AntNet
Routing Algorithm for Data Networks based on Mobile Agents

Benjamín Barán* **and Rubén Sosa**+

* National Computer Center, National University of Asuncion
P.O.Box 1439, San Lorenzo-Paraguay

+ Science and Technology School, Catholic University of Asuncion
Tuyutí 1343 - Tel. 595-21-390378, Asunción-Paraguay

bbaran@sce.cnc.una.py, rsosa@personal.com.py

Abstract

AntNet is an innovative algorithm for packet routing in communication networks, originally proposed by M. Dorigo and G. Di Caro, in 1997. In AntNet, a group of mobile agents (or artificial ants) build paths between pair of nodes, exploring the network concurrently and exchanging obtained information to update the routing tables. This work analyzes AntNet and proposes improvements that were implemented, comparing their performance with respect to the original AntNet and other commercial routing algorithms like RIP and OSPF.

Keywords: mobile agents, routing, performance, communication network.

1. Introduction

Routing is fundamental in communication network control. In data networks it means the action of addressing data traffic between pairs of source-destination nodes. In conjunction with a flow control, congestion and admission, routing determines the total network performance, in terms of quality and amount of offered services [Dorigo et al 98].

The routing task is performed by routers, which update their routing tables by means of an algorithm specially designed for this purpose. The first routing algorithms addressed data in a network minimizing any costs function, like physical distance, link delay, etc. However, throughput optimization remained in a second plane, possibly due to a relatively small amount of users. This is the case of the RIP algorithm (Routing Information Protocol), based on the distance-vector method and the OSPF (Open Shortest Path First), algorithm thoroughly used in Internet, based on the link-state method. Both methods choose the path with minimum cost (generally the shortest path) between pair of nodes [Dorigo et al 97]. This can lead to "bottlenecks", because all traffic congests the best path, while other paths are left without use [Shankar et al 92].

As a result, traditional routing methods do not have enough flexibility to satisfy new routing demands. New network services and the impressive increase in the amount of users force network administrators to improve throughput in order to satisfy the immense amount of simultaneously requested services. This situation has impelled the study and development of other routing methods, to satisfy these new demands. Such is the case, for example, of a routing method known as LBR (Load Balancing

Routing) [Back et al 99], based on a load-balancing scheme. This method addresses routing by equally distributing load over all possible paths. This diminishes the congestion probability in more inexpensive links, improving the overall network performance.

Nowadays, most studied routing alternatives are based on mobile agents [Dorigo et al 97][Dorigo et al 98][Schoonderwoerd et al 97]. In fact, the present work analyzes an algorithm based on mobile agents, known as AntNet, which was first proposed by M. Dorigo and G. Di Caro, of the Free University of Brussels-Belgium (Dorigo 1997, 1998). AntNet was inspired in previous successful works, based on ant colonies (ACS: Ant Colony Systems) [Dorigo et al 96][Dorigo et al 97][Barán et al 99].

ACS is an optimization method where a group of artificial ants moves around a graph, which represents the instances of the problem. So, they move building solutions and modifying the problem using the obtained information, until they find good solutions to the problem.

The ACS concept is used in AntNet. Here, each artificial ant builds a path from its source node to its destination. While an ant builds a path, it gets quantitative information about the path cost and qualitative information about the amount of traffic in the network. Then, another ant travelling through the same path will carry this information, but in the opposite direction modifying the visited nodes routing tables. The first simulations with AntNet [Dorigo et al 97][Dorigo et al 98] showed a promising performance, overcoming traditional algorithms like RIP and OSPF, demonstrating that it is a valid alternative for data routing.

The present work analyzes Dorigo and Di Caro versions of AntNet [Dorigo et al 97][Dorigo et al 98]. After that, it proposes two improved versions, which were implemented in C language together with the two original AntNet versions, besides versions of RIP, OSPF and three versions of LBR, adding a dozen of algorithms to be compared. Finally, the simulation results show the improvements in throughput and packet delay obtained with the proposed modified versions.

2. AntNet Algorithms

The AntNet first version, presented in 1997 [Dorigo et al 97], will be denominated AntNet1.0, and the second version published in 1998 [Dorigo et al 1998] will be called AntNet2.0. Both versions are briefly discussed.

2.1. Algorithm AntNet1.0

Suppose a data network, with **N** nodes, where *s* denotes a generic source node, when it generates an agent or ant toward a destination *d*. Two types of ants are defined:

a) Forward Ant, denoted $F_{s\rightarrow d}$, which will travel from the source node *s* to a destination *d*.

b) Backward Ant, denoted $B_{s\rightarrow d}$, will be generated by a forward ant $F_{s\rightarrow d}$ in the destination *d*. It will come back to *s* following the same path traversed by $F_{s\rightarrow d}$, with the purpose of using the information already picked up by $F_{s\rightarrow d}$ in order to update routing tables of the visited nodes.

Every ant transports a stack $S_{s\rightarrow d}(k)$ of data, where the *k* index refers to the *k*-eth visited node, in a journey, where $S_{s\rightarrow d}(0)= s$ and $S_{s\rightarrow d}(m)= d$, being *m* the amount of jumps performed by $F_{s\rightarrow d}$ for arriving to *d*.

Let *k* be any network node; its routing table will have N entries, one for each possible destination.

Let *j* be one entry of *k* routing table (a possible destination).

Let N_k be set of neighboring nodes of node *k*.

Let P_{ji} be the probability with which an ant or data packet in *k*, jumps to a node *i*, $i \in N_k$, when the destination is *j* ($j\neq k$). Then, for each of the N entries in the node *k* routing table, it will be n_k values of P_{ji} subject to the condition:

$$\sum_{i\in N_k} P_{ji} = 1 \ ; \ j=1,\ldots,\mathbf{N} \,. \qquad (1)$$

The following lines show AntNet1.0 pseudocode, using the symbols and nomenclature already presented:

```
BEGIN
{
    Routing Tables Set-Up: For each node $k$ the routing
    tables are initialized with a uniform distribution of
    probability:
    $P_{ji} = \frac{1}{n_k}, \quad \forall i \in N_k$                          (2)
    DO always (in parallel)
    {
        STEP 1: In regular time intervals, each node $s$
        launches an $F_{s\to d}$ ant to a randomly chosen
        destination $d$.
        /*when $F_{s\to d}$ reaches $k$, ($k\neq d$), it performs step 2*/
        DO (in parallel, for each $F_{s\to d}$)
        {
            STEP 2: $F_{s\to d}$ pushes in its stack $S_{s\to d}(k)$ node $k$
            identifier and the time between its launching
            from $s$ to its arriving to $k$.
            $F_{s\to d}$ selects the next node in two ways:
            (a) It draws between $i$ nodes, $i \in N_k$, where
                each node $i$ has a $P_{di}$ probability (in the $k$
                routing table) to be selected.
                IF the node selected in (a) was already
                visited
                    (b) It draws again the jumping node,
                        but now with the same
                        probability for all neighbors $i$, $i \in N_k$.
                    IF the selected node was already
                    visited
                        STEP 3: A cycle is found. $F_{s\to d}$
                        pops from its stack all data of the
                        cycle nodes, because the optimal
                        path must not have any cycle. $F_{s\to d}$
                        returns to 2 (a) if the time spent in
                        the cycle is less than a threshold;
                        else it dies in order to avoid
                        infinite loops.
                    END IF
                END IF
        }WHILE jumping node$\neq d$
        STEP 4: $F_{s\to d}$ generates in $d$ another ant, called
        backward ant $B_{s\to d}$. $F_{s\to d}$ transfers to $B_{s\to d}$ its stack.
        /*$B_{s\to d}$, will return to s, following the path used b
        $F_{s\to d}$*/
        DO (in parallel, for each $B_{s\to d}$ ant)
        {
            /*When $B_{s\to d}$ arrives from a node $f$, $f \in N_k$ to a
            node $k$, it performs step 5*/
            STEP 5: $B_{s\to d}$ updates the $k$ routing table and
            list of trips, for the entries regarding to nodes
            $k'$ between $k$ and $d$ inclusive, according to the
            data carried in $S_{s\to d}$ ($k'$), increasing
            probabilities associated to path used and
            decreases other paths probabilities, by mean of
            a criteria explained in [7].
            IF $k\neq s$
                $B_{s\to d}$ will leave $k$ and jump to a node given
                by $S_{s\to d}$ ($k$-1).
            END IF
        }WHILE ($k\neq s$)
    }
}END
```

The routing table and list of trips updating methods for k are described as follows:

Routing table of node k is updated for the entries corresponding to all nodes k' between k and d inclusive. For example, the updating approach for the d node, when $B_{s\to d}$ arrives to k, coming from f, $f \in N_k$ is briefly explained, as following:

a) A P_{df} probability associated with the node f when it wants to update the data corresponding to the d node is increased, according to:

$$P_{df} \leftarrow P_{df} + (1 - r').(1 - P_{df}). \tag{3}$$

where r' is an adimensional measure, indicating how good (small) is the elapsed trip-time T with regard to what has been observed on average until that instant. Experimentally, r' is expressed as:

$$r' = \begin{cases} \frac{T}{c\mu} & c \geq 1 \text{ if } \frac{T}{c\mu} < 1 \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

where: μ is average of the observed trip-time T ; and. c is a scale factor experimentally chosen as 2 [Dorigo et al 97].

More details about r' and its significance can be found in [Dorigo et al 97].

b) The other neighboring nodes ($j\neq f$) P_{dj} probabilities associated with node k are diminished, in order to satisfy equation (1), through the expression:

$$P_{dj} \leftarrow P_{dj} -(1 - r')P_{dj} \; ; \forall j \in N_k , \; j \neq f \tag{5}$$

A list $trip_k(\mu_i,\sigma_i^2)$ of estimate arithmetic mean values μ_i and associated variances σ_i^2 for trip-times from node k to all nodes i ($i\neq k$) is also updated. This data structure represents a *memory* of the network state as seen by node k. The list trip is updated with information carried by $B_{s\to d}$ ants in their stack $S_{s\to d}$. For any node pair source-destination, μ after (n+1) samples (n>0) is calculated as follows:

$$\mu_{n+1} = \frac{n\mu_n + x_{n+1}}{n+1} \tag{6}$$

where: x_{n+1} trip-time T sample n+1, and μ_n is the arithmetic mean after n trip-time samples.

2.2 AntNet2.0 Algorithm

AntNet2.0 is an AntNet1.0 modified version [Dorigo et al 98] with the five main steps of the presented pseudocode, which perform basically the same actions as AntNet1.0. The differences are how the routing tables and the lists trips (now known in as traffic local model M_k) are updated, so, only these two differences will be explained.

Suppose that a $B_{s\to d}$ arrives to a node k, in its return trip to node s. The $B_{s\to d}$ ant will update the traffic local model M_k (list trip in AntNet1.0) and the neighbor nodes probabilities of k, associated to node d in the routing table Γ_k. Also, as in AntNet1.0 step 5, the update is performed in the entries corresponding to every node $k' \in S_{s\to d}$, $k' \neq d$ in the subpaths followed by $F_{s\to d}$ after visiting k. If a subpath trip-time T is statistically good (i.e: T is smaller than $\mu + I(\mu,\sigma)$, where I is a μ interval confidence estimator), then T is used to update the statistics related and the routing table. However, if T is bad, it is not used, because it does not give a true idea about the required time to arrive to the subpath nodes. The traffic local model M_k and the routing table Γ_k are updated for a generic destination $d' \in S_{s\to d}$ in the following way

1. M_k is updated with the values carried in $S_{s\to d}$. The trip-time $T_{k\to d'}$ employed by $F_{s\to d}$ to travel from k to d' is used to update $\mu_{d'}$, $\sigma_{d'}^2$ and the best observed value inside window $W_{d'}$ according to the expressions:

$$\mu_{d'} \leftarrow \mu_{d'} + \eta(T_{k\to d'} - \mu_{d'}) \qquad (7)$$

$$\sigma_{d'} \leftarrow \sigma_{d'} + \eta((T_{k\to d'} - \mu_{d'}) - \sigma_{d'}) \qquad (8)$$

   where η is the weight of each trip-time observed. The effective number of samples will be approximately 5(1/η). Therefore, for 50 samples, η=0.1, and for 100 samples η=0.05. The role of $W_{d'}$ will be explained later.

   The $T_{k\to d'}$ mean value and its dispersion could vary strongly, depending on traffic conditions: a poor (large) time with low data traffic could be very good with relation to another measure with more traffic. The statistical model should reflect this variability and continue the traffic fluctuations in a robust way. This model plays a critical role in routing table updating.

2. The routing table for k is updated in the following way:

   a) The value $P_{fd'}$ (the probability for selecting the neighbor node f, when the node destination is d') is incremented by means of the expression:

$$P_{fd'} \leftarrow P_{fd'} + r(1-P_{fd'}). \qquad (9)$$

   where r is a reinforcement factor indicating the goodness of the followed path.

   b) The $P_{nd'}$ probabilities associated to other nodes decrease respectively:

$$P_{nd'} \leftarrow P_{nd'} - r\, P_{nd'}. \quad n \in N_k,\ n \neq f \qquad (10)$$

   The factor of reinforcement r is calculated considering three fundamental aspects: (i) the paths should receive an increment in their probability of selection, proportional to their goodness; (ii) the goodness is a traffic condition dependent measure, that can be estimated by M_k,; and (iii) they should not continue all the traffic fluctuations in order to avoid uncontrolled oscillations. It is very important to establish a commitment between stability and adaptability. Between several tested alternatives [Dorigo et al 98], expression (11) was chosen to calculate r:

$$r = c_1\left(\frac{W_{best}}{T}\right) + c_2\left(\frac{I_{\sup} - I_{\inf}}{(I_{\sup} - I_{\inf}) + (T - I_{\inf})}\right) \qquad (11)$$

   where: W_{best}. is the best trip of an ant to node d', in the last observation window $W_{d'}$,
   $I_{\inf} = W_{best}$ (lower limit of the confidence interval for μ,
   $I_{\sup} = \mu + z * \left(\sigma/\sqrt{|w|}\right)$. (upper limit of the confidence interval for μ, with:
   $z = 1/\sqrt{(1-\gamma)}$, γ = confidence level, $\gamma \in$ [0.75, 0.8].
   c_1 and c_2 are weight constants, chosen experimentally as c_1=0.7 y c_2=0.3.

More details about r and its significance can be found in [Dorigo et al 98].

3. Modifications Proposed to AntNet

This section describes several modifications tested for AntNet1.0 and AntNet2.0, in order to improve their performances.

3.1. Intelligent Initialization of Routing Tables

AntNet1.0 and AntNet2.0 do not specify a initialization method for the routing tables [Dorigo et al 97][Dorigo et al 98]. For this reason, a uniform distribution of probabilities is assumed, according to the initialization given in the pseudocode. Due to this situation of null a-priori knowledge, an initialization of each node routing table that reflects a previous knowledge about network topology is proposed. Furthermore, an initial greater probability value is assigned to the neighboring nodes that simultaneously could be destinations. For a node k this could be described as follows:

1. If a destination node d for a table entry is at the same time a neighbor node, that is $d \in N_k$, then the initial probability in the routing table of k is given by:

$$P_{dd} = \frac{1}{n_k} + \frac{3}{2} * \frac{(n_k - 1)}{n_k^{\,2}} \tag{12}$$

For the rest of the neighboring nodes $i \in N_k$, it will be:

$$P_{di} = \begin{cases} \frac{1}{n_k} - \frac{3}{2} * \frac{1}{n_k^2} & \text{if } n_k > 1 \\ 0 & \text{if } n_k = 1 \end{cases} \tag{13}$$

Of course, (12) and (13) satisfy (1).

2. If the destination d is not a neighbor node, that is $d \notin N_k$, then a uniform distribution is initially assumed:

$$P_{di} = \frac{1}{n_k}. \tag{14}$$

Due to the advantage of network topolog knowledge reflected by the initial probability values in the routing tables, this method showed a transient regime shorter than the observed during simulations on AntNet1.0 and AntNet2.0.

3.2. Intelligent Updating of Routing Tables after Network Resources Failures

Original AntNet algorithms [Dorigo et al 97][Dorigo et al 98] do not mention the following cases:

1. Routing tables updating in case of links or node failure, that is, immediately after a node k loses its link l_{kj} with its neighbor node j. At first, it is supposed that if an ant is in k, the probability P_{dj}, of arriving to a destination d across a jumping node j, i.e. the probability of using the link l_{kj}, is distributed uniformly between the remaining n_k-1 neighbor nodes for the entry d in the routing table of k. Mathematically, P_{dj} =0 during a link l_{kj} failure (it isn't possible to jump from k to j to arrive to d).

$$P_{di} = P_{di} + \frac{P_{dj}}{n_k - 1}. \quad \forall i \neq j \quad i, j \in N_k \tag{15}$$

Alternatively, the present work proposes the idea of new P_{di} values immediately after the l_{kj} link failure. These probabilities will be proportional to their relative values, before the failure, according to the acquaintance until that instant, instead of "forgetting" everything he learned until the moment of the failure, according to (15). So, in a node k, after the l_{kj} link failure, a factor Q is calculated like:

$$Q = \frac{P_{dj}}{1 - P_{dj}}. \tag{16}$$

then, P_{di} is updated according to:

$$P_{di} = (1 + Q) * P_{di}. \quad \forall i \neq j; \quad i \in N_k, P_{dj}=0. \tag{17}$$

This method reflects the node knowledge about the network traffic and topology before the failure, so during the event the algorithm should show a better performance than the original algorithms.

2. Routing table updating for the k-j node pair when the link l_{kj} is up at time t_2 (t_2>0), since this link was down at time t_1, 0<t_1<t_2. AntNet1.0 and AntNet2.0 use a routing table reinitialization for k and j according to (2), losing all information learned right before the link failure. As an alternative, this work proposes a reinitialization subject to a commitment between learned information until instant t_1, before the link failure, and total ignorance of the node as in t=0. So, the probabilities in the routing table for a node k, whose link failed in t_1, but was recovered in t_2 will be:

$$P_{di}(t_2) = (1-\lambda)P_{di}(0) + \lambda P_{di}(t_1). \quad 0 \leq \lambda < 1 \tag{18}$$

The factor λ is a constant, known a *coefficient of memory*, since its value indicates how much it remembers what it has learned until time t_1. An empirical value of 0,6 was adopted. This criterion makes the algorithm more robust allowing a faster recovery time after link or node failure.

3.3. Introduction of a noise factor and limitation of probability values

With the routing tables updating methods in AntNet1.0 and AntNet2.0, the distribution of probabilities eventually "would freeze" with an probability value, near to one, with the rest of them remaining with insignificant values. Thus, in any node, ants and data packets would mostly choose the output line with the highest probability. In order to prevent this, a noise factor *f* is defined. Every time an ant should jump to a following node, it chooses that node with a probabilit *f*, according to an uniform distribution probability, and with a probability (1-*f*), according to the values stored in the routing tables [Schoonderwoerd97]. With this, the ants " could discover new and better paths by "accident. Then, potentially both the delay and throughput could improve.

Also, any probability in the routing tables was limited to a maximum value of: P=1-(n_k-1)*ε, (experimentally ε=0.05), being ε, the minimum probability admitted.

3.4. Dual Method for selection of jumping node

In the AntNet1.0 and AntNet2.0 algorithms, being in a node *k*, a data packet, whose destination is a node $d \neq k$, will select a jumping node *j* randomly, according to P_{dj}, $\forall j \in N_k$. Also, this work considers the possibility of selecting directly the output line corresponding to node *j* with the highest probabilit value in the routing table of *k*, between the n_k probabilities associated to node *d*. This last method is called hierarchical method [Schoonderwoerd97].

As mentioned before, in each node, packets will decide randomly whether to use the usual method (random) or the hierarchical method, in order to choose the jumping node. Particularly, a better behavior for the case of P = 0.5 was observed, where P is the probability of using the random method, normally used in AntNet. So, for a data packet, there will exist a probabilit $P = 0.5$ of using the rando approach, and a probabilit $\overline{P} = 0.5$ of directl using a node, whose probability is the highest in the *k* routing table, for the destination *d*. For AntNet1.0 and AntNet2.0 P=1 is considered.

3.5. Control of the number of ants in the network

AntNet1.0 and AntNet2.0 do not mention any method to maintain control of the total numbers of ants moving inside the network, which, under certain circumstances, could contribute to congestion. In order to control the number of ants, it was first attempted to limit the maximum number of ants in an amount equal to the square of the number of network nodes. This approach was computationally very heavy, in addition to requiring very large data structures. For this reason, the number of ants was limited to an amount four times the number of network nodes. With this alternative, the simulation results were improved and the computing load diminished. So, this approach was adopted for the implemented algorithms.

3.6. Self-destruction of a Backward Ant

Self-destruction of a backward ant $B_{s \to d}$ refers to a $B_{s \to d}$ ant that can not return to its source node, because its return path was cut, due to a link or node failure. Under this situation the ant is self-destroyed, because the information stored in its stack already does not reflect the real state of the network. This point was very important, so it was added to all the AntNet algorithms, including AntNet1.0 and AntNet2.0.

4. Experimental Results

With the proposed modifications two alternative algorithms were implemented:

1. AntNet1.1. It is a modified version of AntNet1.0.

2. AntNet2.1. It is a modified version of AntNet2.0.

The main characteristics of the implemented algorithms are:

- All algorithms are implemented in C language.

- The parallel behavior is simulated with a serial code.

- The data traffic simulation analysis is performed after each time slot.

For the simulations, three networks have been used as models:

1. A simple network, with eight nodes and links with unitary cost, given in Figure 1 [Dorigo et al 98].

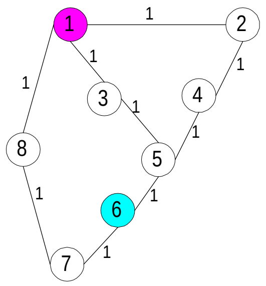
Figure 1. Simple Network

2. The NSFNET (National Science Foundation network) from the United States, with fourteen nodes and 1.5 Mbps links. Figure 2 presents its link delays in [ms].

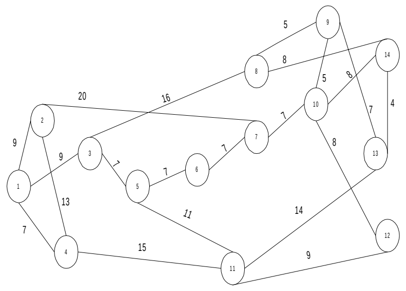
Figure 2. NSFNET

3. The NTT network (Nippon Telephone Telegraph) of Japan, composed by fifty nodes and links of 6 Mbps. It is showed in Figure 3.

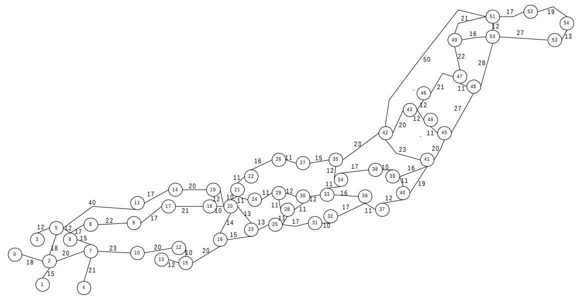
Figure 3. NTTnet

A benchmark was established for the simulations. twelve simulation scenarios compose our benchmark, as shown in Table 1. Each scenario was tested for each of the three model networks presented above.

Traffic	Lost Packet threshold	Transient Regime	Link Fail	Node Fail	Hot Spot
Low	5%	✓	✓	✓	✓
Medium	10%	✓	✓	✓	✓
High	20%	✓	✓	✓	✓

Table 1. Benchmark for each model network

For each simulation cycle, the traffic simulator will stop generating packets when a certain fraction (expressed in %) of the generated packet has not arrived to destination (Lost Packet threshold). The link transmission delay is used as metric for link costs, expressed in milliseconds. The performance parameters for the algorithms are:

- *Instantaneous Packet Delay*. It is the average delay of all data packets routed successfully for a given instant *t* in the algorithm simulation.
- *Average Packet Delay*. It is the average delay of all data packets routed successfully during the whole simulation period.
- *Instantaneous Throughput*. It is the amount of packets routed successfully for a given instant *t* in the algorithm simulation.
- *Average Throughput*: It is the amount of packets routed successfully during the whole simulation period.

Figures 4 to 7 and Tables 2 to 4 show the simulation results for some of the experiments performed for the three mentioned networks and only for medium traffic. It is important to mention that the obtained results for the LBR algorithms were not as good as first expected, so they will not be presented nor discussed for any of the experiments to be considered in this publication. In the tables that follow, THR stands for average throughput and for AVP for average packet delay. The following abbreviations will be used for AntNet algorithms: AntNet1.0=A1.0, AntNet1.1=A1.1, and so on.

In what follows, the simulations results for the networks mentioned will be presented:

1. Simple Network (Figure 1): It is considered to compare the routing methods used by the traditional algorithms RIP and OSPF with A1.0.

For this, data traffic between nodes 1 and 6 was simulated, with the following results.

a) Throughput: A1.0 have a performance about three times better than RIP (Table 2). This is because there are three possible paths between nodes 1 and 6; two of which have the same cost. In spite of this, RIP and OSPF send the packets only through one path (the chosen best path). On the contrary, A1.0 distributes the load between the three paths, proportionally to their goodness (cost).

b) Packet Delay: In table 2 a greater packet delay for A1.0 is observed, because approximately the third part of the packets is sent through the longest path.

Algorithms	THR [packets]	AVP [ms]
RIP	2367	3.01
OSPF	2589	3
A1.0	7245	3.03

Table 2: Average parameters results

Clearly, and as it was expected, AntNet has a better throughput at a price of a larger packet delay, a situation that is found in almost every experiment.

2. NSFNET (Figure 2): results for transient regime experiment for AntNet algorithms (Table 3) and for a transitory link failure experiment (link 5-6, see Fig. 3) (Figures 4-5, Table 3) are shown here, concluding the following:

a) Transient Regime: Table 3 indicates how the modified algorithms "learn" quicker (better throughput and packet delay) than A1.0 and A2.0, due to the use of routing tables intelligent initialization (section 3.1) and the dual method for jumping node selection by data packets (section 3.4). Also, the superiority of A2.1 is observed between all the algorithms.

b) Link 5-6 Failure: Throughput. RIP and OSPF decay completely at the instant of the failure (Figure 4); however, the AntNet algorithms are not severely affected, demonstrating their robustness for this type of failures. In particular, it is observed in Table 3 that A2.1 has the best average throughput. Also, A1.1 overcome A1.0 (Table 3), due to routing tables intelligent reinitialization method (section 3.2).

c) Link 5-6 Failure: Packet Delay. All algorithms are proportionally affected (Figure 5), while the inherent advantage of RIP and OSPF in this figure of merit remains. Here, the modified AntNet algorithms also overcome A1.0 and A2.0.

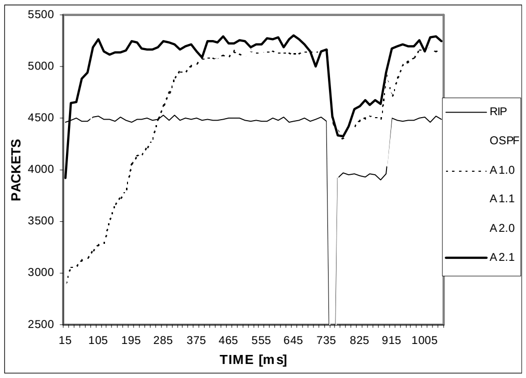

Figure 4. Link 5-6 failure. Instant. Throughput

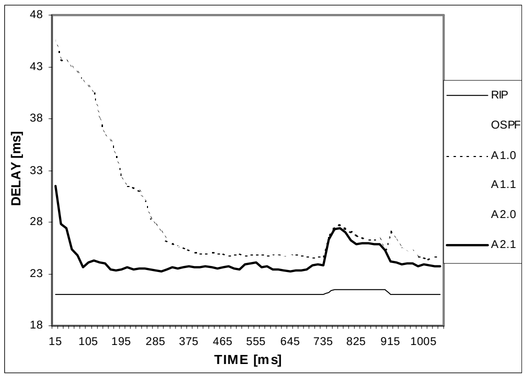

Figure 5. Link 5-6 failure: Instant. Packet Delay

	Transient Regime		Link 5-6 Failure	
	THR [packets]	AVP [ms]	THR [packets]	AVP [ms]
A2.1	5008.4	24.27	5081.3	24.25
A2.0	4716.79	27.17	4844.56	25.58
A1.1	4509.77	28.72	4869.94	26.81
A1.0	3572.75	37.79	4609.02	28.53
OSPF			4450.33	20.1
RIP			4347.61	21.06

Table 3. Experimental Results for Average throughput and Average packet delay

3. NTTnet (Figure 3): Experimental results are showed for link failure and hotspot.

   a) Link 31-32 failure Throughput. AntNet algorithms are more robust than RIP and OSPF. A2.1 has the best average THR (Table 4) and A1.1 is better than A2.0.

   b) Link 31-32 Failure: Packet Delay. Again, our AntNet are the best (Table 4).

   c) Transient Hotspot: Throughput. Node 41 was chosen as a hotspot (Figure 3). Here, A2.1 is the best (Table 4), although finally, all AntNet converge to a similar behavior (Figure 6). AntNet algorithms show small oscillations during the hotspot.

   d) Transient Hotspot: Packet Delay. A2.1 has the best behavior in presence of a hotspot (Figure 7), possibly due to change in the data traffic spatial distribution, caused by the hotspot. Table 4 shows the best average values for our AntNet.

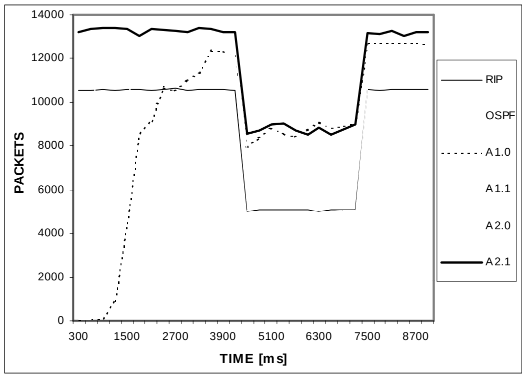

Figure 6. Transient Hotspot. Instant. Throughput

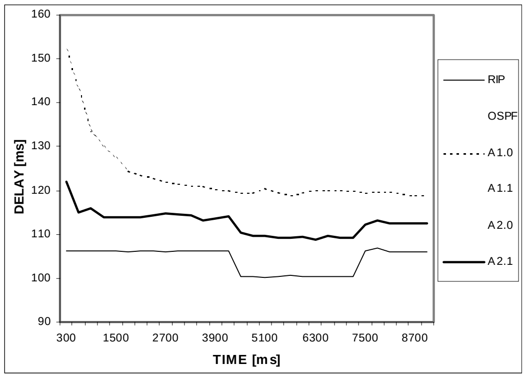

Figure. 7. Transient Hotspot. Inst. packet delay

	Link 31- 32 Fail		Transient HotSpot	
	THR [packets]	AVP [ms]	THR [packets]	AVP [ms]
A2.1	12879	114.53	11759	112.58
A2.0	10774.9	118.02	9423.13	116.25
A1.1	11061.8	120.72	11065.6	119.41
A1.0	9717.1	124.16	8891.3	123.19
OSPF	10803.3	104.61	8848.26	102.63
RIP	10201.8	107.49	8736.23	104.26

Table 4. Experimental Results for Average throughput and Average packet delay

After the analysis of simulation results, these general conclusions can be summarized:

- Proposed modifications of original AntNet have shorter transient regime than A1.0 and A2.0 (Table 3).
- AntNet algorithms are more robust than RIP and OSPF algorithms for link failure, because their instantaneous throughput does not decay completely at the time of failure (Tables 3-4).
- For hotspots, the results suggest that AntNet algorithms are sensitive to changes in data traffic geographical distribution, because oscillations in the instantaneous packet delay were observed, during the presence of a hotspot (Figures 6, 7).
- In most of the experiments A2.1 showed the best performance.

Among all the AntNet algorithms, A1.0 showed the worst performance and it never performed better than A2.0, due to the superior method used for the routing tables and trip lists updating [Dorigo et al 98]. However, A2.0 proved worse than A1.1 in some circumstances (Figures 11, 13). After the analysis of simulation results, these general conclusions can be done:

- Our AntNet have shorter transient regime than A1.0 and A2.0 (Table 3).
- AntNet algorithms are more robust than RIP and OSPF algorithms for link failure, because their instantaneous throughput does not decay completely at instant of the failure (Tables 3-4).
- For hotspots, the results suggest that AntNet algorithms are sensitive to changes in the data traffic geographical distribution, because oscillations in the instantaneous packet delay

were observed, during the presence of the hotspot (Figures 6, 7).

- In most of the experiments A2.1 showed the best performance.

Among all the AntNet algorithms, A1.0 showed the worst performance, and it never performed better than A2.0, due to the superior method used for the routing tables and trip lists updating [Dorigo et al 98]. However, A2.0 proved worse than A1.1 in some circumstances (Figures 11,13).

5. Conclusions

In this work two versions of AntNet algorithms, -a novel adaptive routing technique for data networks, based on mobile agents, oriented towards packet switching, such as Internet-, were studied. Then, two improved versions were presented.

AntNet algorithms, in addition to RIP, OSPF and LBR [Back99] were implemented and simulated. A better performance of our versions of AntNet was observed in most of the experiments. The modifications implemented in our versions that contributed the most for a better behavior were: a) routing tables intelligent initialization and, b) dual method for selecting jumping node for data packets.

In general, the results of the experiments remained proportional [Sosa00]. Results obtained in a different simulation scope suggest that AntNet algorithms could have better throughput as well as packet dela than RIP and OSPF [Dorigo et al 97] [Dorigo et al 98]. If this is the case, it is expected that the modified algorithms proposed here will have better performance than the original AntNet versions.

An efficient AntNet behavior with flow control, congestion and admission schemes is also expected [Barán et al 00]. Therefore, it can be inferred that commercial implementation of this algorithm may be feasible and it can even be considered its usage in large networks, such as Internet, as a future option.

Acknowledgements

This work was partially supported by a DIPRI Research Grant of the National University of Asuncion.

References

[Barán et al 99] B. Barán, M Almirón, E. Chaparro. 'Ant Distributed System for Solving the Traveling Salesman Problem'. pp. 779-789. Vol. 2 15 th Informatic Latinoamerican Conference-CLEI,. Paraguay (1999).

[Barán et al 00] B. Barán, R. Sosa. 'A New Approach for Antnet Routing'. pp. 303-309. IEEE 9 thInternational Conference on Computer Communications and Networks. USA (2000).

[Bak et al 99] S. Bak, J. Cobb, E. Leiss. 'Load Balancing Routing via Randomization'. pp. 999-1010. Vol. 2. 15 th Informatic Latino american Conference-CLEI. Paraguay (1999).

[Dorigo et al 96] M. Dorigo, V. Maniezzo, A. Colorni. 'The Ant System: Optimization by a colony of cooperating agents'. pp. 1-13. Vol. 26-Part B. IEEE Transactions on Systems, Man, and Cybernetics,. Number 1 (1996).

[Dorigo et al 97] M. Dorigo, G. Di Caro. 'AntNet. A Mobile Agents Approach to Adaptive Routing'. Technical Report. IRIDIA Free Brussels University. Belgium (1997).

[Dorigo et al 98] M. Dorigo, G. Di Caro. 'AntNet. Distributed Stigmergetic Control for Communications Networks'. pp. 317-365 Journal of Artificial Intelligence Research. Number 9 (1998)

[Feit96] S. Feit. 'TCP/IP. Architecture, Protocols and Implementation.'. McGraw & Hill (1996).

[Rutkowski98] T. Rutkowski. 'Dimensioning the Internet'. pp. 8-10. Vol 2. IEEE Internet Computing. Number 2 (1998).

[Schoonderwoerd et al 97] R. Schoonderwoerd, O. Holland, J. Bruten. 'Ant-like agents for load balancing in telecommunications networks'. Hewlett-Packard Laboratories, Bristol-England (1997)

[Shankar et al 92] A. Shankar, C. Alaettinoglu, I. Matta. 'Performance Comparison of Routing Protocols using MaRS. Distance Vector versus Link-State'. Technical Report, Maryland-USA (1992)

[Tanenbaum96] A. Tanenbaum. Computer Network. Prentice & Hall. Third Edition (1996).