The Antnet Routing Algorithm - A Modified Version

F. Tekiner\textsuperscript{a}, F. Z. Ghassemlooya and S. Al-khayatt\textsuperscript{b}
\textsuperscript{a}: Optical Communications Research Group, School of Engineering and Technology, Northumbria University, Newcastle upon, Tyne, NE1 8ST, UK.
\textsuperscript{b}: School of Computing and Management Sciences, Sheffield Hallam University, Sheffield, S1 1WB, UK.
Email: ftekiner@ieee.org

Abstract – Antnet is an agent based routing algorithm that is influenced from the unsophisticated and individual ant’s emergent behaviour. Ants (software agents) are used in antnet to collect information and to update the probabilistic distance vector routing table entries. Modified antnet algorithm has been introduced, which improve the throughput and average delay. Results shows that by detecting and dropping 0.5% of packets routed through the non-optimal routes the average delay per packet decreased and network throughput can be increased. The effect of the traffic fluctuations has been limited with the boundaries introduced in this paper and the number of ants in the network has been limited with the current throughput of the network at any given time.

I. INTRODUCTION

In today’s fast growing Internet traffic conditions changes and failures occurs at some parts of the network from time-to-time, in an unpredictable manner. Therefore, there is a need for an algorithm to manage traffic flows and deliver packets from the source to the destination in reasonable time.

In a network there are more than one optimal path exist. The challenge is to deliver packets to their destinations with minimum time delay, if necessary routing them via more nodes. However, it is neither realistic nor feasible to have a common control unit that controls all links and processing elements (nodes) in the network or part of the network. Moreover, such systems are not fault tolerant, since a node failure will result in unavoidable network delays and shut down of some parts of the network. Therefore, the solution would be to divide the task into a number of sub-tasks distributed across the entire network. Moreover, the network should have a capability of organising/reorganising itself whenever changes take place within the network.

Routing algorithm is the key element in networks performance and reliability, thus it can be seen as the “brain” of the network. An ideal routing algorithm should be node and link independent, and be able to deliver packets to their destination with the minimum amount of delay, regardless of their size and traffic load. Formally speaking, a routing algorithm should be intelligent, adaptive and fault tolerant. The only way to achieve this would be by employing an intelligent and distributed routing algorithm.

The routing algorithms currently in use [1] lack intelligence, and need human assistance and interpretation in order to adapt themselves to failures and changes. Moreover, they are mainly table-based and search process is highly undesirable. Routing is considered to be NP-Hard Optimization problem, therefore widely used optimization problems have been applied widely. To name a few, Genetic Algorithms, Neural Networks, Simulated Annealing, Software Agents and Reinforcement Learning [2].

In recent years, agent based systems and reinforcement learning have been widely applied to routing. This is because these methods do not need any supervision and are distributed in nature. Swarm intelligence particularly ant based systems [3] [4], Q-learning [5] methods and hybrid agent based Distance Vector algorithms [6] have also shown promising and encouraging results. In this paper, the focus will be on the antnet routing algorithm.

II. SOFTWARE AGENT AND ANT BASED ROUTING

Unlike other optimisation problems, the routing problem has to be solved in real time since it is a dynamic optimisation problem. The ant-based approach applied to routing problem was first reported in [7]. This work was influenced from the work done on the software agents used for control in telecommunication networks [8]. Improved version of the algorithm in [7] was applied to
the connection oriented systems [9]. For the first time ant based routing was applied to the packet based connection-less systems [10]. This was followed by a mobile agents approach to adaptive routing in [11].

A. Antnet Routing Algorithm
Individual ants are unsophisticated and simple insects, but collectively that are capable of performing a variety of complicated tasks [12]. Antnet [13] [14] is an agent based routing algorithm that is influenced from the real ants behaviour. In antnet ants (nothing but software agents) explores the network to find the optimal paths from the randomly selected source destination pairs. Moreover, while exploring the network ants updates the probabilistic routing tables and builds a statistical model of the nodes local traffic. Ants use these tables to communicate with each other. The algorithm uses two types of agents namely, forward agents and backward agents to collect network statistics and to update the routing table. In each node there are 2 types of queues, low priority and high priority. The data packets and the forward agents use low priority queues, whereas the backward agents use the high priority queues. Later forward agents do also use the high priority queues [14].

Agent communicate through the two data structures stored in every node (see figure 1) as outlined below [13,14]:

i. A distance vector routing table $T_k$ with distance metric defined with probabilistic entries where for each possible destination $d$ and neighbour node $n$ there is a probability value $P_{nd}$, which reflects the goodness of the link (path), given as:

$$\sum_{n \in N_k} P_{nd} = 1, \quad d \in \{1, N\}$$

$$N_k = \{\text{Neighbours}(k)\}$$

ii. An array $M_k(\mu_d, \sigma_d^2, W_d)$ defines a simple statistical traffic model experienced by the node $k$ over the network. Where $W_f$ is the observation window used to compute the estimated mean $\mu_d$ and the variance $\sigma_d^2$ parameters given as, respectively:

$$\mu_d = \mu_d + \eta(o_{k \rightarrow d} - \mu_d)$$

$$\sigma_d^2 = \sigma_d^2 + \eta((o_{k \rightarrow d} - \mu_d)^2 - \sigma_d^2)$$

Where $o_{k \rightarrow d}$ is the new observed trip time experienced by the agent while travelling from node $k$ to destination $d$.

The antnet behaviour is summarised as [13]:

1. At regular intervals (defined by the user) from every node an agent $A$ is sent to a randomly selected destination node $d$.
2. Each agent first defines the possible neighbour nodes (unvisited neighbour nodes) at the current node by using its routing table. Then, an agent chooses the next node $n$ within the identified possible (unvisited) nodes by using the probabilistic values in the table by taking into account the state of the associated queue as follows:

$$P_{nd}' = \frac{P_{nd} + \alpha l_n}{1 + \alpha (N_k - 1)}$$

Where $l_n$ is the heuristic correction value with respect to the probability values stored in the routing table that gives a quantitative measure of the queue waiting time, which is defined as:

$$l_n = 1 - \frac{q_n}{\sum_{n'=1}^{N_k} q_{n'}}$$

Where, $q_n$ is the bit length (or number of packets if packet size is fixed) waiting to be sent to the queue on the output port $n$ of the node $k$.

3. If the selected port is full, then an agent is directed to the FIFO output buffer associated for that port and waits for its turn. It is assumed that the buffer size is infinitely large.
4. The identifiers of every visited node and time elapsed since the agent is despatched

---

$^1\eta$ weights the number of samples that affect the average and set to 0.05[13].

$^2\alpha$ weights $l_n$ with respect to routing table entries and set to 0.3[13].
from the source are stored in the stack \( S(k) \) carried by the agent.

5. If an agent is forced to visit previously visited node, hence if cycle exists, then it deletes all the entries for the nodes associated with the cycle.

6. When an agent reaches the destination it changes its type as backward agent and by using the same path, to reach the destination, it returns back to the source node. On its way back, agent pops every visited node from its stack and updates the associated routing table entries (probabilities) for all the nodes along the path by using the following rules:

   (i) \( M_k (\mu_d, \sigma_d^2, W_d) \) is updated with values stored in the stack memory \( S(k) \). The time elapsed to arrive (for the forward ant) to the destination \((d_k \rightarrow d)\) is used to update the estimated mean \( \mu_d' \), variance \( \sigma_d'^2 \) and the best trip value over the observation window \( W_d' \).

   (ii) The routing table \( T_k \) is updated by incrementing the probability of selecting neighbour \( f \) when the destination is \( d' (P_{df}) \) is given as:

   \[
   P_{fd'} = P_{fd} + r(1 - P_{fd'})
   \]  
   
   Probabilities \((P_{nd'})\) for rest of the neighbour nodes are updated as:

   \[
   P_{nd'} = P_{nd'} - rP_{nd'}, n \in N_k, n \neq f
   \]

   \( r \) defines the goodness of the path taken by the ant, which is used as the only feedback (reinforcement) information to the routing tables. \( r \) is defined as follows:

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

   Where, \( T \) is the observed trip time, \( c_1 \) and \( c_2 \) are constants used to control the effect of the last \( T, W_{best} \) is the ant’s best trip time for a given destination and last observation period. \( I_{inf} \) and \( I_{sup} \) are the lower and upper limits for the confidence interval \( \mu \), respectively. Where \( I_{inf} = W_{best} \) and \( I_{sup} = \mu \ast z \ast (\sigma / \sqrt{\langle w \rangle}) \). More details and discussion can be found in [13].

### III. ANTNET MODIFICATIONS AND RESULTS

Based on the original antnet routing algorithm three modifications has been proposed and tested.

#### A. Deleting aged packets

During the simulations run on the original antnet routing algorithm it was discovered that some packets travel within the network for a number of hops until they reach their destination. This problem occurs because the routing tables used in the antnet routing algorithm is probabilistic and therefore a few packets have chance to be directed to the non-optimal routes and cycle. A simple rule is defined to detect and drop these packets from the network as follows:

\[
\text{if PACKET AGE} > 2 \times \text{NO_OF_NODES} \quad \text{then DROP PACKET}
\]

This rule is defined based on the information gained from experimental results. It was observed that only 0.5% of the packets experience this problem. Therefore, only 0.5% of the packets are dropped from the network. However, when the packet age condition is set to \( 1\times\text{NO_OF_NODES} \), the packet loss increased to almost 7%. On the other hand, when the condition is set to \( 3\times\text{NO_OF_NODES} \), the loss decreased to 0.3% with no further improvement in the performance.

#### B. Limiting the effect of \( r \) due to traffic fluctuations

The reinforcement \( r \) applied to the routing table entries is limited by the lower and upper bounds defined as follows:

\[
\begin{align*}
\text{if} \ (\text{NO_OF_NODES} < 5) & \quad 0.1 < r < (1 - 0.1 \times \text{NO_OF_NODES}) \\
\text{else} & \quad \text{if} \ (\text{NO_OF_NODES} > 5) \quad 0.05 < r < (1 - 0.05 \times \text{NO_OF_NODES})
\end{align*}
\]

The values used are based on experimental results and it is intended to limit the effect of the traffic fluctuations in the network at a given time. Similar method to control the effect of \( r \) has also been reported in [15]. However, if \( r \) does not satisfy these values for three consecutive times and is less than 0.95, then reinforcement is applied to the routing table entry.

It is believed that by limiting the impact of \( r \) on the routing table entries the algorithm

---

\( c_1 \) and \( c_2 \) are chosen experimentally to be 0.3 and 0.7 respectively.

\( z = \frac{1}{\sqrt{1 - \gamma}} \), \( \gamma \) lies is the confidence, \( \gamma \in [0.75, 0.8] \).
would not freeze as easily as it was on the originally proposed algorithm.

C. Limiting the number of Ants in the system.

In the modified version of the algorithm, the number of ants is limited by the ant/packet ratio. Although, this has no major impact on the performance of the algorithm (when compared with results on $1.0 > ant\ creation\ rate > 0.1$), it would increase the adaptability and suitability of the algorithm. In other words it makes algorithm more generic.

IV. EXPERIMENTAL RESULTS

With the proposed modifications both the original and the modified antnet routing algorithms were implemented in the following environment.

- Algorithms are implemented in the C language in a parallel environment by using PVM.
- Parallel behaviour is simulated by assigning every node to a different process both on the same machine and different machines.
- Poisson traffic distribution is used with three different system loads: low, medium, and high.
- Algorithm is given 5 seconds to discover all the paths and initialise the probabilistic routing table entries.
- NSFNET (figure2) is used as the network topology with each link having equal cost.
- For each simulation, packet generation is stopped after creating 5000 packets per node and simulation is stopped after all packets are arrived to their destinations or detected and deleted from the network.
- Every simulation is run 8 times and the average of the results is used for accuracy.
- It is assumed that there is no packet loss.
- Packet size is fixed.
- All experiments are implemented for varying ant creation rates, since it has a significant effect on the performance of the algorithm.

The performance parameters for the simulations are:

i. Average packet delay: is the average delay a packet experiences while being routed from source to destination.

ii. Average throughput per packet: is the average number of packets being forwarded by node for the duration of the simulation.

From figure 3, it can be seen that as the number of ants increases in the network the average packet delay increases. However, with the modified algorithm, packet delay is reduced compared with the original routing algorithm. This is because in the original routing algorithm some packets travel with very high no of hops within the network thus uses a lot of network resources and occupies a considerable amount of bandwidth.

Moreover, further simulations for ant creation rates greater than 1.0 showed that the average delayed increased and throughput of the algorithm decreased slightly in both algorithms. Therefore, the ant rate between 1 and 0.1 is optimal for our system. However, since node and link failures has not been implemented and investigated in this work it is not possible to comment on the effect of ant rate on the performance of the network in problematic conditions.

\[
\frac{NO\ OF\ ANTS\ CREATED}{NO\ OF\ PACKETS\ SEND} = 0.001
\]

\text{Figure 2: NFSNET topology.}

\text{Figure 3: Average packet delay vs. ant creation rate.}

\text{avg. Packet Delay(s)}

\text{Middle-Mod}  \quad \text{Mid-Orig}

<table>
<thead>
<tr>
<th>Ant Rate</th>
<th>Avg. Packet Delay (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>1.150</td>
</tr>
<tr>
<td>0.100</td>
<td>1.050</td>
</tr>
<tr>
<td>0.010</td>
<td>1.000</td>
</tr>
<tr>
<td>0.001</td>
<td>0.950</td>
</tr>
</tbody>
</table>

Ant creation rate defines the frequency (in time) of the forward ants to be created by the node. Therefore, a low ant rate, means a high number of ants in the system.
Figure 4 shows the throughput against the ant rate for three different system loads for the original and the modified antnet routing algorithms. For all system loads there is an improvement of approx. 0.25 packet/time in the throughput when using the modified algorithm. As the ant rate increases the throughput decreases, but having more ants increases the accuracy of the probability entries of the routing tables. However, some of the traffic created is artificial due to the ants travelling in the system.

V. CONCLUSIONS AND FURTHER WORK

In this paper, it has been shown that by detecting and dropping packets that travel continuously within the network can improve the antnet’s performance in terms of network throughput and the average packet delay. The effect of traffic fluctuations on the network performance has been limited by the introduction of boundaries, by limiting the number of ants within the network at any given time. Although limiting the number of ants increases the network utilization, it reduces the chance of finding the best and new routes and detecting failures and problematic conditions. In future work the ratio proposed for the ant creation rate will be investigated for problematic conditions and hotspot traffic together with the other network topologies.

REFERENCES