An adaptive multi-agent routing algorithm inspired by ants behavior

Gianni Di Caro and Marco Dorigo

IRIDIA – Université Libre de Bruxelles – Belgium
{gdicaro, mdorigo}@ulb.ac.be

Abstract. This paper introduces AntNet, a novel adaptive approach to routing tables learning in connectionless communications networks. AntNet is inspired by the *stigmergy* communication model observed in ant colonies. We compare AntNet with the current Internet routing algorithm (OSPF), some old Internet routing algorithms (SPF and distributed adaptive Bellman-Ford), and recently proposed forms of asynchronous online Bellman-Ford (Q-routing and Predictive Q-routing). In all the experimental conditions considered AntNet outperforms the competing algorithms, where performance is measured by standard measures such as network throughput and average packet delay,

1. Introduction

Real ants are able to find shortest paths using as only information the pheromone trail deposited by other ants [1]. Ant colony optimization (ACO) algorithms which take inspiration from ants' behavior in finding shortest paths have recently been successfully applied to combinatorial optimization [3,6,11,12,13]. In ant colony optimization a set of artificial ants collectively solve a combinatorial problem by a cooperative effort. This effort is mediated by *stigmergetic communication* [3, 14], that is, a form of indirect communication of information on the problem structure ants collect while building solutions.

In this paper we present AntNet, a novel ACO algorithm applied to the routing problem in connectionless communications networks. In AntNet artificial ants collectively solve the routing problem by a cooperative effort in which stigmergy plays a prominent role. Ants build local models of the network status and adaptive routing tables using indirect and noncoordinated communication of information they collect while exploring the network.

We compare AntNet on a variety of realistic experimental conditions with the following state-of-the-art routing algorithms: Open Shortest Path First (OSPF) [16], Shortest Path First (SPF) [15], distributed adaptive Bellman-Ford [18], and to some recently proposed versions of asynchronous online Bellman-Ford [4, 5]. AntNet was the best performing algorithm in all considered cases.

2. Problem Characteristics and Communication Network Model

Routing algorithms have the goal of directing traffic from sources to destinations maximizing some measure of network performance. Often *throughput* (correctly delivered bits per time unit) and *packet delay* (sec) are the performance measures taken into account. Throughput measures the quantity of service that the network has been

able to offer in a certain amount of time, while packet delay defines the quality of service produced.

The optimal routing problem can be stated as a multi-objective optimization problem in a non-stationary stochastic environment. Delays in information propagation, as well as the difficulty to completely characterize the network dynamics under arbitrary traffic patterns, make the routing problem intrinsically distributed. Routing decisions can only be made on the basis of local and approximate information about the current and the future network states, with additional constraints posed by the network switching and transmission technology.

In this article we focus on wide area networks, that is, irregular topology datagram networks with an IP-like (Internet Protocol) network layer and a very simple transport layer. The instance of the communication network is mapped on a directed weighted graph with N nodes. All the links are viewed as bit pipes characterized by a bandwidth (bits/sec) and a transmission delay (sec), and are accessed following a statistical multiplexing scheme. For this purpose, every routing node holds a buffer space where the incoming and the outgoing packets are stored. This buffer is a shared resource among all the queues attached to every incoming and outgoing link of the node. All the traveling packets are subdivided in two classes: data and routing packets. All the packets in the same class have the same priority, so they are queued and served only on the basis of a first-in-first-out policy, but routing packets have a higher priority than data packets.

Data packets are fed into the network by applications (i.e., processes sending data packets from origin nodes to destination nodes) whose arrival rate is dictated by a selected probabilistic model. The number of packets to send, their sizes and the intervals between them are assigned according to some defined stochastic process. A simple flow control mechanism is implemented which uses a fixed production window for the session's packets generation. The window determines the maximum number of data packets waiting to be sent. Once sent a packet is considered to be acknowledged. In fact, the network simulator we developed (in C++) has not a “real” transport layer. That is, we haven’t implemented mechanisms for a proper management of error, flow, and congestion control. The reason is that we want to check the behavior of our algorithm and of its competitors in conditions which minimize the number of interacting components.

At each node, packets are forwarded towards their destination nodes by the local routing component. Decisions about which outgoing link has to be used are made by using the information stored in the node routing table. When link resources are available, they are reserved and the transfer is set up. The time it takes to a packet to move from one node to a neighboring one depends on its size and on the link transmission characteristics. If on packet's arrival there is not enough buffer space to hold it, the packet is discarded. Packets are also discarded because of expired time to live. After transmission, a stochastic process generates service times for the newly arrived data packet, that is, the delay between its arrival time and the time when it will be ready to be put in the buffer queue of the selected outgoing link.

3. AntNet: Adaptive Agent-based Routing

As we said, the routing problem is a stochastic distributed multi-objective problem. The problem is therefore well suited for a multi-agent approach like our AntNet system, composed of two sets of *homogeneous mobile agents* [19], called *forward* and

backward ants. Agents in each set possess the same structure, but they are differently situated in the environment; that is, they can sense different inputs and they can produce different, independent outputs. In AntNet we retain the core ideas of the ant colony optimization paradigm [10, 12, 13], but we have translated them to match a distributed, dynamic context, different from combinatorial optimization. Ants communicate in an undirect way according to the stigmergy paradigm, through the information they concurrently read and write in two data structures stored in each network node k:

(a) an array $M_k(\mu_d, \sigma_d^2)$ of data structures defining a simple parametric statistical model for the traffic distribution for destinations d, as seen by the local node k,

(b) a routing table, organized as in distance-vector algorithms; in the table, a probability value P_{dn} which expresses the goodness of choosing n as next node when the destination node is d, is stored for each pair (d,n) with the constraint

$$\sum_{n \in N_k} P_{dn} = 1,\ d \in [1, N],\ N_k = \{neighbors(k)\}.$$

The AntNet algorithm can be informally described as follows.

- At regular intervals, every network node s launches a forward ant $F_{s \to d}$ with a randomly selected destination node d. Destinations are chosen to match the current traffic patterns.
- Each forward ant selects the next hop node using the information stored in the routing table. The route is selected, following a random scheme, proportionally to the goodness (probability) of each neighbor node and to the local queues status, and trying to avoid previously visited nodes.
- The identifier of every visited node k and the time elapsed since its launching time to arrive at this k-th node are pushed onto a memory stack $S_{s \to d}(k)$ carried by the forward ant.
- If a cycle is detected, that is, if an ant is forced to return to an already visited node, the cycle's nodes are popped from the ant's stack and all the memory about them is destroyed.
- When the ant $F_{s \to d}$ reaches the destination node d, it generates a backward ant $B_{d \to s}$, transfers to it all of its memory, and then dies.
- The backward ant makes the same path as that of its corresponding forward ant, but in the opposite direction. At each node k along the path it pops its stack $S_{s \to d}(k)$ to know the next hop node.
- Arriving in a node k coming from a neighbor node f, the backward ant updates M_k and the routing table for all the entries corresponding to every node i on the path $k \to d$, that is the path followed by ant $F_{k \to d}$ starting from the current node k.
  - the sample means and variances of the model $M_k(\mu_i, \sigma_i^2)$ are updated with the trip times $T_{k \to i}$ stored in the stack memory $S_{s \to d}(k)$
  - the routing table is changed by incrementing the probabilities P_{if} associated with node f and the nodes i, and decreasing (by normalization) the probabilities P_{in} associated with the other neighbor nodes n. Trip times $T_{k \to i}$ experienced by the forward ant $F_{s \to d}$ are used to assign the probability increments.

$T_{k \to d}$ is the only explicit feedback signal we have: it gives an indication about the goodness r of the followed route because it is proportional to its length from a physical point of view (number of hops, transmission capacity of the used links, processing speed of the crossed nodes) and from a traffic congestion point of view. This is an extremely important aspect of AntNet: forward ants share the same queues as data

packets[1], so if they cross a congested area, they will be delayed. This has a double effect: the trip time will grow and then back-propagated probability increments will be small, and at the same time these increments will be assigned with a bigger delay. The problem is that $T_{k\to d}$ can only be used as a reinforcement signal. In fact, it cannot be associated with an exact error measure, given that we don't know the optimal trip times, which depend on the net load status. The values stored in the model M_k are used to score the trip times by assigning a goodness measure $r \equiv r\,(T_{k\to d}, M_k)$, $r \in]0,1]$ (the smaller $T_{k\to d}$, the higher r). This dimensionless value takes into account an average of the observed values and of their dispersion: $r \propto (1 - W_{k\to d}/T_{k\to d}) + \Delta(\sigma, W)$, where $W_{k\to d}$ is the best trip time experienced over an adaptive time window, and $\Delta(\sigma, W)$ is a correcting term (the rationale for this choice for r is discussed in [7]); r is used by the current node k as a positive reinforcement for the node f the backward ant $B_{d\to s}$ comes from.

P_{df} is increased by the reinforcement value: $P_{df} \leftarrow P_{df} + (1 - P_{df})\cdot r = P_{df}\cdot(1 - r) + r$. In this way, the probability P_{df} will be increased by a value proportional to the reinforcement received and to the previous value of the node probability (that is, given a same reinforcement, small probability values are increased proportionally more than big probability values).

Probabilities P_{dn} for destination d of the other neighboring nodes n implicitly receive a negative reinforcement by normalization. That is, their values are reduced so that the sum of probabilities will still be 1: $P_{dn} \leftarrow P_{dn}\cdot(1 - r)$.

It is important to remark that every discovered path receives a positive reinforcement in its selection probability. In this way, not only the (explicit) assigned value r plays a role, but also the (implicit) ant's arrival rate.

In Fig. 1 an high-level description of the algorithm is summarized in pseudo-code.

An important aspect of the AntNet algorithm is that the routing tables are used in a probabilistic way not only by the ants, but also by the packets. This mechanism allows an efficient distribution of the data packets over all the good paths. This has been observed to improve AntNet performance. The choice of links with very low probability is avoided by setting for each node a threshold whose value is a function of the node's number of links.

As a last consideration, note the critical role played by ant communication. In fact, each ant is complex enough to solve a single sub-problem but the global routing optimization problem cannot be solved efficiently by a single ant. It is the interaction between the ants that determines the emergence of a global effective behavior from the network performance point of view. The key concept in the cooperative aspect lies in the indirect and non-coordinated way communication among ants happens (stigmergy) [14]. We used stigmergy as a way of recursively transmitting, through the nodes' structures, the information associated with every "experiment" made by each ant (AntNet can be seen as a particular instance of a parallel replicated Monte Carlo simulation).

4. Experimental Settings

We have selected a limited set of classes of tunable components and for each of them we have made realistic choices.

[1] Backward ants do not: they have priority over data to faster propagate the accumulated information.

```
t:= current_time
t_end:=time_length_of_simulation
Δt:=time_interval_for_ants_generation
foreach Node  # Concurrent activity over the network
# s=source node, d=destination node,
# n=next node, c=current node
  while (t≤tend)
    if ((t mod Δt)=0)
      d:=Select_d_Node()
      Launch_Forward_Ant(d)
    endif
    foreach ActiveForwardAnt(s,c,d)
      while(c≠d)
        next_node:=Select_Link_Using_Routing_Table(c,d)
        Put_Ant_On_Link_Queue(c,n)
        Wait_On_Data_Link_Queue(c,n)
        Cross_The_Link(c,n)
        Push_On_The_Stack(n,elapsed_time)
        c:=n
      endwhile
      Launch_Backward_Ant(d,s,stack_data)
      Die()
    endfor
    foreach ActiveBackwardAnt(s,c,d)
      while (c≠d)
        next_node:=Pop_The_Stack(c)
        Wait_On_High_Priority_Link_Queue(c,n)
        Cross_The_Link(c,n)
        Update_Traffic_Model(c,s,stack_data)
        Update_Routing_Table(c,s,stack_data,traffic_model)
      endwhile
    endfor
  endwhile
endfor
```

Fig. 1. AntNet behavior in pseudo-code. Only a top-level description of the algorithm is reported. All the described actions take place in a completely distributed and concurrent way over the network nodes. The processes of data generation and routing are not described, but they can be thought of as acting concurrently with the ants.

Topology and physical properties of the net. In our experiments we used two networks: SimpleNet and NTTnet. SimpleNet is a small network specifically designed to study some aspects of the behavior of the algorithms we compare, while NTTnet (the private NTT backbone) is the major Japanese backbone. SimpleNet is composed of 8 nodes and 9 bi-directional links. NTTnet is composed of 57 nodes and 162 bi-directional links. The topology of SimpleNet and NTTnet are shown in Fig. 2. The transmission delays are of 1 msec for SimpleNet and range from 1 to 5 msec for NTTnet. Links have a bandwidth of 6 Mbit/s in NTTnet, of 10 Mbit/s in SimpleNet. All nets have null link and node fault probabilities, local buffers of 1 Gbit capacity, and packets maximum time to live set to 15 sec.

Traffic patterns. Traffic is defined in terms of open sessions between a pair of active applications situated on different nodes. We considered three basic spatial and temporal traffic pattern distributions:

- Uniform Poisson (UP): for each node is defined an identical Poisson process for sessions arrival, that is, inter arrival times are negative exponential distributed.
- Hot Spots (HS): some nodes behave as hot spots, concentrating a high rate of input/output traffic. Sessions are opened from the hot spots to all the other nodes.
- Constant Bit Rate (CBR): at the beginning of the simulation a fixed number of one-to-all sessions is setup and left constant for the remaining of the simulation. Their packet production rate is fixed.

All the experiments have been realized considering various compositions of the

above main patterns. For the UP and HS cases, the sessions characteristics are: (i) packets sizes and inter arrival times follow a negative exponential distribution, (ii) the total number of bits generated by a session follows the same distribution with mean value fixed to 2 Mbit.

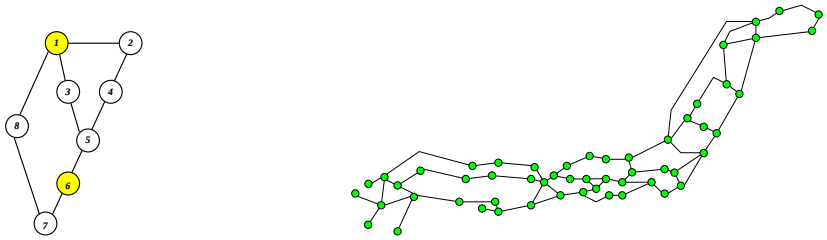

Fig. 2. a) SimpleNet. b) NTTnet. Each edge in the graph represents a pair of directed links.

Metrics for performance evaluation. We used two standard performance metrics: *throughput* (delivered bits/sec), and *data packets delay* (sec). For data packets delay we use either the average value over a moving time window, or the empirical distribution that takes into account the intrinsic variability of packet delays.

Routing algorithms used for comparison. Comparisons were run using the following state-of-the-art algorithms: (i) OSPF is our simplified implementation of the official Internet routing algorithm [16]. (ii) SPF is the prototype of link-state algorithms with dynamic metric for link costs evaluations. A similar algorithm was implemented in the second version of ARPANET [15]. We implemented it with state-of-the-art flooding algorithms and link cost [17] which are evaluated over moving windows using a link usage metric based on the fraction of time the link has been used during the last observation window. (iii) BF is an adaptive implementation of the distributed Bellman-Ford algorithm with dynamic metrics [2]; link costs are evaluated as in SPF. (iv) Q-R is the Q-routing algorithm as proposed in [4]. (v) PQ-R is the Predictive Q-routing algorithm [5], an extension of Q-routing.

Routing algorithms parameters. The sizes of the packets generated by each routing algorithms, in bytes, are: for AntNet $24+8 \cdot N_h$; for SPF and OSPF $64+8 \cdot N_n$; for BF $24+12 \cdot N$; for Q-R and PQ-R 12; where N_h is the incremental number of hops done by the forward ant, N_n is the number of neighbors of node n, and N is the number of network nodes. The packet elaboration times, in msec, are: for AntNet 3; for SPF and OSPF 6; for BF 2; for Q-R and PQ-R 3. The other main parameters are the following. For AntNet, the generation interval of ants is set to 0.3 sec (AntNet performance was experimentally found to be very robust with respect to variations in the value of this parameter, see [7]). For OSPF, SPF, and BF, the length of the time interval between consecutive routing information broadcasting and the length of the time window to average link costs are the same, and they are set to 0.8 or 3 seconds, depending on the experiment. For Q-R and PQ-R the transmission of routing information is data-driven.

5. Results

Experiments reported in this section (averages are over 10 trials) compare AntNet with the previously described routing algorithms. Parameters values for traffic characteristics are given in the figures' captions with the following meaning: NHS is the

number of hot spot nodes, MSIA is the mean of the sessions inter arrival time distribution, MPIA-UP, MPIA-HS, and MPIA-CBR are the mean of the packet inter arrival time distributions for the UP, HS, and CBR sessions respectively. The mean of the packet size distribution is set to 4096 bit in all experiments.

It is important to note that the goal of a routing algorithm is to route all the generated traffic without losses, while keeping packet delays as low as possible (i.e., it should keep the network far from saturation conditions). Since packet losses require retransmissions (these are managed by the transport layer, which is not implemented in our simulator), which in turn cause a traffic increase, results are presented as follows: the first performance comparison will be done on throughput, and a comparison on packet delays is done only for those algorithms which have a similar throughput.

Experiments with SimpleNet were designed to study how the different algorithms manage to distribute the load on the different possible paths. In these experiments all the traffic, of CBR type, is directed from node 1 to node 6 (see Fig. 2a), and the traffic load has been set to a value higher than the capacity of a single link, so that it cannot be efficiently routed on a single path. Results regarding throughput (outer graph in Fig. 3) discriminate between two groups of algorithms: those with a good and approximately equivalent performance, BF, SPF, and AntNet, and those which very rapidly saturate (Q-R, PQ-R, and OSPF). The inner graph in Fig. 3 shows that, on SimpleNet, AntNet is the best algorithm as far as packet delay is concerned. This is a property that AntNet maintains for all the test problems we have used. The second best is SPF, while all the others have a much worse behavior. In conclusion, on this simple net, AntNet is the only algorithm which is able to distribute all the load on the three paths 1–2–4–5–6, 1–3–5–6 and 1–8–7–6 keeping at the same time the average delays at a reasonably low level. The main reasons for this behavior, as discussed in the next section, are to be attributed to the probabilistic routing of both routing and data packets.

Results obtained on the NTTnet for a uniform Poisson traffic load (UP) distribution and for hot spots superimposed to a uniform Poisson traffic load (UPHS) are shown in figures 4 and 5. The uniform Poisson traffic was chosen to be “heavy”, that is, we set the values of the traffic patterns parameters to values that caused the network to reach a state close to saturation. The reason to do this is that it is only under heavy load conditions that differences among competing algorithms can be appreciated in a meaningful way. In fact, when the traffic load is low, almost all the algorithms perform similarly. On the other hand, if the traffic load is too high, then a reasonable assumption is that it is a temporary situation. If it is not, structural changes to the network characteristics, like adding new and faster connection lines, rather than improvements of the routing algorithm, are in order. In both figures 4 and 5, the bigger, outer graph shows the throughput, while the smaller, inner graph shows the empirical distribution of packet delays. From these two figures we can extract the following information: (i) All the algorithms, with the exception of OSPF, can successfully route the totality of the offered throughput, and (ii) AntNet is the algorithm with the best empirical distribution of packet delays.

In Fig. 6 we investigate the answer of the algorithms to a sudden increase of traffic load. During the whole simulation the network is given a uniform Poisson traffic load distribution; at simulation time t=400 four hot spots are switched on, to be subsequently switched off at time t=520. The graphs in Fig. 6 show the instantaneous throughput (upper graph) and the instantaneous packet delay (lower graph) averaged over a moving time window of 5 sec.

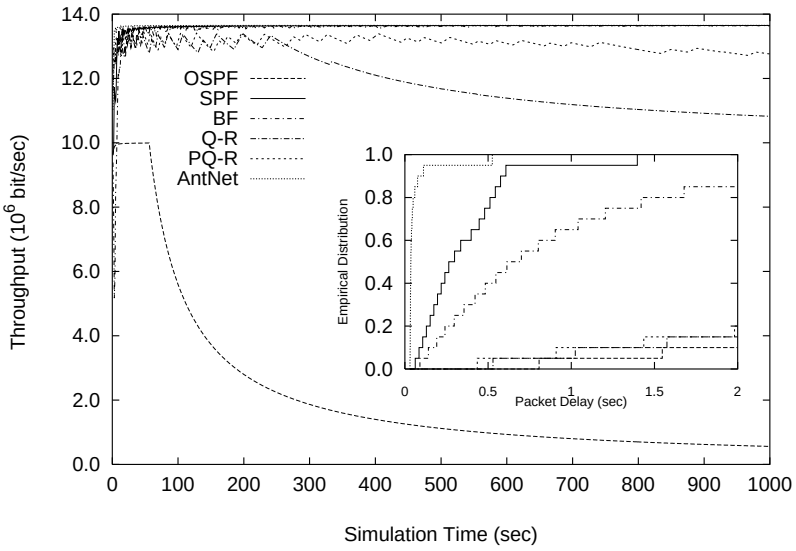

Fig. 3. SimpleNet: A comparison of AntNet with five competing algorithms with constant traffic (CBR) from node 1 to 6. Average over 10 trials. MPIA-CBR=0.0003. Throughput: AntNet, SPF, and BF have very good and similar performance. Packet delays: AntNet is the only algorithm able to keep all delays under 0.6 sec.

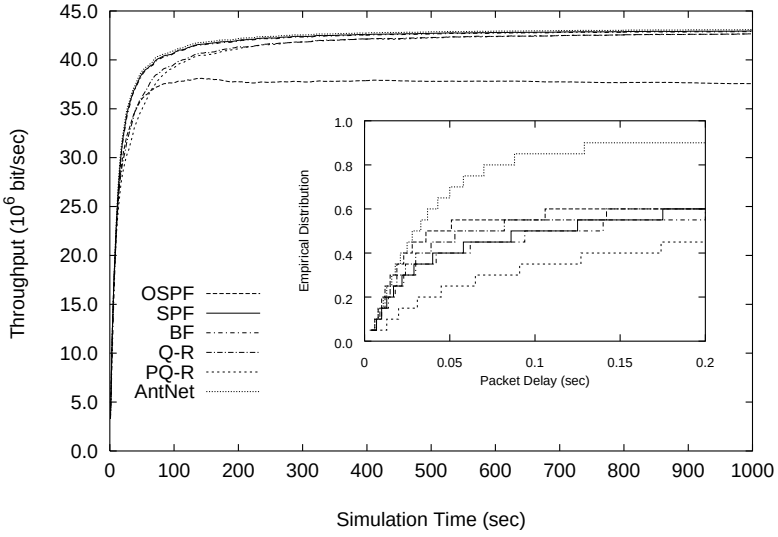

Fig. 4. NTTnet: A comparison of AntNet with five competing algorithms for heavy traffic conditions and uniform Poisson traffic (UP) distribution. Average over 10 trials. MSIA=1.5, MPIA-UP=0.2. Throughput: all algorithms, but OSPF, have very good and similar performance. Packet delays: AntNet is the only algorithm able to keep more than 90% of the delays under 0.15 sec.

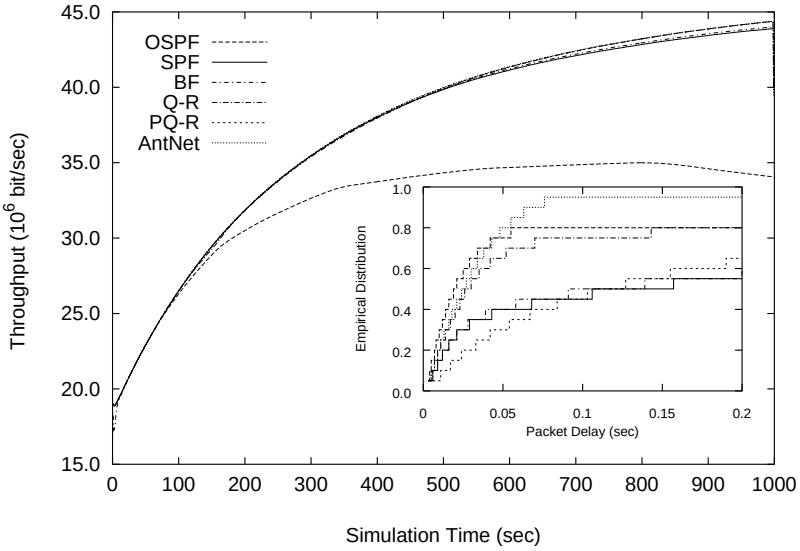

Fig. 5. NTTnet: A comparison of AntNet with five competing algorithms for heavy traffic conditions and hot spots superimposed to a UP traffic (UPHS). Average over 10 trials. NHS=4, MSIA=3.8, MPIA-UP=0.3, MPIA-HS=0.05. Throughput: all algorithms, but OSPF, have very good and similar performance. Packet delays: AntNet is the only algorithm able to keep more than 90% of the delays under 0.1 sec.

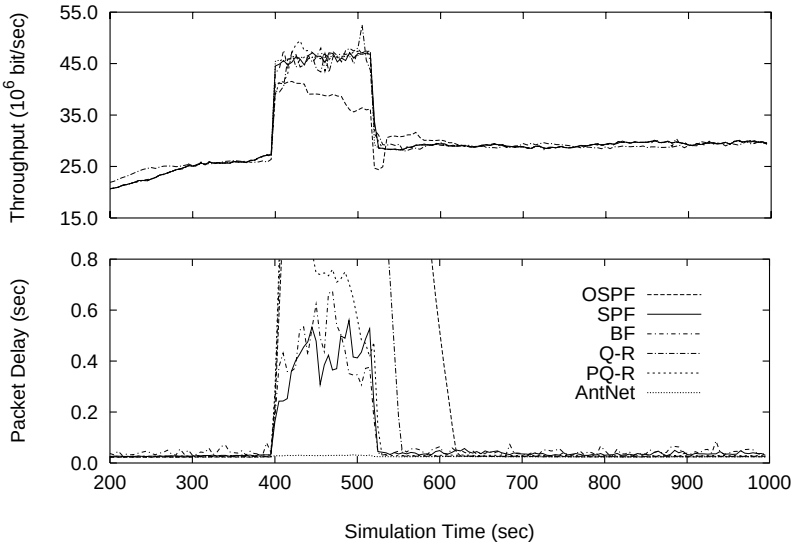

Fig. 6. NTTnet: A comparison of AntNet with five competing algorithms: after 400 sec some hot spots are superimposed for 120 sec to a UP traffic causing a sudden surge in traffic load. We report instantaneous values for throughput and packet delays based on statistics collected over 5 sec time windows. Average over 10 trials. NHS=4, MSIA=4.0, MPIA-UP=0.3, MPIA-HS=0.05. Throughput: all algorithms, but OSPF, have a very good and similar response to the transitory. Instantaneous packet delays: AntNet maintains the delays during the surge in traffic load at a level very similar to standard load conditions. Some of the competing algorithms (OSPF, Q-R, and PQ-R) reacted so badly that their graphs are out of scale.

Observing the upper graph, we can see that all the algorithms increase their throughput when the hot spots are switched on, and quickly forget the transitory situation once the hot spots are switched off. The only significant difference in throughput among algorithms is OSPF lower performance. The lower graph confirms AntNet better management of packet delays also in this case: after time t=400 the load surge causes the average packet delay to greatly increase for all the algorithms, except for AntNet, which maintains packet delays at a much lower level.

6. Discussion

The results presented are rather sharp: AntNet performs better, both in terms of throughput and of average delay, than both classic and recently proposed algorithms. (Experiments similar to those presented in this paper have been run on other network topologies with increasing number of nodes and different traffic patterns obtaining similar results (see [7, 8, 9]).) Among the competitors there is not a clear winner, although OSPF seemed to have a lower performance than the others. Concerning network resources utilization, in Table 2 we report, for each algorithm, the routing overhead expressed as the ratio between generated routing traffic and total available network bandwidth. Even if AntNet's overhead is higher than that of some of its competitors, it must be considered that (i) the relative weight of the routing packets on the net resources is negligible, and (ii) this slightly higher network resources consumption is compensated by the much higher performance it provides.

Table 2. Routing overhead for experimental conditions considered in the paper expressed as the ratio between the generated routing traffic by each algorithm and total available network bandwidth (note: the ratios are scaled by 10^{-3}).

	AntNet	OSPF	SPF	BF	Q-R	PQ-R
SimpleNet (10^{-3})	0.20	0.01	0.10	0.06	2.3	2.6
NTTnet UP (10^{-3})	2.85	0.14	3.68	1.39	3.72	6.77
NTTnet UPHS (10^{-3})	3.81	0.15	4.56	1.39	3.09	4.81

Differences among algorithms performances can be understood on the basis of the different degree of adaptivity and of speed with which the different algorithms respond to changing traffic conditions. The very low performance of OSPF can be explained by both the lack of use of an adaptive metric (which all the other methods use), and by the fact that we set link costs only on the basis of a shortest path computation. Differently, on real networks (e.g. on the Internet) these are set by network administrators who use additional heuristic knowledge about traffic patterns. To explain why AntNet performs better than the others is slightly more tricky. We identified the following main reasons: (i) the deterministic versus probabilistic use of routing tables to route data packets, (ii) the use of local versus global information, and (iii) the different routing table update frequencies. These are discussed in the following.

(i) All the tested algorithms but AntNet use deterministic routing tables. In these algorithms, entries in the routing tables contain distance/time estimates to the destinations. These estimates can provide misleading information if the algorithm is not fast enough to follow the traffic fluctuations, as is the case under heavy load conditions. Differently, AntNet routing tables have probabilistic entries that, although reflecting the goodness of a particular path choice with respect to the others available, do not force the data packets to choose the perceived best path. This has the positive effect to allow a better balancing of the traffic load on dif-

ferent paths, with a resulting better utilization of the resources (as shown in the experiments with SimpleNet). Probabilistic routing tables provide some remarkable additional benefits: (i) they give to the ants a built-in exploration method in discovering new competing paths, (ii) the information they contain, being non-metric, doesn't depend in an absolute way on the current topological and physical characteristics of the network, and (iii) since ants and data routing are decoupled in AntNet, the exploration of new routes can continue while, at the same time, data packet can exploit previously learnt, reliable information.

(ii) BF, Q-R and PQ-R work with local estimates of distances to destinations. These estimates are updated by using strictly local information, that is: the traffic situation on outgoing links, and the distance estimates maintained by neighbor nodes. Differently, AntNet samples the network and redistributes the global information ants collect: backward ants redistribute the global information relative to the paths sampled by the corresponding forward ants to all the nodes they visited. SPF maintains a global representation of the whole network in each node, which is updated by periodic flooding of local link costs information. If one of this cost information is badly estimated (as it is often the case when dynamic metrics are used), the wrong estimate propagates to all the local representations of the network. Here it is used to calculate shortest paths to build the new routing tables. The result is that a single erroneous estimate will negatively affect all the routing tables. From this point of view, AntNet is more robust: an incorrect update will affect only entries relative to the ant destination in those routing tables belonging to the ant path.

(iii) In BF and SPF the broadcast frequency of routing information plays a critical role, particularly so for BF which has only a local representation of the network status. This frequency is unfortunately problem dependent, and there is no easy way to make it adaptive, while, at the same time, avoiding large oscillations. In Q-R and PQ-R, routing tables updating is datadriven: only those Q-values belonging to pairs (i,j) of nodes visited by packets are updated. Although this is a reasonable strategy, given that the exploration of new routes could cause undesired delays to data packets, this causes delays in discovering new good routes, and is a great handicap in a domain where good routes change all the time. In AntNet, we experimentally observed the robustness to changes in the ants' generation rate. For a wide range of values of the generation rate, the more the ants generated, the better the algorithm works, until the traffic induced by ants ceases to be negligible with respect to the data traffic [7].

7. Conclusions

In this paper we introduced a novel algorithm for routing in communications networks, called AntNet. AntNet was inspired by previous work on artificial ants colonies in combinatorial optimization [11, 12, 13]. We compared AntNet to a set of state-of-the-art algorithms using a realistic network simulator, a simple network of 8 nodes, and the NTT corporate network of 57 nodes.

Experimental results showed that AntNet had the best distribution of packet delays, and was among the best algorithms as far as throughput was concerned for all the experimental conditions we tested. AntNet showed a robust behavior under the different traffic conditions and the ability to reach a stable behavior very quickly. It also had, as the other algorithms used for comparison, a negligible impact on the use

of network bandwidth.

Acknowledgments

This work was supported by a Madame Curie Fellowship awarded to Gianni Di Caro (CEC-TMR Contract N. ERBFMBICT 961153). Marco Dorigo is a Research Associate with the Belgian FNRS.

References

[1] Beckers R., Deneubourg J.L., & Goss S. 1992. Trails and U-turns in the Selection of the Shortest Path by the Ant Lasius Niger. *J. of Theor. Biology* 159:397–415.

[2] Bertsekas D. & Gallager R. 1992. *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall.

[3] Bonabeau E., Dorigo M. & Theraulaz G. (in press). *From Natural to Artificial Swarm Intelligence*. Oxford University Press.

[4] Boyan J. A. & Littman M. L. 1994. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. In *Proc. of NIPS-6*, San Francisco, CA: Morgan Kaufmann, 671–678.

[5] Choi S.P.M. & Yeung D.-Y. 1996. Predictive Q-Routing: A Memory-based Reinforcement Learning Approach to Adaptive Traffic Control. In *Proc. of NIPS-8*, Cambridge, MA: The MIT Press, 945–910.

[6] Costa D. & Hertz A. 1997. Ants Can Colour Graphs. *J. of Oper. Res. Soc.* 48:295-305.

[7] Di Caro G. & Dorigo M. 1998. AntNet: Distributed Stigmergetic Control for Communications Networks. *Tech.Rep. IRIDIA/98-01*, Univ. Libre de Bruxelles, Belgium.

[8] Di Caro G. & M. Dorigo 1998. Distributed Adaptive Routing by Artificial Ant Colonies. *PDCS'98 – 1998 International Conference on Parallel and Distributed Computing and Systems*, October 28–31, 1998, Las Vegas, Nevada.

[9] Di Caro G. & M. Dorigo 1998. Ant colonies for Adaptive Routing in Packet-switched Communications Networks. *Proceedings of PPSN V–Fifth International Conference on Parallel Problem Solving From Nature*, 27–30 September 1998, Amsterdam, NL, in press.

[10] Dorigo M. 1992. *Ottimizzazione, Apprendimento Automatico, ed Algoritmi Basati su Metafora Naturale* (*Optimization, Learning and Natural Algorithms*). Ph.D.Thesis, Politecnico di Milano, Italy (in Italian), pp.140.

[11] Dorigo M. & Gambardella L.M. 1997. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Trans. on Evol. Comp.* 1(1): 53–66.

[12] Dorigo M., Maniezzo V., & Colorni A. 1991. Positive Feedback as a Search Strategy. *Tech. Rep. No. 91-016*, Politecnico di Milano, Italy

[13] Dorigo M., Maniezzo V., & Colorni A. 1996. The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Trans. on Sys., Man, and Cyb.–Part B* 26(2): 29–41.

[14] Grassé P. P. 1959. La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalensis et Cubitermes sp*. La théorie de la stigmergie: Essai d'interprétation des termites constructeurs. *Insect Sociaux* 6: 41–83.

[15] McQuillan J. M., Richer I., & Rosen E. C. 1980. The New Routing Algorithm for the ARPANET. *IEEE Trans. on Comm.* 28:711–719.

[16] Moy J. 1994. OSPF Version 2. Request for Comments (RFC) 1583, Network Work Group.

[17] Shankar A.U., Alaettinoglu C., Dussa-Zieger K., & Matta I. 1992. Performance Comparison of Routing Protocols under Dynamic and Static File Transfer Connections. *ACM SIGCOMM Comp. Comm. Review* 22(5): 39–52.

[18] Steenstrup M. E. (ed.) 1995. *Routing in Communications Networks*. Englewood Cliffs, NJ: Prentice-Hall.

[19] Stone P. & Veloso M. 1996. Multiagent Systems: A Survey from a Machine Learning Perspective. *Tech. Rep. CMU-CS-97-193*, Carnegie Mellon University, PA.