone on their way to the destination, so that they can test the proposed new paths. This way, promising virtual pheromone is investigated, and can be turned into a regular path which can be used for data. This increases the number of paths available for data routing, which grows to a full mesh, and allows the algorithm to exploit new opportunities in the ever changing topology.

3.4. Stochastic data routing

Nodes in AntHocNet *forward data stochastically* according to the pheromone values. 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 like for reactive forward ants, using equation 1. However, a higher value for the exponent β is used in order to be greedy with respect to better paths. According to this strategy, we do not choose a priori how many paths to use: their number is automatically selected in function of their quality. This probabilistic routing strategy leads to data load spreading according to the estimated quality of the paths. When a path is worse than others, it is avoided, and its congestion is relieved. Other paths get more traffic, leading to higher congestion, which makes their delay increase. If estimates are kept up-to-date, this leads to *automatic load balancing*.

3.5. Link failures

Nodes can detect link failures (e.g., a neighbor has moved away) when unicast transmissions fail, or when expected periodic pheromone diffusion messages were not received. When a node i discovers the disappearance of a neighbor n, it takes a number of actions. In the first place, i registers that n is no longer a neighbor, and removes all associated entries from its pheromone table. Next, i broadcasts a *link failure notification* message. This message contains a list of the destinations to which i lost its best path, and the new best pheromone to this destination (if it still has entries for the destination). All its neighbors receive the message and

update their pheromone table using the new estimates. If they in turn lost their best or their only path to a destination, they will pass the updated message on to their neighbors, until all concerned nodes are notified of the new situation.

If i detected the link failure via the failed transmission of a data packet, and it has no further paths available for the destination of this packet, it starts a *local route repair*. i 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 restricted number of broadcasts so that its proliferation is limited. If the local repair fails, i broadcasts a new link failure notification message to warn its neighbors.

4. EXPERIMENTAL RESULTS

We evaluate our algorithm in a number of simulation tests. We compare its performance with AODV [16] (with local route repair), a state-of-the-art MANET routing algorithm and a de facto standard. As simulation software, we use QualNet, a commercial simulation package [18].

We study the behavior of the algorithm in function of a number of different properties of the network scenario. All of the test scenarios are obtained by varying parameters in a specific base scenario. In this base scenario, 100 nodes move in a flat, rectangular area of $3000 \times 1000\ m^2$. Movement patterns are generated according to the random waypoint mobility model (RWP) [13]: they choose a random destination point and a random speed, move to the chosen point with the chosen speed, and rest there for a fixed amount of pause time before they choose a new destination and speed. The speed is chosen between 0 and 20 m/s, and the pause time is 30 seconds. Each simulation runs for 900 seconds. 20 different Constant Bit Rate sources start sending at a random time between 0 and 180 seconds and keep sending till the end. At the Medium Access Control layer, the IEEE 802.11b DCF protocol is used. As measures of performance, we use the *average end-to-end delay* for data packets and the *ratio of correctly delivered versus sent packets*. These are standard measures of *effectiveness* in MANETs. We also report *delay jitter*, the average difference in inter-arrival time between packets. As measure of *efficiency*, we consider *routing overhead*, in terms of number of control packets forwarded per successfully delivered data packet.

We investigate AntHocNet's performance for varying levels of mobility and node density, for increasing network sizes, and for different data traffic patterns. At the end of the section, we also show results reporting the performance of the algorithm at a smaller time scale: following the evolution of the end-to-end delay over the course of a simulation session while some disruptive events take place, we attempt to give an idea of the adaptivity of the algorithm.

To obtain scenarios with *different levels of mobility*, we vary the pause time. Higher pause time means lower mobility, but also lower connectivity (due to specific properties of RWP mobility, see [2]). The results are reported in figures 1 and 2. AntHocNet shows much better effectiveness than AODV, in terms of average delay, delivery ratio, and jitter. AODV has better efficiency, measured as routing overhead, but the difference is rather small. The bad performance for high pause times are due to the reduced connectivity.

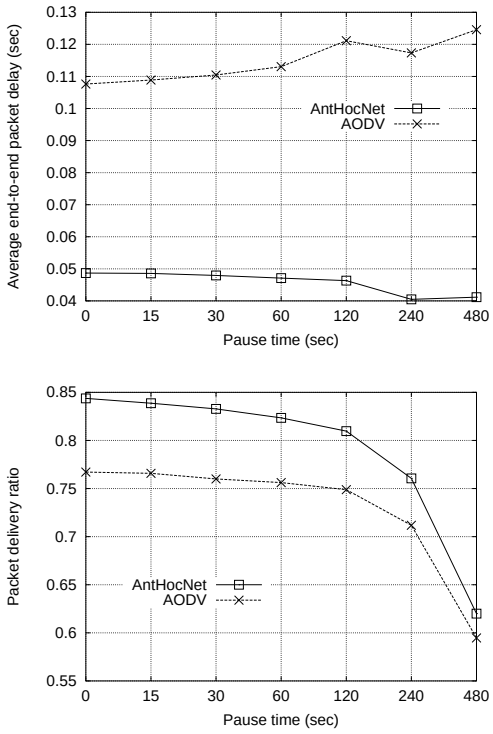

Figure 1. Average delay and delivery ratio for different levels of mobility

Different node density levels are obtained by keeping the area size constant and increasing the number of nodes. The results of these tests are reported in figure 3. Again, AntHocNet performs better than AODV in terms of average end-to-end delay and delivery ratio, and the difference increases with the density. In terms of overhead, AntHocNet is worse than AODV for low densities, but better for high densities. Jitter was not reported here, nor for the remaining tests, due to space limitations. It always follows more or less the trend visible for delay and delivery ratio.

For *different network sizes*, we increase the number of nodes (up to 800) and the area size together, keeping the node density constant. The results are presented in figure 4. AntHocNet's advantage over AODV in terms of average delay and delivery ratio grows with the size of the network.

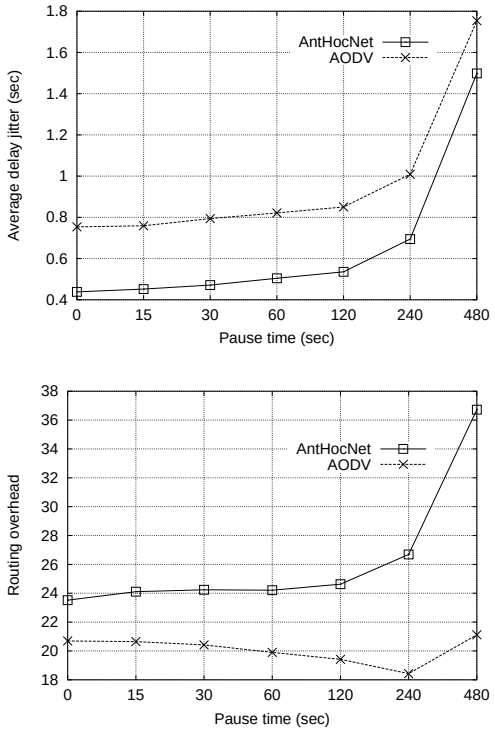

Figure 2. Average jitter and overhead for different levels of mobility

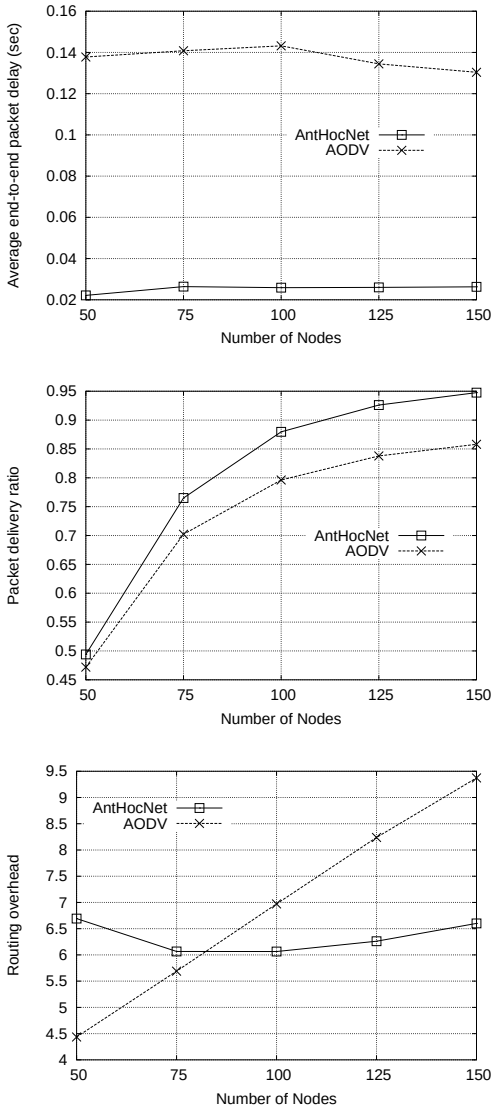

Figure 3. Average delay, delivery ratio and overhead for increasing node density

AntHocNet's overhead grows less fast than that of AODV. This is an important result which indicates that AntHocNet is more scalable with respect to the number of nodes.

For all the previous tests the data traffic consisted of 20 randomly placed CBR sessions. In figure 5 we show results of tests which use *different traffic loads and patterns*. We did tests with 20 and 50 sessions. The sessions are organized around a number of hot spots: 20 (or 50) randomly chosen CBR sources send to a limited number of different destinations. This number of destinations was increased from 1 up to the total number of sessions (corresponding to the randomly placed traffic we used before). Organizing traffic sessions around hot spots reflects the typical situation where traffic is concentrated around a number of important nodes. Again we observe an advantage for AntHocNet in terms of average delay and delivery ratio. This advantage is smaller for the easier scenarios where traffic is concentrated on a low number of hot spots. For the tests with 20 sessions, AntHocNet has higher overhead than AODV, while for those with 50 sessions the picture is more balanced.

For the last test, we report the evolution of the end-to-end delay over the course of a test run in which some important events take place. In this test, 10 randomly chosen sources start to send to the same hot spot between 100 and 110 seconds after the start of the simulation, and keep on sending till the end. After 300 seconds, 20 new sources start to send to a different hot spot. 200 seconds later they stop sending again. All sources send four 64 byte packets per second. Figure 6 shows for one communication session how the end-to-end delay, averaged per 10 seconds, evolves throughout the simulation. The arrival of 20 new sessions after 300 seconds is clearly visible and leads to a long period of unstable behavior: the congestion caused by the high data load can cause strong fluctuations for the delay. AntHocNet's behavior is much smoother than that of AODV however. After the end of the 20 sessions, at second 500, the situation stabilizes again, but faster for AntHocNet than for AODV. The slow decrease of the delay under stable con-

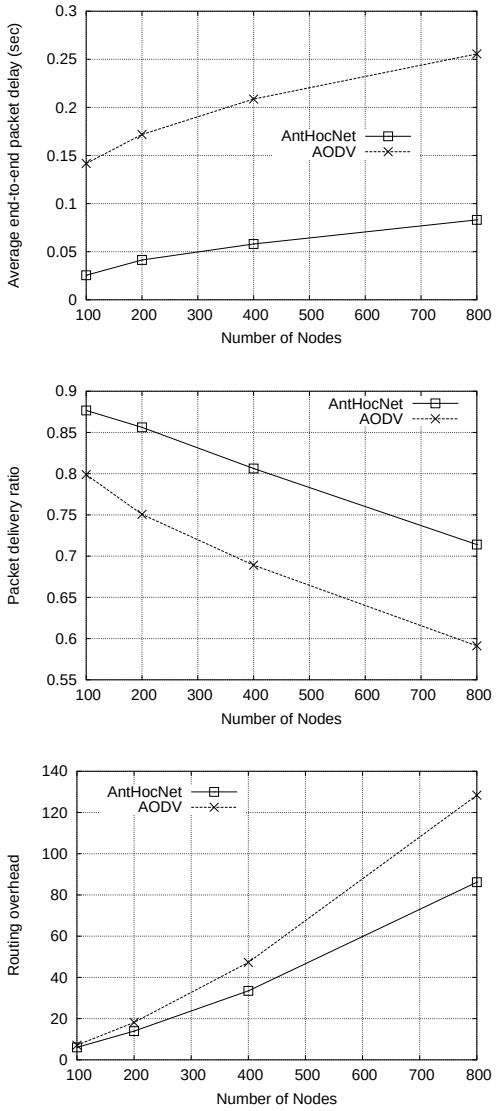

Figure 4. Average delay, delivery ratio and overhead for increasing network sizes

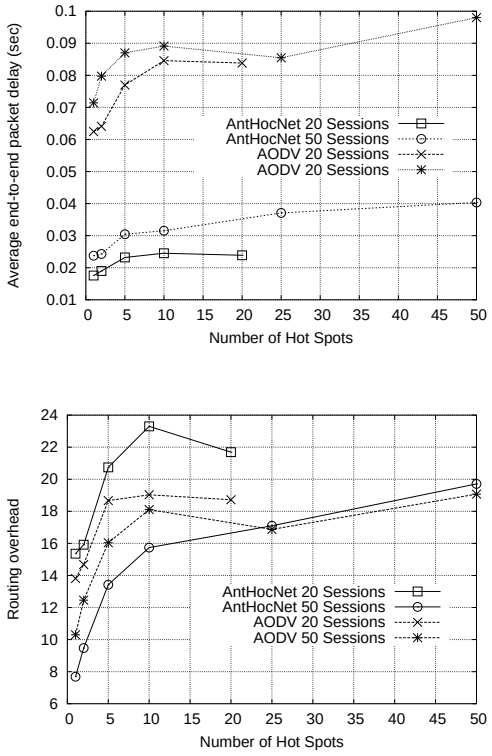

Figure 5. Average delay, delivery ratio and overhead for an increasing number of hot spots

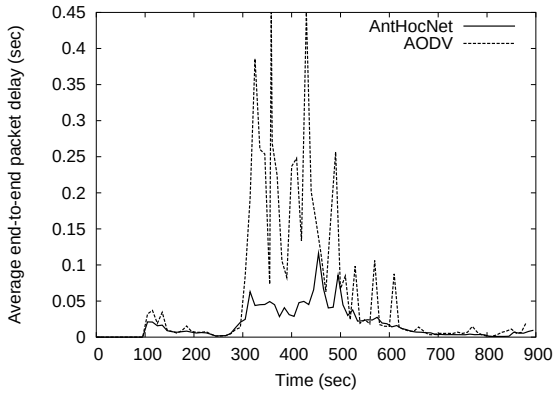

Figure 6. Evolution of the end-to-end delay over the course of a test run

ditions (from 100 till 300 seconds, and from 500 till 850) is due to normal topology changes inside the MANET.

5. CONCLUSIONS

In this paper we have described AntHocNet, a routing algorithm for MANETs which was inspired by ideas from Swarm Intelligence, and more specifically by the framework of ACO. The algorithm combines reactive and proactive behavior to deal with the specific challenges of MANETs in an efficient way. Routing information is learned through Monte Carlo sampling of paths using repeatedly and concurrently generated ant agents, as is common in ACO routing algorithms. This learning process is supported by a secondary process, called pheromone diffusion. Pheromone diffusion provides an alternative way to learn pheromone information, using an information bootstrapping mechanism rather than Monte Carlo sampling. It is used to help update pheromone on existing paths and to provide guidance to ants in search of new paths.

We have evaluated the algorithm in an extensive set of simulation tests. The tests were carried out in a commercial simulator and comparisons were each time made with AODV, a de facto standard in the community. AntHocNet was shown to outperform AODV over the wide range of tested scenarios in terms of delivery ratio, average end-to-end delay and average jitter, while generating a comparable amount of control overhead. An important observation was that the advantage of AntHocNet over AODV grew for larger networks, especially in terms of overhead, suggesting that AntHocNet is more scalable than AODV.

6. REFERENCES

[1] D. Bertsekas and R. Gallager. *Data Networks*. Prentice–Hall, Englewood Cliffs, NJ, USA, 1992.

[2] C. Bettstetter, G. Resta, and P. Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(3):257–269, 2003.

[3] J.A. Boyan and M.L. Littman. Packet routing in dinamically changing networks: A reinforcement learning approach. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6 (NIPS6)*, pages 671–678. Morgan Kaufmann, San Francisco, CA, USA, 1994.

[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 *Proc. of the* 4^{th} *Annual ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom98)*, 1998.

[5] G. Di Caro. *Ant Colony Optimization and its application to adaptive routing in telecommunication networks*. PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, 2004.

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

[7] G. Di Caro, F. Ducatelle, and L.M. Gambardella. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 2005. To appear.

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

[9] F. Ducatelle, G. Di Caro, and L.M. Gambardella. Using ant agents to combine reactive and proactive strategies for routing in mobile ad hoc networks. *International Journal of Computational Intelligence and Applications (IJCIA)*, 2005. To appear.

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

[11] Z.J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proc. of the IEEE Int. Conf. on Universal Personal Communications*, 1997.

[12] 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.

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

[14] J. Kephart and D. Chess. The vision of autonomic computing. *IEEE Computer magazine*, 36(1), 2003.

[15] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, 1994.

[16] C.E. Perkins and E.M. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of the* 2^{nd} *IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

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

[18] Scalable Network Technologies, Inc., Culver City, CA, USA. *Qualnet Simulator, Version 3.6*, 2003. `http://stargate.ornl.gov/trb/tft.html`.

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

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

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