Implementation and evaluation of AntNet, a distributed shortest-path algorithm

Matthias Strobbe, Vincent Verstraete, Erik Van Breusegem,
Jan Coppens, Mario Pickavet, Piet Demeester
Department of Information Technology (INTEC)
Ghent University - IMEC
Sint-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
+32 9 3314900
{matthias.strobbe, vincent.verstraete}@intec.ugent.be

Abstract

In the last decade, the OSPF routing protocol has proven its robustness, but it also has some major drawbacks. It reacts quite slowly to network changes, and load balancing is limited to paths with equal cost (ECMP). In this paper, we present our implementation and evaluation of an alternative: AntNet [1]. AntNet is a distributed shortest-path algorithm based on the principles of Ant Colony Optimization that takes care of load balancing in a very natural way.

1. Introduction

As the ICT world is evolving towards a completely network-driven infrastructure, more and more attention is paid to the underlying network technology and protocols. OSPF, Open Shortest Path First, is one of the most widely used routing protocols. Nevertheless, it has some drawbacks and a lot of research focused on finding alternatives. Genetic algorithms, neural networks, etc. have all been used to adapt the existing protocols or to invent new ones. Very promising are protocols based on mobile agents. This paper focuses on such a protocol: AntNet.

The purpose of a good routing protocol is to minimize a certain metric. OSPF e.g. minimizes a static link cost. As a consequence, OSPF cannot react to dynamic phenomena such as congestion. AntNet on the other hand introduces a dynamic, probabilistic approach. The user latency is constantly measured and AntNet converges to the paths with the lowest latency. AntNet also takes link utilization into account, so suboptimal paths are used as well.

AntNet is based on the principles of Ant Colony Optimization (ACO). ACO studies the behaviour of ants in a colony, and mimics this behaviour in software. It is amazing how ants, with a minimum of intelligence, succeed in accomplishing quite complex tasks. An example of this is food collection. Initially, all the ants take random paths

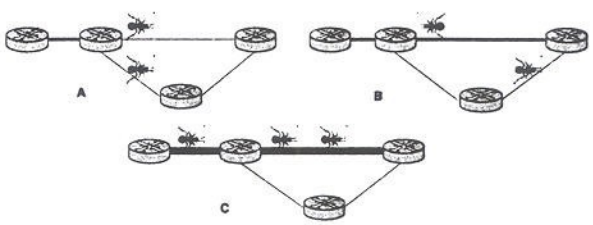

Figure 1. AntNet principle

to the food source(A). During their search, the ants drop slowly dissipating pheromone trails, that attract other ants. The shorter the path, the faster the ants can travel back and forth between the food source and the nest, and the stronger the pheromone trail gets(B), attracting other ants (positive feedback loop). Finally, we get an 'ant highway' following the shortest path(C).

ACO has been applied to many domains, e.g. the Travelling Salesmen Problem[2], the calculation of equation arrays[3], the University Course Timetabling Problem[4], manufacturing control systems[5], data-mining[6], etc. A good overview of the ACO meta-heuristic and a number of applications can be found in [7]. The most straightforward application though, is routing. Objects (persons, packets, etc.) need to go from a source to a destination in an efficient way. [8] gives an extensive overview of the application of ACO for routing and load-balancing. In this paper we focus on AntNet[1], the routing algorithm developed by M. Dorigo and G. Di Caro, based on the ACO principle. They simulated it, and concluded that AntNet scores better than OSPF in both throughput and adaptivity. To prove

0-7695-2388-9/05 $20.00 © 2005 IEEE

practical applicability, our goal was to implement AntNet on a physical network. It turned out that, due to some practical problems, we had to make a few adaptations to the original algorithm. In the next section, we present the resulting algorithm and the required data structures. Section 3 describes our experiments and compares AntNet to OSPF. Finally, in section 5, we state our conclusions.

2 Architecture

2.1 Data Structures

In the OSPF protocol, every router gathers information about the entire network, and uses the Dijkstra algorithm to calculate the best path to each destination. For every destination, the interface and gateway to this path are saved in the routing table. AntNet abandons this deterministic way of routing and introduces a probabilistic approach. A big advantage is that no router needs an overview of the entire network. AntNet however, requires three data structures in every router instead of two:

- a routing table (similar to OSPF)
- a link state database (similar to OSPF)
- a statistical model

For every destination, the data traffic is distributed over the different interfaces. Every entry in an OSPF routing table thus results in N entries in an AntNet routing table, with N the number of interfaces. The interface to the best path still has the highest probability, but the other interfaces do not necessarily have probability 0. The link state database of AntNet is smaller than the database of OSPF, as an AntNet router only needs to keep information about the links between itself and the adjacent routers, and not about all the routers in the network. Finally, the statistical model contains the mean value and the standard deviation of the trip times to every destination. These values will be used as reference values.

2.2 The Adapted AntNet Algorithm

There are two types of mobile agents: Forward Ants and Backward Ants. Forward Ants gather information. On a regular time base, every router sends one Forward Ant with a random destination over the network. This Forward Ant is forwarded by some intermediate routers to its final destination, in a way that balances between exploitation of known good paths and the exploration of new, possibly better, paths. This is accomplished by the 'exploration probability' parameter. As Forward Ants pass through the network, they save information about the intermediate routers on an

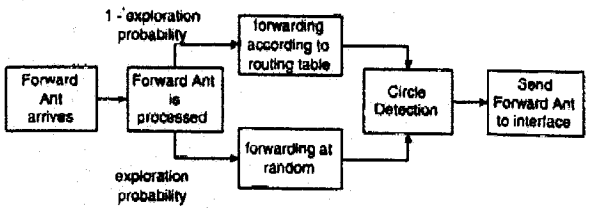

Figure 2. Flowchart Forward Ants

internal stack. The scheme of Fig. 2 shows the processing of Forward Ants within a router.

The other kind of mobile agents are the Backward Ants. Backward Ants are created out of Forward Ants, once they have reached their destination. They inherit all the information on the internal stack of a Forward Ant. The Backward Ant follows exactly the same path as the Forward Ant, but in the opposite direction. In all the intermediate routers, the information of the Backward Ant is compared to the corresponding entry in the statistical model. The result of this comparison is used to adapt the probabilities in the routing table, as well as the statistical model itself. Once the Backward Ant arrives in the starting router, it is discarded (after adapting the routing table and the statistical model).

Backward Ants have a higher priority than data packets. The sooner they are processed, the sooner the extra information is taken into account. Forward Ants cannot have a higher priority. They need to suffer the same network delay as the data packets to be able to measure the network congestion.

2.2.1 Adaptation of the Statistical Model

We used the formulae, as proposed by Baran and Sosa in [9]. Only for the calculation of r (see further), we followed the original specification of Dorigo and Di Caro[1].

$$\mu_{d'}^{i+1} = \mu_{d'}^{i} + a \times (T - \mu_{d'})$$
$$s_{d'}^{i+1} = s_{d'}^{i} + a \times (|T - \mu_{d'}| - s_{d'})$$

T is the measured trip time. $\mu_{d'}$ and $s_{d'}$ are resp. the mean trip time and its standard deviation. Parameter a indicates how strong the influence of a trip time is on the values in the statistical model.

2.2.2 Adaptation of the Routing Table

When a Backward Ant arrives in a router, the probability is increased for the link that leads to the router from where the Backward Ant came. This is done for all destinations d' between the current router and the starting point of the Backward Ant:

$$P_{d'f}^{i+1} = 1 - r \times (1 - P_{d'f}^{i})$$

f is the ID of the next hop, and $P_{d'f}$ the probabilities that are to be increased. The probabilities of the other links are decreased:

$$P_{d'j}^{i+1} = r \times P_{d'j}^{i}$$

Parameter r indicates how much the probabilities have to be changed. r depends not only on the value of the measured trip time (compared to the reference values in the statistical model), but also on the stability of the current situation. Six parameters are used in the calculation of r, the discussion of which we left out due to space limitations. Table 1 however, gives a resume of all the parameters.

We have implemented this algorithm using Click[10]. One of the biggest problems we noticed during the implementation process, was the synchronization of the internal clocks of the routers. The solution was offered by an abstraction mechanism: the times that are saved onto the internal queue of the Forward Ants, are not computed as the difference between two timestamps. Instead, they are computed as the sum of two terms, one representing the delay due to link load, and one for the router load.

Router load In the Click configuration, the time is measured between the arrival of a packet and the moment it is actually processed. This time gives an idea of the load of the router.

Link load On every interface, the byte rate of the incoming packets is measured. This is compared to the capacity of the link, and a load factor is computed:

$$load\ factor = exp(\frac{byte\ rate - threshold\ capacity}{maximum\ capacity - threshold\ capacity}) \quad (1)$$

We define the threshold as being 80% of the capacity. The load factor then varies between 1 and 2.7. If the measured byte rate is lower than the threshold, the load factor is 1. The abstract time is multiplied by this load factor. A path with a heavily loaded link will become less attractive and the load will be balanced over multiple paths.

The introduction of this 'abstract time' has also solved an important performance issue. Ants are sent over the network in a probabilistic manner. As a result, the path of an Ant will often contain circles. This is useless information, and it is better not to take this detour into account. If absolute time stamps are used, the deletion of a circle will require the (quite complex) adaptation of timestamps. Abstract time is calculated as a sum of terms, and therefore it suffices to leave out the terms associated with the circle.

In the basic AntNet protocol, adaptations of the network required the routers to be offline, and configuration files had to be adjusted manually. To avoid this we extended AntNet with a hello protocol. This allows the existing routers to change their configuration dynamically, and a new router automatically learns the existing network configuration. The protocol updates the routing tables and local link state databases of the routers and a first estimation of the trip times to new destinations is provided. This hello protocol is also based on the AntNet principle, i.e. with forward and backward hello ants combined with flooding. The discussion of the concrete protocol is left out due to space limitations.

3 Experimental Results

3.1 Evaluation of AntNet

We evaluated Antnet, using the test network shown in figure 3. This network consists of 5 Click routers and 2 hosts. The hosts are 2 interfaces of a SmartBits 6000B machine[11]. All links are of equal, unit cost.

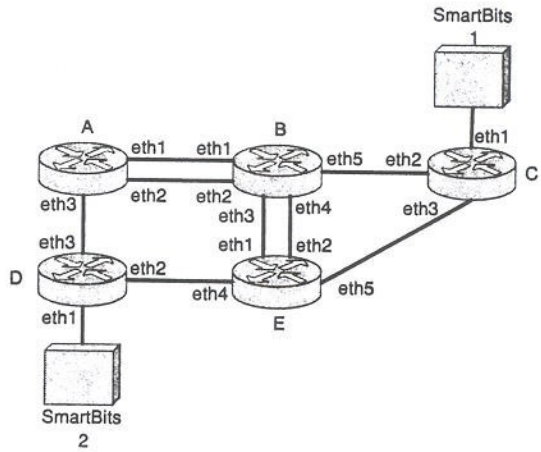

Figure 3. Test Network

3.1.1 Optimization of the Parameters

The first step was to optimize the parameters. We tuned them sequentially, always letting only 1 parameter vary. In every experiment, the convergence time of the probabilities in 1 routing table was measured. After optimization, we obtained the parameter values of table 1. The convergence was about 10 times faster than with the values of [1].

3.1.2 Evaluation and Illustration in Different Situations

We evaluated AntNet in 4 different situations and compared the obtained convergence times (Table 2).

Table 1. Parameters used in the AntNet algorithm

Parameter	Influence	Optimal Value
a	Determines influence on model	0.1
c	Determines start value for r	2
eps	Boundary between stable and unstable situation	0.5
b1	Correction factor (stable situation)	4
b2	Correction factor (unstable situation)	1
t	Good or bad r?	$= 1/c$
h	Constrains r approx. to the interval [0.8, 1]	0.15
ep	Exploration probability	0.2
far	Forward ant rate	3 per sec

Table 2. Comparison of 4 scenarios

Experiment	99% convergence(s)
Convergence of the 5 routers, all started simultaneously.	21,333
Link failure after convergence. The routers are started simultaneously and after 3 minutes a link failure occurs. The time is measured until an alternative path is found.	75
Insertion of a new router. One router is started 3 minutes later. The convergence times in this router are measured.	40,167
Convergence to a new shorter path. Again 1 router is started later. This router creates a new, shorter path. The time is measured until this better path is used.	47

The convergence took by far the longest time in the 2nd experiment. The reason for this, is that the network had already completely converged at that time. Ants are mainly sent over the broken link, and only the 'exploring' Ants reach their destination. Moreover, as the paths over those links will be longer than the original (but now broken) path, the probabilities will not increase that much for the first few ants. This explains the long convergence time. The same reason explains the difference between the convergence time of experiments 1 and 3. In experiment 4, the times are a lot better because an Ant that follows the new, shorter path, greatly boosts the probability of that interface.

Figure 4 gives an illustration of AntNet. It shows how the probabilities in the routing table of router Router B evolve for destination Router C in the case of a link failure. Initially we see the convergence to the shortest path (via eth5). Every now and then, the probability increases on suboptimal paths (via eth3 and eth4). This is a consequence of the dynamic character of AntNet: paths that are just a little longer than the shortest path are also interesting. After 3 minutes we introduce a link failure on eth5 and the probabilities converge to the alternative path via eth3 and eth4. As we have 2 parallel paths, the probabilities constantly alternate between those paths. By adjusting parameter h, a more even distribution can be achieved, but this slows down the convergence. This parameter can thus be seen as a policy parameter. A network operator can weigh up good load balancing and good adaptivity of the network.

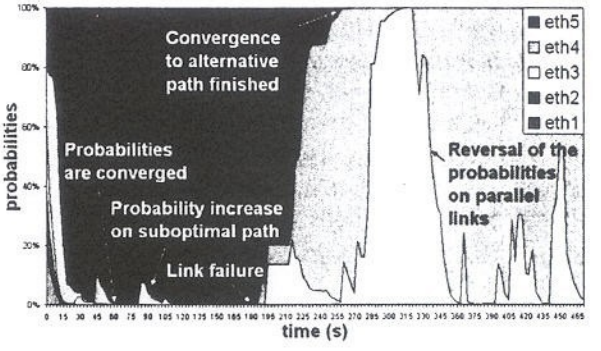

Figure 4. Probabilities with link failure

AntNet inherently uses load balancing as suboptimal paths can also get reasonable probabilities. This property can be amplified or tempered by a good choice of the threshold value in formula (1). A low value of this parameter results in getting a good load balance and protection against bursts, but the user latency and the number of out-of-order packets will increase. A higher value lessens the load balance and decreases the out-of-order rate. So, this threshold value can also be seen as a policy parameter. As stated before, we have used 80% of the link capacity as threshold value in our experiments.

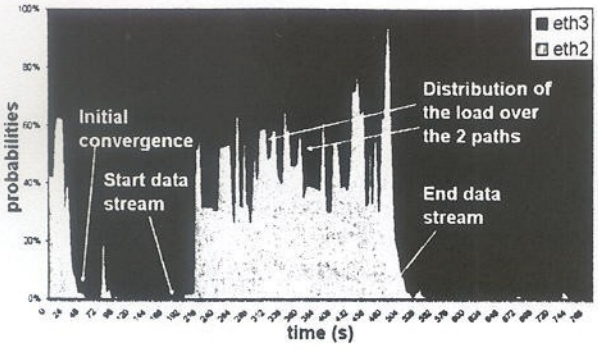

Figure 5. Probabilities with a link load of 200%

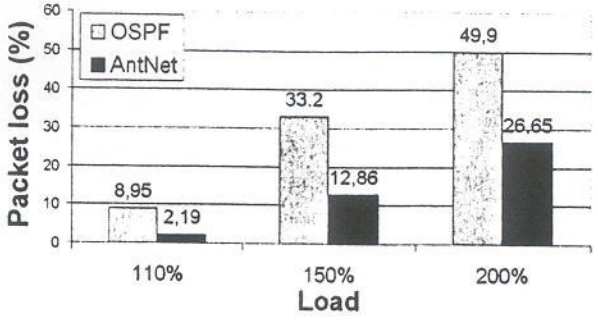

Figure 6. OSPF vs AntNet: throughput

Figure 5 illustrates the distribution of a very heavy load (twice the capacity of 1 link) over 2 paths. Traffic is sent from SmartBits 1 to SmartBits 2, and the probabilities in Router C are monitored for destination SmartBits 2. The shortest path is 3 hops (via eth3). The alternative path (via eth2) is 4 hops. The load is nicely distributed over the 2 paths.

3.1.3 Load

The real purpose of a network is the transportation of data traffic and not of control traffic. It is important that the consumption of bandwidth by the routing protocol is almost negligible for the protocol to be economically interesting. Our experiments showed that this is indeed the case, even with a forward ant rate of 3 ants/s per router. Moreover, the consumption of bandwidth is independent of topological changes in the network which is not the case for OSPF.

3.2 OSPF vs. AntNet

In a second series of experiments we compared the throughput and adaptivity of AntNet and OSPF. We used the Zebra software[12] for the OSPF tests.

3.2.1 Throughput

AntNet distributes a heavy load over several paths. As a result, higher throughputs are possible than with standard OSPF. To measure the difference, we generated a heavy load (higher than the capacity of 1 link) and measured the number of lost packets. Figure 6 shows the results. AntNet performs a lot better than OSPF. With OSPF the surplus of packets is completely lost whereas AntNet succeeds in forwarding a lot of these packets to their destination. With OSPF it is often the case that some links in a network are heavily loaded, while others are almost not used. AntNet is more network optimal as it uses the capacity of the whole network in a more efficient way.

3.2.2 Adaptivity

To measure the adaptivity, we tested how long it took before the network converges after a link failure or the addition of a new link. Especially the case of link failure is important. Unfortunately, in that case AntNet does not perform as well as OSPF. In our tests, it took about 45s for OSPF to switch to the alternative path whereas AntNet needed 112s. The reason is that it takes quite some time before enough ants have followed the alternative (long and therefore bad) path. Luckily, it is possible to extend the protocol with mechanisms to detect link failures locally. This technique allows for a very fast detection of link failures and provides a solution to the high convergence times.

When a new link was added (shorter path), AntNet performed well. OSPF needed 21s to use the new link whereas AntNet needed only 17s. However, as no data can be lost by adding a new link, this difference is less crucial.

4 Future Work

AntNet has 9 parameters. Some can be seen as policy parameters (e.g. the mentioned parameter h or the threshold value of formula1). It would be interesting if the other parameters could be tuned automatically, possibly taking the state of the network (stable, unstable) into account.

Another possible extension is the separation of the data and control routing tables. In the current implementation the probabilities continuously vary. For data packets it's better if the probabilities only change from time to time when something essential changes in the network. A solution is to separate the data and control routing tables with the ants only updating the control routing table. Now and then this control routing table is mapped on the data routing table according to some rules set by the network operator. For example when the system notices that the probabilities

constantly alternate between 2 paths (like e.g. in Figure 4 after the link failure), this means 2 parallel paths to the destination exist. To get a maximal throughput, the probabilities of the 2 interfaces on these paths in the data routing table preferably get a value of 50%. A possible rule for this phenomenon would be: 'If a link between 2 mappings reaches a probability of at least 60%, this link belongs to a good route. Spread the total probability over the good routes.'

AntNet can also be applied in other domains. Possibilities are e.g. a BGP like protocol based on AntNet or the application of AntNet in peer-to-peer networks. An example of the latter is MUTE [13], an anonymous file sharing system. Finally AntNet can also be applied to the type of networks which are becoming more and more available: mobile ad hoc networks. In [14] and [15] the combination of mobile agents and ad hoc networks is already simulated.

5 Conclusions

Up until now, AntNet was only simulated ([1], [9]). In this paper we described our implementation of AntNet on a physical network by using the modular Click software. After implementation we tested the algorithm thoroughly with the following conclusions as a result:

- When it comes to throughput, AntNet exceeds OSPF by far. The difference is caused by the load balancing inherent to AntNet. The probabilistic way of routing makes it possible to distribute a data stream over several optimal and suboptimal paths. OSPF uses only the shortest path.
- We found no clear winner with respect to adaptivity, in contrast to the statements in [1] and [9]. AntNet reacts faster to the introduction of new links, but in the (more important) case of link failures, OSPF still performs better. The incorporation of a mechanism to detect link failures locally should solve this problem, resulting in a similar adaptivity of AntNet and OSPF.

References

[1] Di Caro, G., Dorigo, M.: AntNet: A Mobile Agents Approach to Adaptive Routing. *Technical Report IRIDIA*, 1997.

[2] Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transaction on Evolutionary Computation*, 1(1):53–66, 1997.

[3] Solnon, C.: Ants can solve constraint satisfaction problems. *IEEE Transactions on Evolutionary Computation*, 6(4):347–357, 2002.

[4] Socha, K., Sampels, M., Manfrin, M.: Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. *Proceedings of EvoCOP 2003 - 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science*, 2611:334–345, 2003.

[5] Karuna, H., Valckenaers P., Zamfirescu, C.B., Van Brussel, H., Saint Germain, B., Holvoet, T., Steegmans, E.: Self-Organising in Multi-Agent Coordination and Control Using Stigmergy. *Engineering Self-Organising Applications*, 105–123, 2003.

[6] Ramos, V., Abraham, A.: Evolving a Stigmergic Self-Organized Data-Mining. *Proceedings of ISDA-04, 4th Int. Conf. on Intelligent Systems, Design and Applications*, 725–730, 2004, Hungary.

[7] Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(2):137–172, 1999.

[8] Sim, K.M., Sun, W.H.: Ant Colony Optimization for Routing and Load-Balancing: Survey and New Directions. *IEEE Transaction on Systems, Man and Cybernetics, Part A. Special Issue on Collective Intelligence*, 33(5):560–572, 2003.

[9] Baran, B., Sosa, R.: AntNet: Routing algorithm for data networks based on mobile agents. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial*, 3(12):75–84, 2001.

[10] Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000.

[11] SmartBits. http://www.cs.ucsb.edu/ almeroth/classes/193/W01-shadow/smartbits.html

[12] Gnu zebra. http://www.zebra.org/

[13] Mute: Simple, anonymous file sharing. http://mute-net.sourceforge.net/

[14] Matsuo, H., Mori, K.: Accelerated Ants Routing in Dynamic Networks. *Proceedings Int. Conf. on Software Engineering, Artificial Intellingence, Networking and Parallel/Distributed Computing*, 333–339, 2001.

[15] Shivanajay, M., Tham, C.K., Srinivasan, D.: Mobile Agents based Routing Protocol for Mobile Ad Hoc Networks. *Proceedings IEEE Globecom*, 2002, Taiwan.

TELECOMMUNICATIONS 2005

Lisbon, Portugal
July 17-20, 2005

Editors

Petre Dini
Pascal Lorenz
Mario Freire
Pierre Rolin
Pawel Szulakiewicz
Alexandra Cristea

IEEE

IEEE Portugal

IARIA

Published by the IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Order Number P2388
Library of Congress Number 2005925129
ISBN 0-7695-2388-9