Intelligent Ants for Adaptive Network Routing

H. Yun
Dalhousie University
yun@cs.dal.ca

A. N. Zincir-Heywood
Dalhosuie University
zincir@cs.dal.ca

Abstract

An adaptive routing algorithm based on AntNet algorithm is designed and implemented with a new routing table formation scheme. This addresses the unrealistic requirement for global information of the original AntNet algorithm. The new algorithm requires limited routing information in the routing and traffic statistics tables. The routers only have the most popular destinations in their routing tables and update these destinations at a scheduled time. Each router keeps a traffic table to record data packets visiting. Under this approach, a data packet will be forwarded randomly if its destination does not exist in the routing table. Unlike the original AntNet algorithm, ants and data packets will have time flags to avoid them having infinite lives. Experiments show that the new approach gives promising results.

1. Introduction

Routing is the process used to determine the path by which a packet travels from source to destination. Network information systems and telecommunication in general rely on a combination of routing strategies and protocols to ensure that information sent by a user is actually received at the desired remote location. The growing size and increasing demands placed on packet switched networks has pushed their application into areas not necessarily considered at their conception. As a consequence, routing techniques currently in operation are increasingly being shown to be ineffective [7].

Routing strategies currently in widespread use (e.g. OSPF, RIP, and BGP) are implemented through the information contained in routing tables independently available at each node in the network [5]. Moreover, in all the above cases the content of such tables consists of specific entries for the neighboring nodes and then a series of default paths for packets with any other destination [5]. The manner in which this information is gathered has implications for the ability of the respective algorithms to respond to changing load conditions or changes to network topology. Specific properties of this problem, which make it particularly challenging, include the distributed nature; hence a solution that assumes access to any form of global information is not desirable. The problem is also dynamic; hence a solution that is sufficient for presently experienced network conditions may well be inefficient under other loads experienced by the network.

Several approaches have been proposed for addressing the above objectives including: active networking [10], social insect metaphors [1, 4, 6] and what might be loosely called other "adaptive" techniques (e.g. evolutionary computation [3], neural networks [2]). The latter typically involve using evolutionary or neural techniques to produce a 'routing controller' as opposed to a 'routing table' at each node, where the controller may require knowledge of the global connectivity to ensure a valid route. All methods as currently implemented, however, suffer from one drawback or another. Cognitive packet networks and active networking algorithms attempt to provide routing protocols at the packet level, hence achieving scalable run time efficiency becomes an issue. Implementations of "adaptive" techniques or social insect metaphors – AntNet – frequently rely on the availability of global information [4, 9].

The purpose of this work is to investigate the use of social insect metaphors to solve the "adaptive" routing problem without relying on the availability of *a priori* global information. In particular, the objective is to dynamically identify (learning to adapt to the network load) what destinations should be included in the routing table. This is a novel adaptive routing technique for data networks, whose use is currently oriented towards packet switching networks, such as the Internet. We believe that such an approach will not only impact the area of adaptive routing, but also can complement the existing routing protocols in terms of scalability properties. In the following, section 2 introduces the "AntNet" based social insect metaphor against which the proposed approach is compared. Section 3 introduces the proposed modified scheme for "Antnet" to avoid the use of *a priori* global information. Section 4 summarizes

the network on which experiments are performed. Results are presented in section 5 and conclusions are drawn in section 6.

2. Routing Using Social Insect Metaphor

As indicated above, active networking and adaptive techniques based approaches emphasize a per packet mechanism for routing. The aforementioned "adaptive" techniques [3] tend to emphasize adding "intelligence" to the routers leaving the packets unchanged. A social insect metaphor provides a middle ground in which the concepts of a routing table and data packet still exist, but in addition, new control packets – *ants* – are introduced that interact to keep the contents of the routing tables up to date. To do so, the operation of ant packets is modeled on observations made regarding the manner in which worker ants use chemical trails as a method of indirect *stigmergic* communication.

Specifically, ants are only capable of simple stochastic decisions influenced by the availability of previously laid stigmergic trails. The chemical denoting a stigmergic trail is subject to decay over time, and reinforcement proportional to the number of ants taking the same path. Trail building is naturally a bidirectional process, ants need to reach the food (destination) and make a successful return path, in order to significantly reinforce a stigmergic trail (forward only routing has also been demonstrated) [6]. Moreover, the faster the route, then the earlier the trail is reinforced. An ant on encountering multiple stigmergic trails will *probabilistically* choose the route with greatest stigmergic reinforcement. Naturally, this will correspond to the 'fastest' route to the food (destination). The probabilistic nature of the decision, however, means that ants are still able to investigate routes with a lower stigmergic trial. The following section summarizes the AntNet algorithm of Di Caro and Dorigo [4].

	All Network Nodes (Possible Destinations)			
Outgoing Links (Each node has L neighboring links)	$P_{1,1}$	$P_{1,2}$	------	$P_{1,55}$
	$P_{2,1}$	$P_{2,2}$	------	$P_{2,55}$
	---	---	------	---
	$P_{L,1}$	$P_{L,2}$	------	$P_{L,55}$

Table 1.Original AntNet routing table at node *k* on NTTNET

2.1 AntNet Algorithm

In AntNet, it is assumed that routing tables (Table 1), T_k, exist at each node, k, in which a routing decision is made. Tables consist of L rows, one row for each neighboring node/link. As far as a normal data packet is concerned, if the destination, d, from current node, k, is a neighbor then the routing is still a stochastic decision. In all other cases, a route is selected based on the neighbor node probabilities.

1. New forward ants, F_{sd}, are created periodically, but independently of the other nodes, from source, s, to destination node, d, in proportion to the destination frequency of passing data packets. Forward ants travel the network using the same priority structures as data packets, hence are subject to the same delay profiles.
2. Next link in the forward ant route is selected stochastically, $p(j)$, in proportion to the routing table probabilities and length of the corresponding output queue.

$$p'(j) = \frac{p(j) + \alpha l_j}{1 + \alpha(|N_k| - 1)}$$

where $p(j)$ is the probability of selecting node j as the next hop; α weights the significance given to local queue length verses global routing information, $p(j)$; l_j is proportional to the inverse of queue length at destination j normalized to the unit interval; and N_k is the number of links from node k.

3. On visiting a node different from the destination, a forward ant checks for a buffer with the same identifier as itself. If such a buffer exists, the ant must be entering a cycle and dies. If this is not the case, then the ant saves the previously visited node identifier and time stamp at which the ant was serviced by the current node in a buffer with the forward ant's identifier.
4. When the current node is the destination, $k = d$, then the forward ant is converted into a backward ant, B_{ds}. The information recorded at the forward ant buffer is then used to retrace the route followed by the forward ant.
5. At each node visited by the backward ant, routing table probabilities are updated using the following rule,
   IF (node was in the path of the ant)
   THEN $p(i) = p(i) + r\,\{1 - p(i)\}$
   ELSE $p(i) = p(i) - r\,P(i)$

where $r \in (0, 1]$ is the reinforcement factor central to expressing path quality (length), congestion and underlying network dynamics.

As indicated above, the reinforcement factor should be a factor of trip time and local statistical model of the node neighborhood. To this end [4] recommend the following relationship,

$$r = c_1\left(\frac{W_{best}}{t_{ant}}\right) + c_2\left(\frac{I_{sup} - I_{inf}}{(I_{sup} - I_{inf}) + (t_{ant} - I_{inf})}\right)$$

where W_{best} is the best case trip time to destination d over a suitable temporal horizon, W; t_{ant} is the actual trip time taken by the ant; $I_{inf} = W_{best}$; $I_{sup} = \mu_{kd} + \{\mu_{kd} / [W (1 - \eta)]^{0.5}\}$. The estimates for mean, μ_{kd}, and variant, η_{kd}, of the trip time are also made iteratively, using the trip time information, o_{kd}. Thus,

$$\mu_{kd} = \mu_{kd} + \eta(o_{kd} - \mu_{kd})$$
$$(\eta_{kd})^2 = (\eta_{kd})^2 + \eta \{(o_{kd} - \eta_d)^2 - (\eta_{kd})^2\}$$

trip time information is now updated incrementally based on the recorded trip duration between current node, k, and ultimate destination, d.

3. Proposed Scheme

Experiments of AntNet showed that results are encouraging [4]. The main characteristics of AntNet showed better performance under different experimental conditions with respect to other dynamic routing algorithms [5], e.g. RIP, OSPF. However, there are still some problems with this adaptive algorithm:

1. The definition of routing tables is, such that every node has a unique location in the routing table, see Table 1, or a total of L (number of neighboring nodes) by K (number of nodes in the entire network) entries. As demonstrated in [9], the global information assumption of AntNet is unrealistic. To do so would assume that it is first feasible, and secondly, should the network configuration ever change, then all nodes should be updated with the new configuration information.
2. As demonstrated in [1], a node prefers a link with higher probability to a destination when choosing an outgoing link to send a packet. If a link keeps good condition for a long time, its probability to that destination will be very high. Under such a circumstance, the node will "stick" to this outgoing link and loose its adaptive ability. This is called the problem of "Stickiness".

Thus, this paper proposes a new design of AntNet, where the structure of the routing table in the nodes and the behaviors of ants are different from the ones in [4]. In this new design, each node only has a limited routing table of size L by r. Unlike the original AntNet approach, a node only knows its neighbors, and some "popular" destinations. The total number of "popular" destinations is r, which is a small number, chosen to allow higher performance and minimum possible routing table size, Table 2. After period T, the node will update its routing table and local traffic statistics table by adding popular destinations, and removing destinations, which become unpopular over time (less packets go to those destinations).

When a new node is connected to the network, the node and its neighbors will update their tables. The new neighbor will be added to the routing table, and the local traffic statistics table. Each node will update its tables regularly every T seconds. To do so, a node sorts records in the local traffic statistics table to select r top popular destinations. Then the node checks destinations in the routing table to see whether it is a neighbor or among the top r popular destinations. Destinations that satisfy the above condition will be kept intact; otherwise they will be removed from the routing table and the local traffic statistics table. After checking the routing table, the node will fill the vacancies in the routing table and the local traffic statistics table with nodes appeared in the top popular list until the size of the routing table reaches to r.

In this model, the routing table and the local traffic statistics table are smaller in size than the one in Di Caro's AntNet model. Suppose there are K nodes in the network, each node only keeps r destinations, the routing information kept in each node is therefore $r/(K - 1)$ of that in AntNet. Meanwhile, the fact that popular destinations exist in the routing table will preserve network's "delivering" ability, and trip time of data packets. Thus, this will solve the first problem mentioned above. To solve the second problem, stickiness, we followed the approached employed by [1]. In [1], the following formula is proposed to generate an upper limit for the probability of each outgoing link:

$$P = 1 - (L - 1)*\alpha$$

where L is the number of outgoing links of a node and α is coefficient. It is shown experimentally that the $\alpha = 0.05$, gives the best results in terms of preventing stickiness. Thus, the same value is employed in our experiments. The following discusses the new algorithm step by step.

	All Network Nodes (Possible Destinations)			
Outgoing Links (Each node has L neighboring links)	$P_{1,1}$	$P_{1,2}$	----	$P_{1,r}$
	$P_{2,1}$	$P_{2,2}$	----	$P_{2,r}$
	------	------	---	------
	$P_{L,1}$	$P_{L,2}$	---	$P_{L,r}$

Table 2. Proposed routing table at node k on NTTNET (r<55)

3.1 Proposed Algorithm

1. At regular intervals, each node generates and sends an ant to a destination. In our model, each node selects destinations for forward ants based on "popularity" – the occurrence of the destination in data packets. Popular destinations will have more chance to be selected as destinations of ants.
2. Once a node receives an ant, it will forward the ant if it is not the destination of the ant. In [1, 4], ants that go into dead ends (nodes having only one outgoing link) are killed because the original AntNet model assumes that these ants chose wrong directions. In this work, such ants will be sent back to where they come from. Moreover, each forward ant has a life flag, which is similar to TTL in other algorithms. Thus, before a node forwards an ant, the node will first decrease the life flag of the ant, and then will forward it. If the new flag is zero, this ant will be discarded. However, there are no life flags for backward ants. Backward ants only follow the path that forward ants explored, thus it is not necessary to set a limit for them. If a backward ant cannot be forwarded because of a link or a node failure, it will be killed since its information is not valid any more.
3. When a backward ant goes back to its source, if the destination exists in the routing tables of the nodes that are on its path, the ant will update the routing tables; otherwise the ant will be sent back without updating. When a backward ant reaches the source, it will be killed after it updates the routing table of the source.
4. On the other hand, when a node receives a data packet, which needs to be forwarded, the node will look for its destination in the routing table. If this destination can be found, it will be forwarded based on the algorithm defined in section 2. If the destination does not exist in the routing table, the node will choose randomly one of its neighboring links (except the incoming one) to forward the packet. If a data packet happens to stay in a loop for some time, then its life flag (TTL) will become zero. In return, it will be discarded.

3.2 Deterministic Data Packet Forwarding for AntNet

In the original AntNet algorithm [4], when a data packet is forwarded, the outgoing link is chosen based on the probabilities in a routing table. Indeed, this is very different from what is applied in the current routing schemes. However, there is one possibility where a middle ground can be found between the probabilistic approach and the current routing algorithms. To this end, ants will be forwarded probabilistically as described in section 3, while data packets will be forwarded not probabilistically but to the link with the highest probability. Hereafter, we will refer to this as "deterministic".

4. Simulation Environment

All experiments were implemented with the network simulator JavaSim 1.0 [8]. To make this work comparable with [1, 4], all experiments were conducted on NTTNET, Figure 1. It is a narrow, long network with 55 nodes, and 162 bi-directional links. Link bandwidth is 6Mbps; link propagation delays are between 1 to 5 ms (they are unified to 3 ms in the experiments here). Specifically, NTTNET is modeled, where this represents a narrow long configuration in which the degree of connectivity is low (from 1 to 5), hence the NTTNET provides a more demanding configuration for testing routing algorithms, as higher degrees of connectivity lower the possibility of packet loss due to loops, timeouts.

In all experiments, each node generates ants to random destinations at every 300ms probabilistically (gives approximately 270000 control/ant packets). On the other hand, data packets (512 bytes) are generated randomly according to Poisson distribution, where the mean interval is 21ms (gives approximately 3.5 million data packets). Each experiment lasts 1500s. All nodes stop generating data packets at 1350s. Moreover, time flag is set to 165 for each data packet and 110 for each forward ant. Both are selected empirically. Furthermore, each node updates its routing

table and local traffic statistics table every 10 seconds. Queue lengths are sampled every 0.02 second. The following is the list of parameters, which are given by [1, 4] and are also employed in this work:

α = 0.3, γ = 0.05, η = 0.05, c_1 = 0.7, c_2 = 0.3, ζ = 0.78, δ = 0.05

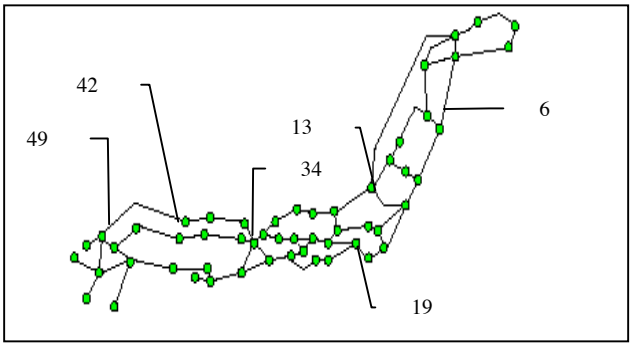

Figure 1. Japanese backbone – NTTNET (55 nodes)

5. Results

The following are experimental results conducted on the proposed new approach and compared against the original AntNet model, which is referred as Global here onwards. The following are the six different routing table sizes studied:

1. Global (size=55): Original AntNet algorithm
2. Size=36: 67% of the Global setting
3. Size=24: 44% of te Global setting
4. Size=12: 22% of the Global setting
5. Size=6: 11% of the Global setting
6. Local: Each node only knows the existence of its neighbors

On measuring the performance of a routing algorithm, we focus on: Network throughput, which is defined as bits/s successfully received at their destination; Queue length (bits), which is defined as the sum of queue lengths over the duration of the simulation; and Percentage of 'lost' data packets.

There are a total of 4 scenarios in each set of experiments; in the first case all routers remain available. The remaining scenarios investigate plasticity of the network by removing different router combinations. In the second scenario, router 34 (a central hub) is down at a time step of 500s and up again at 1000s. In the third scenario, two routers (local hubs) are removed, whereas in scenario four, three routers (2 local hubs and a random router) are removed simultaneously. In both scenarios, nodes are removed at 500s and back again at 1000s.

Tables 3 – 6 show the average percentage of lost data packets (the ones which could not be delivered) under different scenarios. The results show that there is very little difference (if at all) between the deterministic and probabilistic forwarding in terms of lost data packet rate. On the other hand, independent of the forwarding scheme, the loss rate increases as the routing table size decreases specifically when there are link/node outages. However, it should be noted that the loss rate could be kept less than 5% by a routing table size of less than half of the global setting.

Global	36	24	12	6	Local
DETERMINISTIC FORWARDING					
0.3%	1%	2%	4%	6%	6%
PROBABILISTIC FORWARDING					
0.06%	0.2%	1%	4%	6%	7%

Table 3. No node fails – data packet loss rate

Global	36	24	12	6	Local
DETERMINISTIC FORWARDING					
1%	2%	2%	6%	8%	9%
PROBABILISTIC FORWARDING					
1%	1%	3%	7%	9%	10%

Table 4. Node 34 is down & up – data packet loss rate

Global	36	24	12	6	Local
DETERMINISTIC FORWARDING					
1%	2%	3%	6%	9%	9%
PROBABILISTIC FORWARDING					
1%	2%	3%	7%	9%	10%

Table 5. Nodes 13 & 49 are down & up – data packet loss rate

Global	36	24	12	6	Local
DETERMINISTIC FORWARDING					
2%	2%	3%	6%	8%	9%
PROBABILISTIC FORWARDING					
2%	2%	4%	7%	9%	9%

Table 6. Nodes 6, 19 & 42 are down & up – Data Packet Loss Rate

In addition, figures 2 - 3 show the trend of queue lengths, and figures 4 - 5 show the trend of throughput under different scenarios. In this case, the effects of changing the routing table sizes are similar to data packet loss rate, but when the size is less than 24

(44% of the global setting), the slopes are almost linear. However, it should be noted that the trend of queue length under deterministic forwarding, figure 5, does not follow this behavior, we speculate that deterministic forwarding can not adapt to changes happening on the network, and therefore, queue lengths start to increase once a node outage/link failure takes place independent of the size of the routing table. Furthermore, the overheads introduced by ants are trivial: the number of ants is approximately 8% of that of data packets. With such a low overhead, although the routing table sizes are highly limited, all main characteristics (packet loss rates, queue lengths and throughputs) are kept relatively low. When the routing table size is 24, the worst data packet loss rate is 4% whereas queue length and throughput are still manageable compared to the global setting. Based on these results a routing table of size from 44% to 22% of that of global setting seems very reasonable for this network.

6. Conclusion

In this work, we emphasize the case in which routing table features, as well as content, are evolved. Thus, we are not privy to *a priori* knowledge regarding the number of nodes in the network. We proposed a new approach based on AntNet [4], which is an adaptive routing algorithm. However, unlike AntNet our approach can deliver more than 95% of the packets without the global information even under severe node outages/ link failures. In this new approach, many properties of the AntNet are kept intact. The most significant difference is the routing tables and the local traffic statistics tables. They only have the most popular destinations and update the destinations at the scheduled time based on the data packets visiting. Under this approach, a data packet will be forwarded randomly if its destination does not exist in the routing table. Data packets will not be killed until their life flag expire, even if they go into loops. Unlike the original AntNet algorithm, in our approach, ants also have flags to avoid loops. To evaluate this new approach, we have conducted several experiments on NTTNET. The results show that our approach with a limited routing table size gives promising performance. When the sizes are largely limited, there are no significant increases in queue lengths and throughputs. Meanwhile, the network with the new algorithm will work more robustly: once there are severe node failures, the network can find the alternative routes quickly.

Moreover, the experiments also show that the approach that forwards data packets deterministically and forwards ants probabilistically is not feasible. Such an approach is not adaptive because in many experiments, the algorithm fails to adapt to topology changes. From our observations, the proposed approach gives promising results with probabilistic forwarding (for both ants and data packets).

However, there are still different avenues to investigate. More experiments are required to evaluate the performance under various situations, such as traffic pattern and network topology. Furthermore, it would be interesting to enable each node to adjust its own routing table size dynamically. In a network, each router has different significance in the topology. Some are like a "hub" for the network (e.g. node 34 in NTTNET), while some are only trivial routers (e.g. located at the leaves). Then the question is: Can each node know its role in the network recurring to the information collected by ants, and then adjusts its settings, such as the size of the routing table, or intervals to generate ants? Security is another concern that needs to be explored.

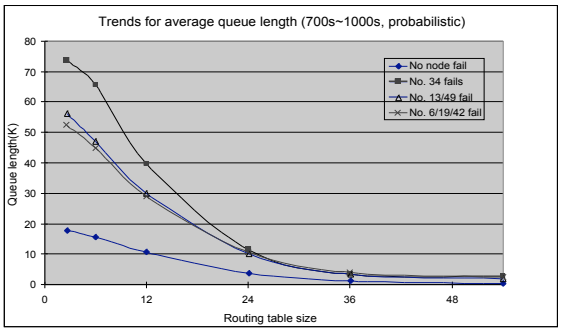

Figure 2. Trends for queue length (probabilistic)

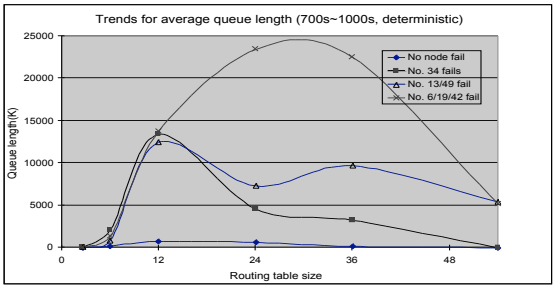

Figure 3. Trends for queue length (deterministic)

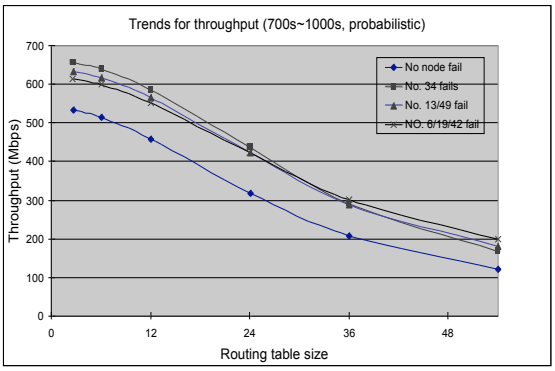

Figure 4. Trends for throughput (probabilistic)

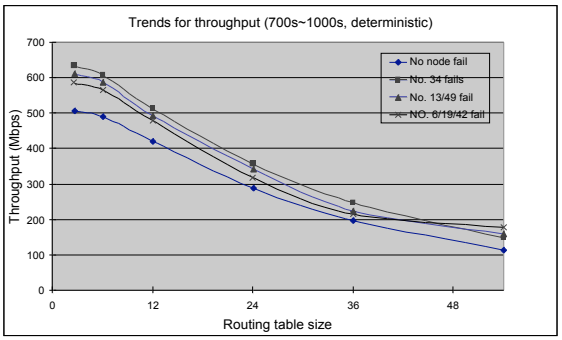

Figure 5. Trends for throughput (deterministic)

Acknowledgements

This work was supported in part by Discovery grant, of the second author from the Natural Sciences and Engineering Research Council of Canada.

References

[1] Barán B, Sosa R., AntNet Routing Algorithm for Data Networks based on Mobile Agents, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No. 12, pp 75-84. 2001.

[2] Boyan J.A., Littman M.L., Packet routing in dynamically changing networks: A reinforcement learning approach, Advances in Neural Information Processing Systems, Vol.6, pp 671-678, 1994.

[3] Corne D.W., Oates M.J., Smith G.D., Telecommunications Optimization: Heuristic and Adaptive Techniques. John Wiley & Sons, isbn 0-471-98855-3, 2000.

[4] Di Caro G., Dorigo M., AntNet: Distributed stigmergetic control for communications networks, Journal of Artificial Intelligence Research, 9, pp 317-365, 1998.

[5] Forouzan B. A., Data communications and networking, Mc-Graw Hill, ISBN 0-07-232204-7, 2001.

[6] Heusse M., Snyers D., Guerin S., Kuntz P., Adaptive agent-driven routing and load balancing in communication networks, Advances in Complex Systems, 1, pp 237-254, 1998.

[7] Huston G., Analyzing the Internet BGP routing table, The Internet Journal, 4(1), pp 2-15, March, 2001.

[8] JavaSim: www.javasim.org, retrieved on March 9, 2003

[9] Liang S., Zincir-Heywood A.N., Heywood M.I., The effect of routing under local information using a social insect metaphor, IEEE International Congress on Evolutionary Computation, pp 1438-1443, May 2002.

[10] Tennenhouse D., Smith J., Sincoskie W., Wetherall D., Minden G., A survey of active network research, IEEE Communications Magazine, 35(1), pp 80-86, January 1997.