Realization of a Distributed Route Management System on Software Routers

Anders Mykkeltveit[1,2] Poul E. Heegaard[1,*] Otto Wittner[2]

[1]Deptartment of Telematics,
Norwegian University of Science and Technology,
Trondheim, Norway

mykkeltv@item.ntnu.no
poulhe@item.ntnu.no

[2]Centre for Quantifiable Quality of Service in Communication Systems.
Norwegian University of Science and Technology,
Trondheim, Norway[†]

wittner@q2s.ntnu.no

Abstract

This paper describes the implementation of a distributed route management system called the CE-ant algorithm. The algorithm is developed at the Department of Telematics at the Norwegian University of Science and Technology (NTNU), and results from simulations are promising. The algorithm has, however, never been implemented and tested in a real network carrying real data traffic. The pioneering implementation presented in this paper is based on two main components; an ant-system and a software router. The ant-system is implemented in Java supported by the Kaariboga mobile agent system, while Click and C++ is used to implement the software router. The algorithm is successfully implemented, and the overall system is tested in a small test network with good results.

1 Introduction

Today, the number of users and services in telecommunications and computer networks is increasing. In the future, it is expected that full-service networks will be developed that handle all types of services, satisfying a variety of Quality of Service (QoS) requirements. These networks will be heterogenous and complex, and sophisticated tools are needed for efficient and dependable management.

A distributed algorithm, capable of solving some of the management tasks in complex networks, has been developed at the Department of Telematics at NTNU [12]. The algorithm is inspired by the emergent behaviour observed in ant colonies. By emergent behaviour, ants find near optimal solutions to complex problems like maintaining an optimal path between their nest and a food source. This problem closely resembles the

[*]This work was also partially supported by the Future & Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923).

[†]"Centre for Quantifiable Quality of Service in Communication Systems, Centre of Excellence" appointed by The Research Council of Norway, funded by the Research Council, NTNU and UNINETT. http://www.q2s.ntnu.no

problem of finding the optimal route between a source and destination node in a computer or telecommunications network.

One important property of ant based systems is their distributed nature. While most network management systems are centralized, there is no central control point in an ant colony. If an ant dies, the colony as a whole is not affected since other ants take over the task of the one that is lost. The use of ant based control over networks eliminates the single point of failure problem of many of today's network management systems.

An intuitive way to implement the ants is to let them be mobile agents. A mobile agent is a program that is able to move itself around in a network and execute its code on different computers. The agent is given a specific goal by its user, and the agent takes autonomous decisions of how to achieve its goal. To be able to operate on a computer and migrate to other locations, the agent needs support from a Mobile Agent System (MAS). The MAS provides a runtime environment and possibly a set of services to the agents [5].

This paper describes an implementation of a system that uses the CE-ant algorithm to manage routing in an IP-network. The implementation was the subject of a Master's Thesis at the Department of Telematics at NTNU [8].

Figure 1 shows the system in operation. Ants belonging to a colony move around in the network, maintaining a connection between the two host computers. Ants that are launced from the source host look for a path to the destination host. When they have reached the destination, they go back to the source host from which they were sent. On the way back, the ants mark the trail they have found in each router of the path. The routers use this information to decide where to forward incoming packets to the destination host.

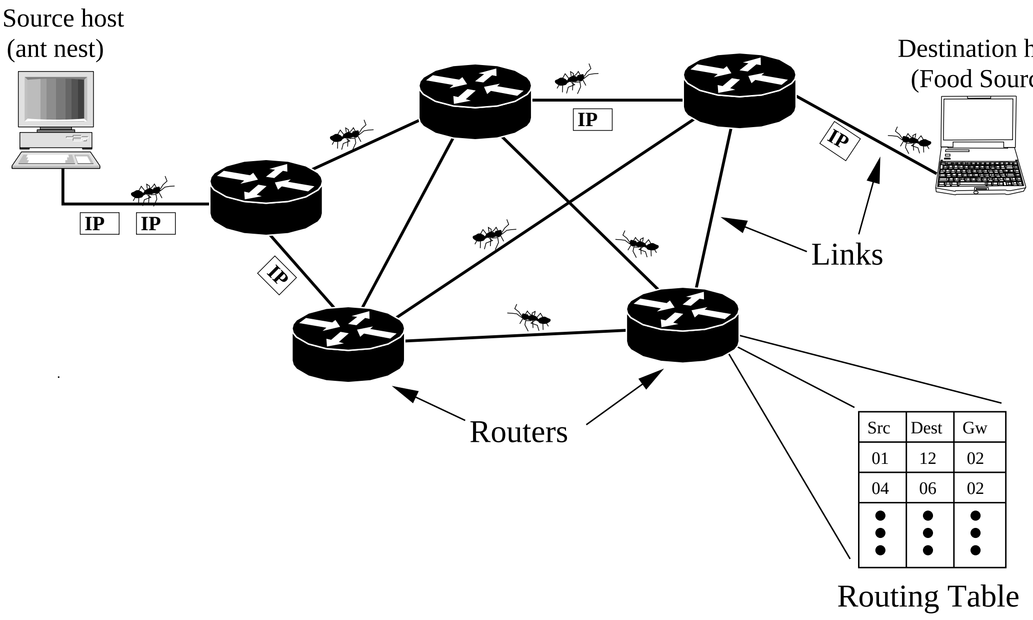

Figure 1: Network overview

The ants are implemented as mobile agents using Java and the Kaariboga [11] MAS. All the computers that are part of the network run an application that provides a runtime environment for the agents in addition to interaction with the routing software. The routers, which are customized implementations of Click software routers, provide dynamic information on traffic load to the ants. The ants use this information to evaluate the quality of the paths found.

The remainder of this paper is organized as follows. In section 2, the concepts of the CE-ant algorithm is presented. Section 3 describes the implementation of the system. Section 4 describes a scenario for testing the implementation. The results from this test scenario are presented in section 5. Finally section 6 provides some concluding remarks and suggestions for future work.

2 The CE-ant Algorithm

In an ant system, the ants are able to find the best path between their nest and a food source. The ants communicate indirectly by dropping a type of hormones called pheromones. Other ants arriving at a place where there are pheromones are influenced by the strength of the pheromones when they decide where to go next. When an ant has found a short path between its nest and a food source, it leaves a trail of pheromone along the path. Other ants follow the path, leading to a trail that gets stronger and stronger since the probability that an ant will follow a given path is proportional to the strength of the trail. Over time, the pheromones evaporate and disappear if they are not refreshed by new ants passing by. Paths that do not lead to a rich food source will therefore disappear [10].

Besides finding the shortest path between their nest and a food source, ant colonies are able to solve many other tasks such as keeping a constant temperature in their nest, cooperate in carrying large objects and even migrating the entire colony to another nest.

A number of systems using ant technology for network management exists, see for example [1, 10].

One of the differences between the CE-ant algorithm and other ant-inspired algorithms is that it is based on the cross-entropy driven optimization method proposed by Rubinstein [9].

Using the cross-entropy method gives the CE-ant algorithm a formal foundation, which makes it easier to analyse and understand the results produced by the algorithm. In short, the cross-entropy method presents an iterative process which updates a matrix of transition probabilities in steps such that a near optimal sequence of transitions appears as high probabilities in the matrix.

CE-Ant Behaviour

The CE-ant algorithm reformulates the cross-entropy method making it resemble a colony of independent ants, and hence suitable for implementation in a distributed environment.

In this section only the principal mechanisms of the CE-ant algorithm are given. The full description of the algorithm including the schemes for updating the pheromones and other algorithm-related parameters can be found in [4].

The algorithm uses a control parameter called the temperature to determine how much emphasis should be placed on the best paths found. The temperature is similar to simulated annealing where low temperatures represent a stable system and increase the emphasis on the best path found.

The process of finding a path between two hosts is described in the following.

- A host that wishes to establish a connection to another host launches a set of ants that move themselves from computer to computer in the network. In the exploration phase, the ants choose which computer to go to next at random among the neighbours that are reachable. In this implementation of the algorithm, the ants try to avoid revisiting nodes they have already visited, but this heuristic is not required for the algorithm to work.

- Along the path, the ants accumulate the “cost” of each link they travel through. This cost can be dependent on the demands of the user of the connection, and any additive cost function may be used. In the implementation of this system, the cost function was based on the delay of the path. This is explained in the section “Calculating Link Costs”.
- When an ant has reached the destination host, it returns on the reverse path. At each router on the way back, the ant updates the pheromones according to the temperature and the quality of the path found.
- After some time, the system enters the maintenance phase. Knowledge about some paths are by now established along with pheromones in most of the network nodes. The ants change strategy and do stochastic routing of themselves based on normalized pheromone values of each interface of each node. This strategy resembles the behaviour of ants in nature and results in dominating pheromone levels associated with one of the paths. The probability of choosing a specific path is the product of the pheromone levels for the links in the path normalized with respect to the total pheromone level in each node.
- When the temperature and probabilities for the different paths are relatively stable, convergence is said to be reached. The probabilities for paths with high costs are now small, while the highest probabilities are found for the best paths. If one path is superior to the others, the probability for this path may approach 1.

Since the network topology may change, the algorithm should be able to adapt to link failures and the appearance of new links in the network. When a link fails, the ants simply do not consider the pheromone for that link, and they choose their path based on the remaining pheromones in a router.

If a new link should appear in the network, the pheromone level for that link may be set to a low value compared to the pheromones of the other links in that node. This would lead to a small probability for an arriving ant to choose the link. In order to speed up the discovery of all the links, some of the ants could continue to chose links at random after the initialization phase is over. These ants are called explorer ants and their task is to discover and adjust the pheromone level for new links so they may be used by the other ants if the link can be part of a good path.

Calculating Link Costs

In general, if packet arrivals to a link buffer are according to a Poisson process, the delay W_l on a link l, can be estimated by applying Pollaczek-Khintchine’s theorem[1] .

$$W_l = \overline{M_l} + \frac{\lambda_l \cdot \overline{M_l^2}}{2(1-\lambda_l \cdot \overline{M_l})}$$

where λ_l is the arrival rate in packets per second. The first term in the equation, $\overline{M_l}$ is the mean packet processing time experienced by all packets. The second term represents the time a packet must wait in the router’s transmit buffer before it can be sent out on the link. $\overline{M_l^2}$ in this term, is the mean square packet processing time. As the arrival rate

[1]see for example [2]

grows toward the packet processing time, the waiting time grows toward infinity due to the increasing number of packets in the buffer.

During forward search, the ants accumulate the delays for each of the links they travel through. The cost, $L(\pi)$, of the path π is the sum of the delay on each of the links in the path.

$$L(\pi) = \sum_{l \in \pi} W_l$$

In order to avoid reporting paths with unacceptably high delay, the ants commit suicide if the accumulated delay of the path becomes too high.

3 Implementation

The goal of the implementation was to create a system that operated in a real network and routed real traffic. The system was only planned as a prototype to gain experience with technical issues and unpredictable "real life" effects. Therefore, performance was not considered important.

The ant-system is implemented in Java and runs on all computers in the network. The ants are implemented as mobile agents using the Kaariboga MAS. Ants are launched from host computers and search for paths to the destination host that the user wants to communicate with. In addition to committing suicide when the calculated delay of a link becomes too large, ants may be lost for other reasons. Ants can be destroyed due to failures in links or nodes in the network, or due to failures or overload in the ant-system. To compensate for the loss of ants, an ant is launched for each connection at regular intervals. Due to the performance limitations of the implemented system, the rate with which ants are launched is relatively low, typically one ant per second. When an ant has reported a path, it is destroyed.

In this system, the capacities of the links and data rates for connections are measured in packets per second. In order for the system to work correctly, the ants must consider whether traffic from their own connections is sent over a link, l, when they compute the arrival intensity λ_l. If data for the connection the ant is working on is sent over a link, the ant can use the packet intensity on that link as the arrival rate. If a link is not used for sending data, the ant will need to add the traffic from its own connection to the packet rate of the link to compute the correct cost if the link was used for sending the data.

In a large backbone network with many connections, the above consideration may be relaxed. Each connection typically contributes with only a small part of the total traffic; hence it will not make a significant difference whether data from the ant's own connection is included in the arrival intensity to the link or not.

In a small network, however, like our test network, one connection might consume nearly the whole capacity of the links. The link may be congested if one additional connection attempts to send its data over the link. Therefore the above consideration is important.

The implementation of the system is different for hosts and routers. The routers forward packets and serve the needs of arriving ants, but do not launch their own ants. The implementations of the routers and hosts are presented in the following subsections.

The Routers

The two main components of the routers are the ant-system and the software routers. These two components interact to integrate the ants with the underlying network. The components of the routers and the interactions between them are shown in figure 2.

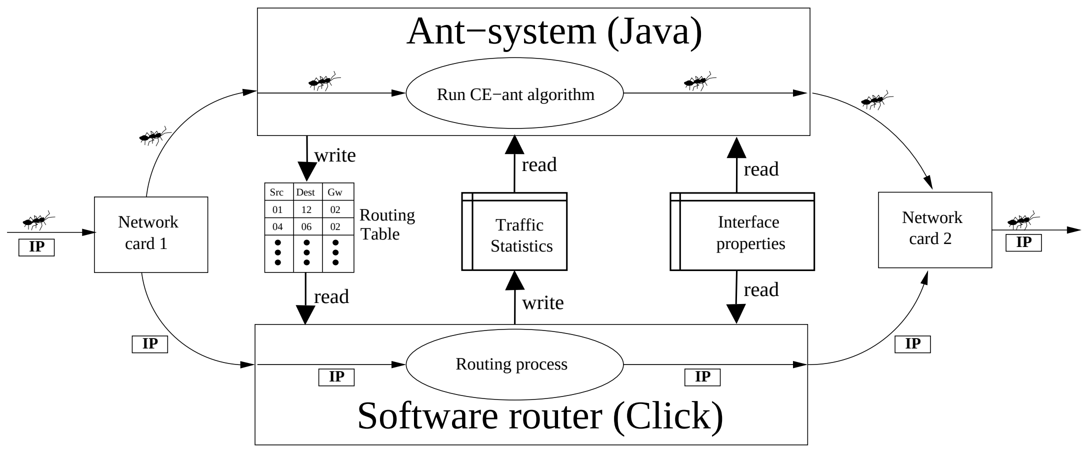

Figure 2: The main components of the routers.

As can be seen from the figure, ants and data packets are treated differently in the router. The ants move from router to router and are addressed directly to the router in question and are ignored by the software router. The ants' program codes are executed in the ant-system. Data packets are addressed to a host computer and are routed by the software router based on the routing table which is maintained by the ants.

The software router is a customised version of the Click Modular Router. Click [6] is an open source router platform that consists of elements implemented in C++. The elements can be grouped together to form a configuration that suits the needs of the users. The Click distribution includes a standard IP-router configuration that handles all the tasks a router is expected to perform, like responding to ICMP messages and decrementing the packets' *Time To Live* field.

For interaction with the ant-system, some new elements were implemented. The software router reads the contents of the routing table and updates the traffic statistics periodically. For simplicity, this information was stored in flat files.

The user must specify the capacities of the links in the system by writing this information to a file denoted "interface properties". The content of the file is read by both the software router and the ant-system. The ant-system provides the information to the ants, while the software router limits the rate at which data is sent to the links. In this way, the ants are able to read information on the actual throughput of the router links. The ants do not pass through the router, and hence the limitation on throughput does not apply to the ants, and the traffic statistics do not include them.

The Hosts

The hosts run the ant-system but have no software router since they do not forward packets to other computers. Since the ants need to distinguish between their own connection's traffic and traffic from other connections, they need to know the amount of traffic generated for their connection. To provide this information to the ants, a packet rate monitor was implemented and inserted into a Click configuration.

Which computers the host wants to communicate with is specified by the user in a file. This information is read into both the ant-system and the packet rate monitor, and the monitor reports the rate of packets sent to each of the destinations. The components of the host part of the implementation are shown in figure 3.

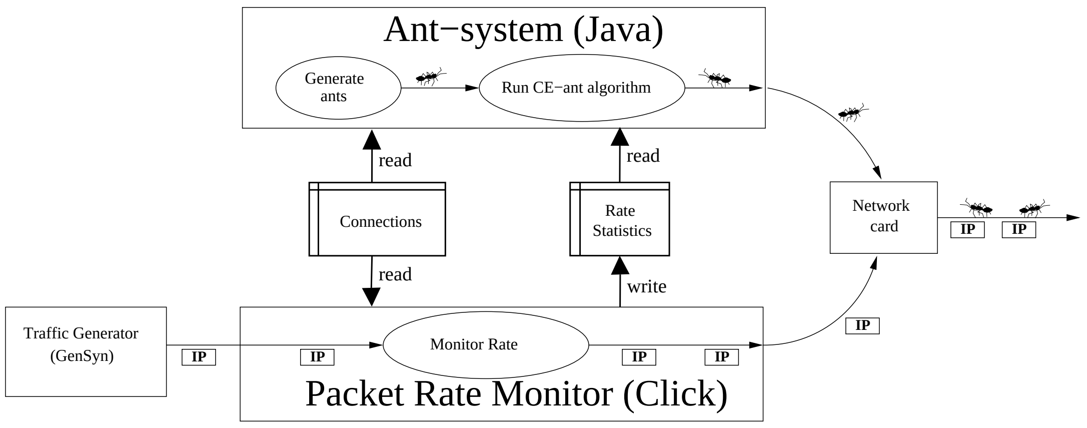

Figure 3: The main components of the hosts.

In the test described in the next section, GenSyn [3] was used to generate traffic into the test network. The traffic could have been generated by any application, but a traffic generator was used to achieve control with the traffic patterns. All packets destined for addresses on other networks were grabbed by Click and sent through the Packet Rate Monitor before they were sent out on the network card.

4 Test Scenario

A network for testing and experimenting with the implemented sytem was set up. The lab consisted of 7 computers equipped with extra network interface cards. A configuration of the network is shown in figure 4. The packet send rate for the links between the software routers should be limited in order to accomodate the ants' cost function. For simplicity the capacities on the links in the core network were set to 50 packets per second.

Each computer in the test network was connected to the Internet via one of their network interfaces, while the other network interfaces were part of the test network. This allows remote login so that all the computers could be administrated from one remotely located computer.

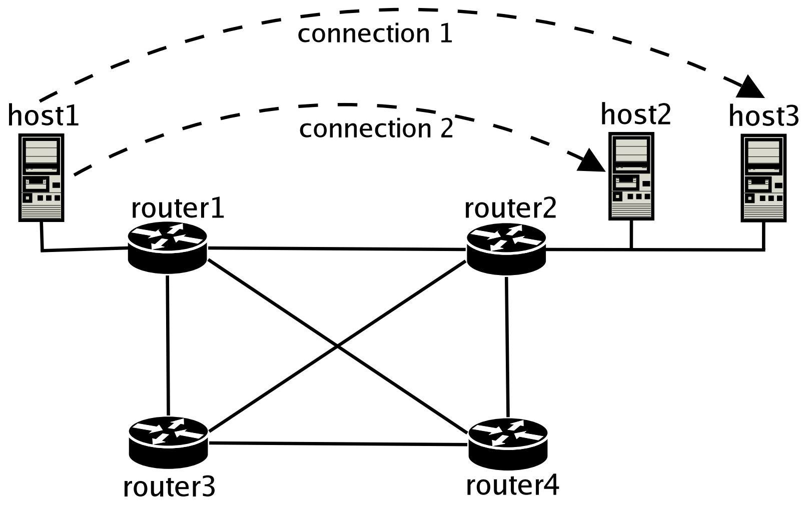

Figure 4: Test network with connections.

A test for studying how the ants coped with the traffic was set up for the network shown in figure 4. In the test, two connections were established. Connection 1 was set up between host1 and host2, while connection 2 was set up between host1 and host3.

The network configuration has four possible paths between the endpoints of the two connections when the ants avoid nodes they have already visited. Without any traffic in the network, the path giving the smallest delay for both of these connections goes through the link from router1 to router2. Two other paths give the same delay when there is no traffic. These were the paths that went via router3 and router4, respectively. The worst possible choice of path was the path that went through all of the routers in the sequence router1 - router3 - router4 - router2.

When the test had been running for some time, and the ants had converged towards the best path, traffic was generated from host1 to host2. The traffic was generated with GenSyn and the packet rate as seen by the ants is shown in figure 5. As can be seen, the traffic is not constant but fluctuates around 40 packets per second. The fluctuations are a challenge for the ants since they may experience acceptable delay values when the observed traffic is at its minimum. The unit of the x-axis of the plots in figures 5 and 6 is time measured in number of ant intervals since the start of the test.

This traffic results in a higher delay for the link from router1 to router2 than for the longer paths that go via router03 and router04. It is therefore expected that the ants working on connection 2 will prefer one of the longer paths when traffic is generated by connection 1.

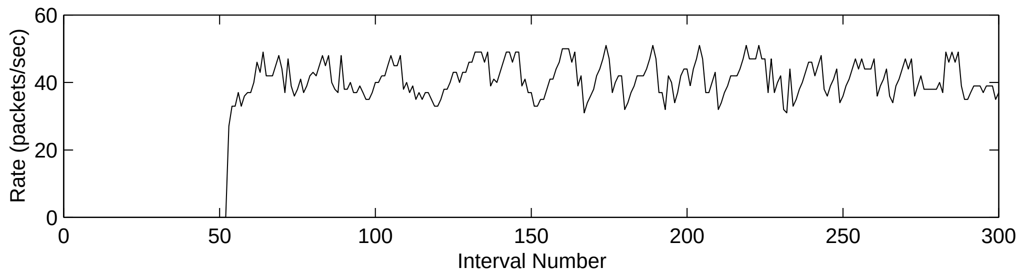

Figure 5: Traffic generated for connection 1 (Multiplexed CBR).

5 Test Results

Figure 6 shows how the traffic on connection 1 influenced the results for connection 2. The upper plot on the figure shows the cost of the path found by an ant measured in milliseconds, while the lower plot shows the probability of finding the specific path. To be able to see the exact path the ants followed through the network, it was necessary to inspect the log file produced by the system.

In the early phase of the test, before the traffic was sent through the network, the pheromones for the path between router1 and router2 had emerged as the preferred path among the ants.

When traffic was added to the network, the costs of the paths found by the ants increased, but the pheromones for the path were nevertheless strengthened until the probability of choosing this path was close to one.

In the time interval between 130 and 150 ant intervals, the cost of the path with highest probability was approximately 83 milliseconds. In this interval, the path router1 - router4 -

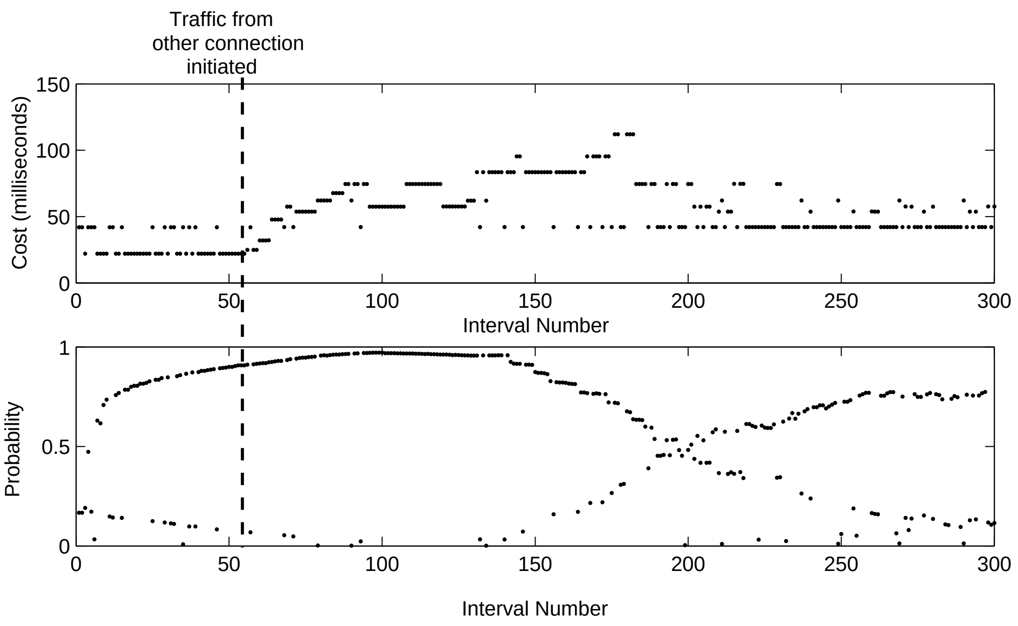

Figure 6: Path cost and probability for connection 2 when traffic was generated on connection 1.

router2 was discovered twice with a very low probability. Since the cost of this path at this time was considerably less than the cost of the converged path, the pheromones for this path were strengthened. The last of these updates resulted in an increase in probability for the path from 3% to 7%. This increase is essential since it resulted in another ant finding the path, thus increasing the probability further. When the test had been running for approximately 200 ant intervals, the probability of choosing the longer path was greater than the probability of choosing the path with traffic, and the path via router4 emerged as the path used by most of the ants. If traffic was now to be sent over the connection from host1 to host3, it would be routed over the path via router4. This path would give the smallest delay for packets as long as the packet rate is smaller than the link capacities.

These results demonstrate the system's ability to adapt to the traffic load on the links in the network. If both connections had generated traffic with the same characteristics as in figure 5, the load had been balanced over two separate paths in the network. If the routing algorithm had only accounted for the number of router hops, as in RIP [7], the traffic for both connections would have been routed over the link from router1 to router2. This link would not be able to handle all the packets. Depending on the transport protocols used, the connections would either have to reduce the packet rates (TCP), or data would be lost or excessively delayed (UDP).

6 Concluding Remarks and Future Work

This paper describes the implementation of the CE-ant algorithm. The ants in the algorithm were implemented as mobile agents and software routers were used to route traffic based on the information gathered by the ants. The system is the first implementation of the algorithm that operates in a real network, routing real traffic.

The performance of the system is limited by the performance of the software router, the ant-system and the communication between these systems. Using Java and mobile agent technology is inefficient. Java is in general slow since the byte-code must be interpreted, and the approach with mobile agents is not efficient either, since a new object has to be created on the server each time an ant arrives. Click is implemented in C++ and is reported by its creators to give much better performance if special drivers that poll the network cards for packets are used rather than the inefficient drivers used in this version of the system. Finally, the communication between the two systems is a bottleneck for the system's performance.

For operation in a larger network, a new version of the system with better performance should be implemented. This future version of the system could integrate the algorithm functionallity into a software router like Click. With proper setup of the hardware devices and by implementing the ants as packets instead of objects, significantly better performance could be achieved.

References

[1] Gianni Di Caro and Marco Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[2] Jeremiah F. Hayes and Thimma V. J. Ganesh Babu. *Modeling and Analysis of Telecommunications Networks*. John Wiley & Sons, 2004.

[3] Poul E. Heegaard and Brynjar Å. Viken. A distirbuted test environment for IP performance evaluation. *Telektronikk*, 97(2/3):245–268, 2001.

[4] Bjarne E. Helvik and Otto Wittner. Using the cross entropy method to guide/govern mobile agent's path finding in networks. In *Proceedings of 3rd International Workshop on Mobile Agents for Telecommunication Applications*. Springer Verlag, August 14-16 2001.

[5] Damir Horvat, Dragana Cvetkovic, Veljko Milutinovic, Petar Kocovic, and Vlada Kovacevic. Mobile agents and java mobile agent toolkits. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[6] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[7] G. Malkin. RIP version 2. RFC 2453, IETF, November 1998.

[8] Anders Mykkeltveit. *Realization of a Distributed Route Management System on Software Routers*. Master's thesis, Department of Telematics, Norwegian University of Science and Technology, June 2004.

[9] Reuven Y. Rubinstein. The cross-entropy method for combinatorial and continous optimization. *Methodology and Computing in Applied Probability*, pages 127–190, 1999.

[10] Ruud Schoonderwoerd, Owen Holland, Janet Bruten, and Leon Rothkrantz. Ant-based load balancing in telecommunications networks. *Adaptive Behaviour*, 5(2):169–207, 1997.

[11] Dirk Struve. *Kaariboga Mobile Agents.* http://www.projectory.de/kaariboga/index.html, 2003. Visited September 5, 2003.

[12] Otto Wittner. *Emergent Behaviour Based Implements for Distributed Network Management*. PhD thesis, Norwegian University of Science and Technology, 2003.