Malaysian Journal of Computer Science, Vol. 16 No. 2, December 2003, pp. 1-10

ADAPTIVE ROUTING IN PACKET-SWITCHED NETWORKS USING AGENTS UPDATING METHODS

Mohammad Saleh
Division of Computer Science & IT
University of Nottingham (Malaysia Campus)
Wisma MISC, No. 2 Jalan Conlay
50450 Kuala Lumpur, Malaysia
Tel.: 603 - 2140 8143
Fax: 603 - 2715 0701

Abdul Azim Abdul Ghani
Faculty of Computer Science & IT
University Putra Malaysia
43400 UPM Serdang
Selangor Darul Ehsan, Malaysia
Tel.: 603 - 89466555

ABSTRACT

This paper investigates a non-trivial, multi-objective and multi-constraint routing optimisation problem for dynamic packet-switched networks. The research adopts the application of the ant colony optimisation process into routing and congestion control in telecommunication networks. This paper suggests the use of epochal updating in conjunction with modified incremental updating to update the routing table in each switching node. The resulting new approach is called the For/Backward approach. Three updating methods (Forward, Backword, For/Backword) are simulated over a packet-switched network, representing Malaysian Backbone Network, using different combinations of traffic and geographical traffic patterns. The simulation results show a clear improvement on network performance (less average packet delay and greater throughput) using the For/Backward approach compared to the Forward and Backward methods. Furthermore, the behavior of the Forward and Backward methods is studied for inconsistency of behavior.

Keywords: Adaptive routing, Optimisation, ACO, Packet-switched networks

1.0 INTRODUCTION

Routing in dynamic networks is a multi-objective, multi-constraint optimisation problem. Although routing techniques designed for older technologies have sometimes proven adequate for newer techniques, novel routing approaches are often required [1]. Current routing methods use a single dynamic metric, static multi-metric or a combination of both for finding the shortest paths among the nodes in the network. Instead, a dynamic, distributed and multi-metric routing method is required.

The natural optimisation process named after the ants called the Ants Colony Optimisation (ACO) process [2] translates the natural process obtained from the study of real ants to an optimisation process that can be applied to solve many continuous and combinatorial problems. One of these problems is the famous Travelling Salesman Problem solved by applying a system called the Ants System derived from the ant optimisation process [3]. This system is a generalised and problem dependent system.

Any network can be modelled by a directed graph with N nodes. The nodes in the directed graph are switching nodes of type store and forward with a shared buffer space. A node is recognised by an identifier which is a serial integer number such as 1, 2, etc. A routing packet with a simple data structure presents an ant and the trail lying of pheromone by ants is mapped to some small influence on the routing tables held by the switching nodes in the network. The size of this influence should be a function of the quality of the route. Updating of the routing table was achieved by the following approaches:

- Epochal update [4]: requiring each agent to record its route and retrace its route to carry out the updates.
- Incremental update [5, 6]: the agent can update the routing tables incrementally as it travels its route.

1.1 Epochal (Backward) Update

The process is described in [4] and works in this manner; at regular time intervals a forwarding agent (ant or routing packet) $F_{s\to d}$ is launched from an arbitrary switching source node s in the network to a randomly chosen destination d. It should be noted that $F_{s\to d}$ has the same priority as any other data packet in the network to sense the congestion and other states of the network directly affecting the choice of the route. The agent $F_{s\to d}$ carries a dictionary

structure $D_{s\to d}$. The identifier of every visited node k and the time elapsed since the agent's launching time to arrive at node k, are pushed onto a stack $S_{s\to d}$ and inserted in $D_{s\to d}$ (see Fig. 1).

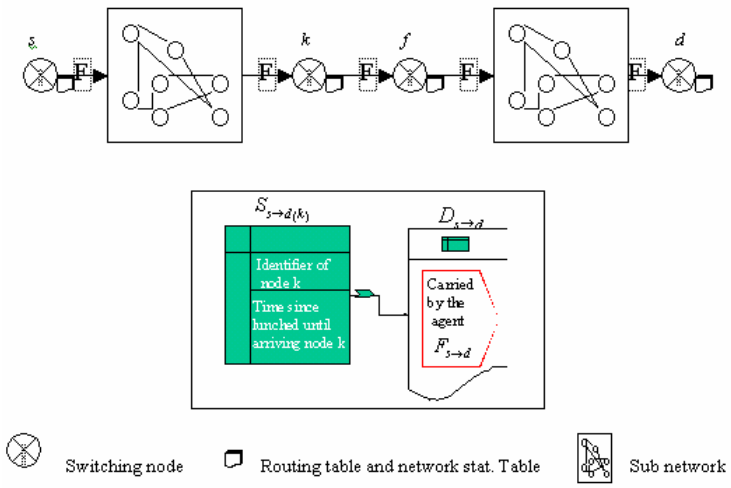

Fig. 1: ACO agent's route selection

At node *k*, the next node to be visited by the agent $F_{s\to d}$ is then selected following a random scheme, proportionally to the "goodness" of each neighbor's node. This is done by a look up operation into the routing table of node k. If a cycle is detected, i.e., the agent visited a node twice; the cycle's nodes are popped from the agent's stack. When finally, the destination node is reached, a backward agent $B_{d\to s}$ is generated and all the information carried by $F_{s\to d}$ is transferred to it. The backward agent will then trace the path back to the source node *s* by popping its stack $S_{s\to d}(k)$ to know the next hop node. On its way back to the source node s, the backward agent will update the entries of the statistical and routing tables at each visited node *k*. The trip time experienced by the forward agent to reach the destination node *d* starting from node *k* is used to modify the statistical table and the routing table at node *k*. If *f* was the next node chosen for the forward agent $F_{s\to d}$ when it was at node *k*, then the backward agent at node *k* will increment the entry (*d, f*) in the routing table and decrement the rest of the entries in row *d*.

It should be noted that $F_{s\to d}$ has the same priority as any other data packet in the network, while $B_{d\to s}$ has a higher priority to swiftly update the switching nodes in the network to complete the cycle of the epochal updating strategy. By using this method and launching agents with the same priority as those of data packets, congestion and other states of the network directly affect the choice of the route.

1.2 Incremental (Forward) Update

As in the Epochal method at regular time intervals a forwarding agent (ant or routing packet) $F_{s\to d}$ is launched from an arbitrary switching source node s in the network to a randomly chosen destination d. The agent $F_{s\to d}$ updates the probability entries corresponding to the source node s in the routing table of a visited node k. The entries in the routing table are updated by increasing the probability corresponding to the node from which the agent has just arrived, according to the formula:

$$P_{new} = \frac{P_{old} + \Delta P}{1 + \Delta P} \tag{1}$$

where P_{new} is the new probability and ΔP is the probability increase given by a very small probability or estimated using the mean and variance values for trip times. To normalise the probabilities in the routing table corresponding to a particular node, the other entries in the table of this node are decreased according to the formula:

$$P_{new} = \frac{P_{old}}{1 + \Delta P} \tag{2}$$

The next node to be visited by the agent $F_{s\to d}$ is then selected in the same way as described in the epochal approach.

1.3 Drawbacks of the Forward and Backward Methods

Adopting an epochal updating strategy, an agent cannot possess information about the quality of its route until it has finished it, resulting in updates of the early stages of the route happening only after a relatively long delay, which is a drawback of using this strategy. Di Caro and Dorigo [4] used this strategy and also tried incremental updating, but the results were significantly worse than their epochal updating strategy. Schoonderwoerd et al. [6] adopted an incremental updating strategy, however, this time they realised that agents do have good recent information about route quality at every instant, but the route about which they have this information is the route from their source node to their current position. Because the network links in this model are bi-directional, this information also applies to the route from the agent's current position back to the source node [1]. Therefore, this information at every node encountered by an agent is used to update incrementally the routing table entry referring to the source node rather than the destination node.

2.0 THE FOR/BACKWARD AGENT UPDATING METHOD

This research paper suggests the use of epochal updating in conjunction with modified incremental updating to update the routing tables in each node in order to overcome some of the disadvantages of using each separately [7]. The resulting new approach is called the For/Backward approach and is described in Fig. 2.

The For/Backward approach works in this manner, at regular time intervals a forwarding agent (ant or routing packet) $F_{s\to d}$ is launched from an arbitrary switching source node *s* in the network to a randomly chosen destination *d*. The agent $F_{s\to d}$ carries a dictionary structure $D_{s\to d}$. The identifier of every visited node *k* and the time elapsed since the agent's launching time until arriving at node *k*, are pushed onto a stack $S_{s\to d}$ and inserted in $D_{s\to d}$ as a part of the epochal updating strategy. The agent $F_{s\to d}$ updates the probability entries corresponding to the source node *s* in the routing table of a visited node *k*. The entries in the routing table are updated by increasing the probability corresponding to the node from which the agent has just arrived from (assume node *g*). The same modification is done to the probability entries corresponding to all nodes *i* such that *i* is a visited node by the forward agent before it reaches the node *k*. The probabilities of the rows (*i*, *g*) are incremented while the probabilities of other entries at rows with index *i* are decremented. In the same manner, the backward agent updates the entries of the rows (*i, f*) where *i* is previous node by the forward agent and positioned between node *k* and node *d*, while *f* is the next node chosen by *k* to forward $F_{s\to d}$ to *d*. The computation of the quantity r' used to update the routing tables is shown in the Appendix.

3.0 NETWORK TOPOLOGY AND SIMULATION SETTINGS

A packet-switched network representing the Malaysia backbone network is considered (see Fig. 3). The network consists of (13) nodes of type store and forward (switches) and a gigabyte buffer space (see Fig. 4).

The links connecting the nodes are bi-directional links with 1.5 Mbit/sec bandwidth. The propagation delays of these links are given in milliseconds (see Table 1). The assigned link or node fault probability is null and the links are accessed by statistical multiplexing.

The measures of interest in this simulation are the throughput (bits/sec) and average packet delay (sec). The generation interval of agents is set to 1 second, and start and end points are sampled uniformly over the network. The agent processing time is set to 2 milliseconds. The length of the simulation time is set to 120 seconds. The first 100 seconds are to the routing methods to learn routing and modify the routing tables. Initially, all links connecting a particular node have equal routing probabilities.

For all nodes in the network $s \in N = \{1,2,\ldots,N_{\max}\}$ {
    select a randomly chosen destination node *d*
    create a forward agent $F_{s \to d}$
    $k = s$
    do {
        remove cycle if encountered
        retrieve next node in the route *k* to *s* = node identifier at the top of $S_{F_{k \to d}}$ $NextNodeR_{k \to s}$
        $g = NextNodeR_{k \to s}$
        push *k* and $t_{s \to k}$ into the stack $S_{F_{k \to d}}$
        do{
            $h = NextNodeR_{h \to s}$
            calculate $t_{k \to h}$
            calculate the quantity Δr (the goodness of choosing g by *k* as $NextNodeR_{k \to h}$)
            use Δr to update the entries in row *h* in the routing table of node *k*
            update the statistical table at node *k*
            retrieve next node identifier following node *h* in the route *k* to *s* = next top node identifier in $S_{F_{h \to d}}$ $NextNodeR_{h \to s}$
        } while ($h \neq s$)
        if (explore is true) {
            from the nodes linked with node *k*, select a randomly chosen node *f* as the next hop
        }
        else {
            look up node *k*'s routing table for the node *f* with highest routing probability in row *d*
        }
        send $F_{s \to d}$ to f
        $k = f$
    }while($f \neq d$)
    create backward agent $B_{d \to s}$ and copy $S_{F_{s \to d}}$ to it
    do {
        next node in the route *s* to *d* = node identifier at the top of $S_{F_{s \to d}}$ ($f = NextNodeR_{s \to d}$)
        pop from $S_{F_{s \to d}}$ the node's identifier *i* and $t_{s \to i}$
        calculate $t_{i \to d}$
        calculate the quantity Δr (the goodness of choosing *f* by *i* as $NextNodeR_{s \to d}$)
        use Δr to update the entries in row d in the routing table of node *i*
        update the statistical table at node *i*
    }while($k \neq s$)
}

Fig. 2: The For/Backward Updating Method

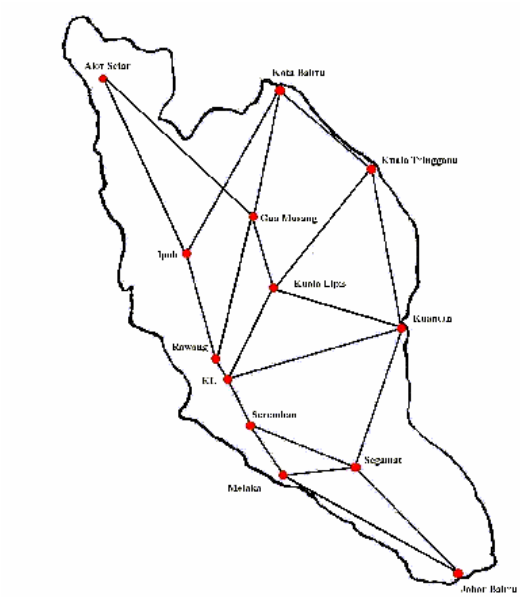

Fig. 3: The network topology showing nodes and their connections

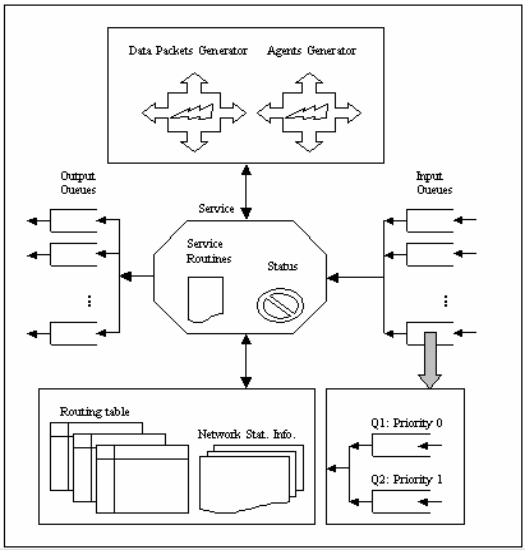

Fig. 4: A switching node structure

TABLE 1

LINKS PROPAGATION DELAYS

	1:Alor Setar	2:Ipoh	3:Rawang	4:KL	5:Seremban	6:Melaka	7:Segamat	8:Johor Bahru	9:Kuantan	10:Kuala Lipis	11:Gua Musang	12:Kuala	13:Kota Bahru
1	0	**21**	0	0	0	0	0	0	0	0	**26**	0	0
2	**21**	0	**15.5**	0	0	0	0	0	0	0	0	0	**21**
3	0	**15.5**	0	**2**	0	0	0	0	0	0	**15**	0	0
4	0	0	**2**	0	**5.5**	0	0	0	**20**	**13**	0	0	0
5	0	0	0	**5.5**	0	**7**	**10**	0	0	0	0	0	0
6	0	0	0	0	**7**	0	**7**	**19**	0	0	0	0	0
7	0	0	0	0	**10**	**7**	0	**16**	**15.5**	0	0	0	0
8	0	0	0	0	0	**19**	**16**	0	0	0	0	0	0
9	0	0	0	**20**	0	0	**15.5**	0	0	**14**	0	**17**	0
10	0	0	0	**13**	0	0	0	0	**14**	0	**8**	**20**	0
11	**26**	0	**15**	0	0	0	0	0	0	**8**	0	0	**14**
12	0	0	0	0	0	0	0	0	**17**	**20**	0	0	**13**
13	0	**21**	0	0	0	0	0	0	0	0	**14**	**13**	0

Two traffic patterns were considered: static and dynamic traffic patterns together with three geographical traffic patterns: uniform deterministic with one open session, uniform deterministic with five open sessions and random uniform. The input values needed by these patterns are assigned as follows:

- Static traffic pattern: the packets inter-arrival time is 150 milliseconds and their size distribution is a negative exponential with mean 512 bytes.
- Dynamic traffic pattern: session activation is a negative exponential with mean 15 seconds. The total number of packets per session, their sizes and their inter-arrival time are negative exponential with, respectively, mean values of 50, 512 bytes, and 10 milliseconds.
- Uniform deterministic geographical traffic patterns with one and five open sessions.
- Uniform random geographical traffic pattern with (800) open sessions and the start and end points chosen uniformly randomly.

4.0 RESULTS AND CONCLUSIONS

The simulation results of the Forward, Backward, and For/Backward methods are shown in Fig. 5 – Fig. 16. Four different network traffic systems are considered: static traffic with uniform deterministic geographical traffic pattern (ST-UDGT), static traffic with uniform random geographical traffic pattern (ST-URGT), dynamic traffic with uniform deterministic geographical traffic pattern (DT-UDGT), and dynamic traffic with uniform random geographical traffic pattern (DT-URGT).

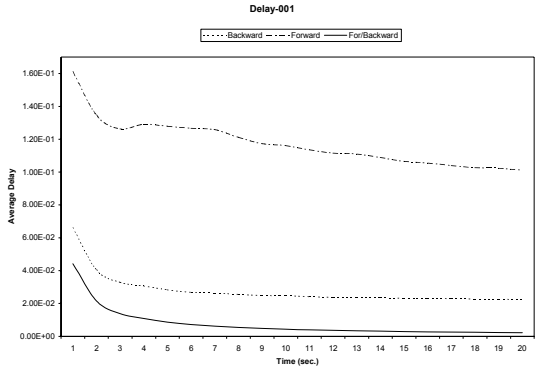

Fig. 5: Average packet delays (ST-UDGT, 1 OS, PIAT 150 ms, PS NE_512 bytes)

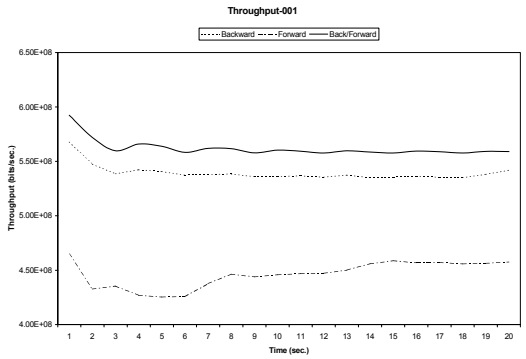

Fig. 6: Average Throughputs (ST-UDGT, 1 OS, PIAT 150 ms, PS NE_512 bytes)

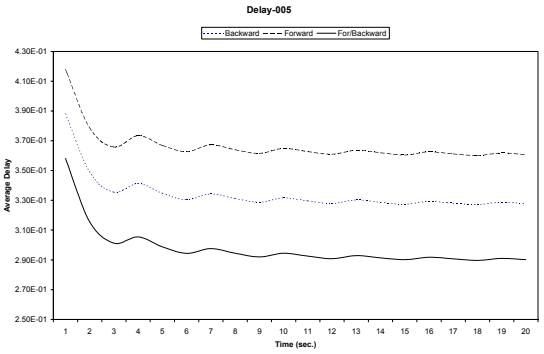

Fig. 7: Average packet delays (ST-UDGT, 5 OS, PIAT 150 ms, PS NE_512 bytes)

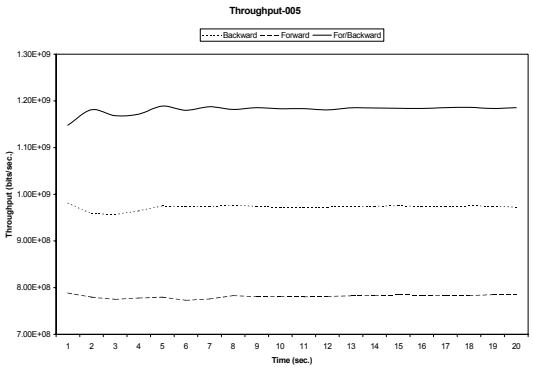

Fig. 8: Average throughputs (ST-UDGT, 5 OS, PIAT 150 ms, PS NE_512bytes)

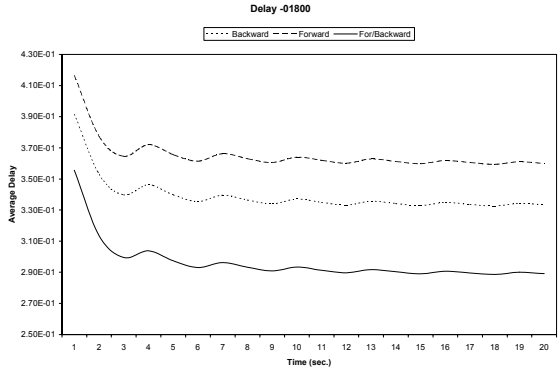

Fig. 9: Average packet delays (ST-URGT, 800 OS, PIAT 150 ms, PS NE_512 bytes)

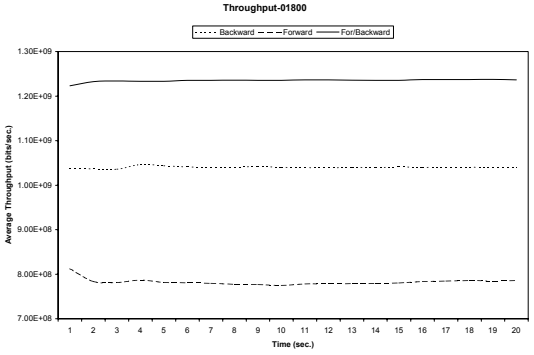

Fig. 10: Average throughputs (ST-URGT, 800 OS, PIAT 150 ms, PS NE_512 bytes)

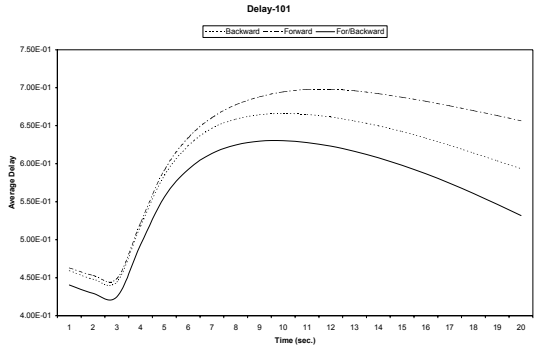

Fig. 11: Average packet delays (DT-UDGT, 1 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

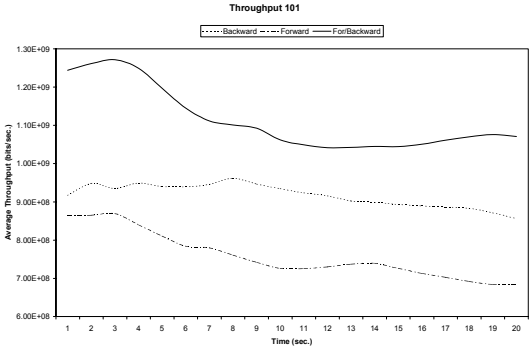

Fig. 12: Average throughputs (DT-UDGT, 1 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

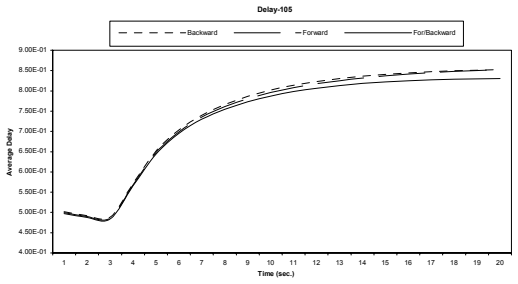

Fig. 13: Average packet delays (DT-UDGT, 5 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

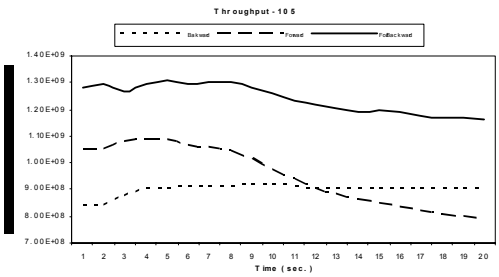

Fig. 14: Average throughputs (DT-UDGT, 5 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

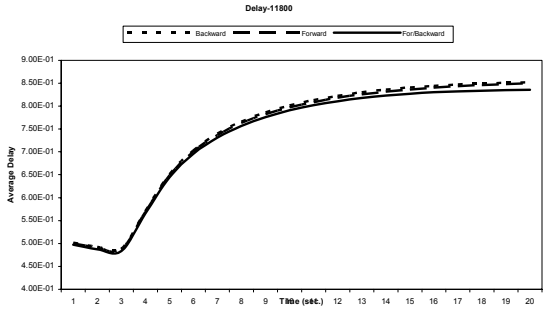

Fig. 15: Average packet delays (DT-URGT, 800 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

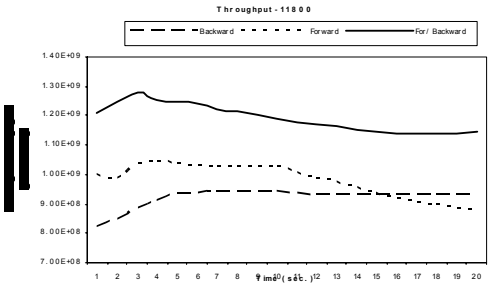

Fig. 16: Average throughputs (DT-URGT, 800 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

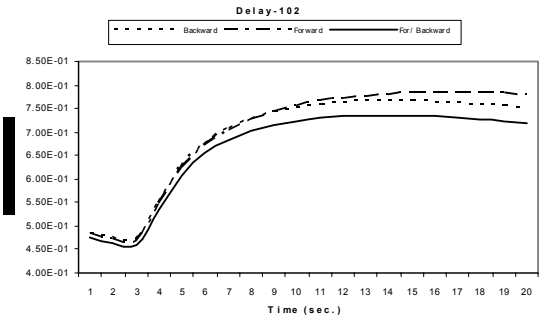

Fig. 17: Average packets delays (DT-UDGT, 2 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

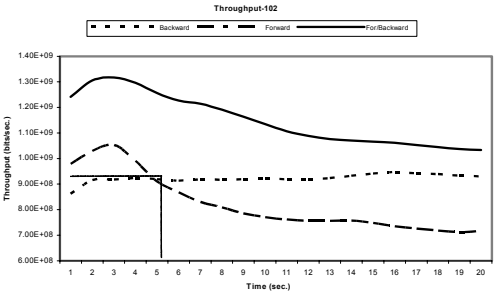

Fig. 18: Average throughputs (DT-UDGT, 2 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

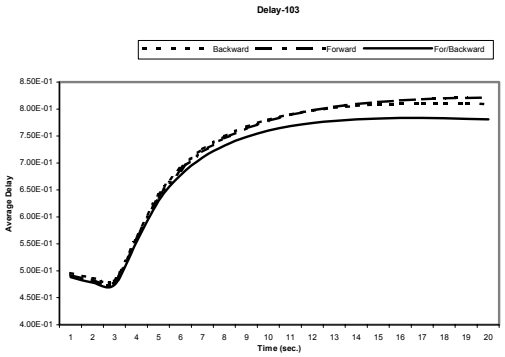

Fig. 19: Average packets delays (DT-UDGT, 3 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

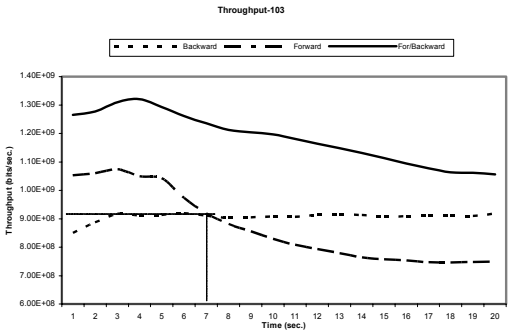

Fig. 20: Average throughputs (DT-UDGT, 3 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

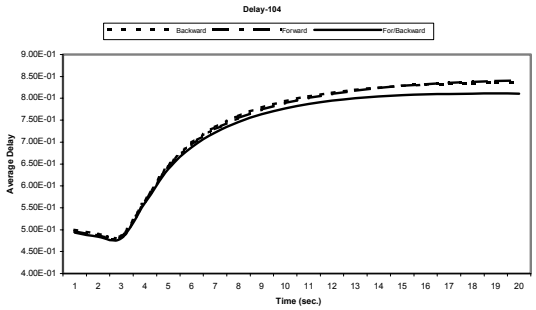

Fig. 21: Average packets delays (DT-UDGT, 4 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

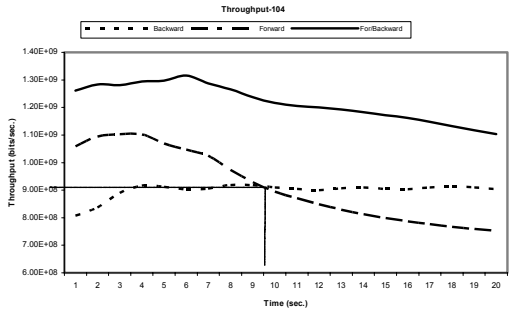

Fig. 22: Average throughputs (DT-UDGT, 4 OS, SA NE_15 sec, P/S NE_50, PS NE_512 bytes, PIAT NE_10 ms)

In these results the For/Backward method shows a clear improvement on the performance of the system compared with the Forward and Backward methods. The average packet delays of the For/Backward method are consistently less than those of the Forward and Backward methods, while the average throughputs of the For/Backward method are consistently greater than those of the Forward and Backward methods. On the other hand, the performance of the Forward method compared to that of the Backward method is not consistent and is based on the amount of traffic in the network.

The crossings in Fig. 14 and Fig. 16 are further studied by simulating DT-UDGT with 2, 3, and 4 OS. The results are shown in Fig. 17 – Fig. 22. The intersections are progressive in time and proportional to the number of open sessions. In DT-UDGT systems with large number of OS, i.e. heavier traffic presented in the system, the Forward method is expected to have a better performance than the Backward method. Inconsistency of throughput superiority exists between the Forward and Backward methods in DT-UGDT system. Similar results were obtained for Fig. 16.

APPENDIX

To fairly compare the simulation results obtained using the three routing methods (Backward, Forward, and For/Backward) consistency in updating the entries in the routing table must be maintained. Di Caro and Dorigo [4] describe a method to obtain a quantity r' that is used to determine the increment and decrement values necessary to update the entries of the routing tables held by the nodes in the network. The entries in the row with index equals to the identifier of the source node of the forward agent are updated in the routing table held by the current node. The quantity r' is obtained as follows:

If T is the observed trip time and μ is its mean value, as stored in the statistical table, the computed raw quantity r' measures the goodness of T, with small values of r' corresponding to satisfactory trip times:

$$r' = \begin{cases} \dfrac{T}{c\mu} & c \geq 1 \text{ if } \dfrac{T}{c\mu} < 1 \\ 1 & \text{otherwise} \end{cases} \qquad (1)$$

r' shows how good is the currently observed trip time with respect to what has been on average observed until now. c is a scale factor that is set to 2. A correction strategy is applied to the goodness measure r' taking into account how reliable is the currently observed trip time with respect to variance in the so far sampled values. The observation in the mean are stable if $\sigma / \mu < \varepsilon, \varepsilon << 1$. In this case, a good trip time is decreased by subtracting a value

$$S(\sigma, \mu; a) = e^{-a\frac{\sigma}{\mu}} \qquad (2)$$

from the value of r' while a poor trip time is increased adding the same quantity. On the other hand, if the mean is not stable, the quantity

$$U(\sigma, \mu; a') = \frac{e^{-a'\frac{\sigma}{\mu}}}{e^{a'}}, \frac{\sigma}{\mu} \in [\varepsilon, 1] \qquad (3)$$

with $a' \leq a$, is added to a good r' value and subtracted from a poor one. The above correction strategy, for both cases of σ / μ values, can be summarized as:

$$r' \leftarrow r' + sign(t - r') sign(\frac{\sigma}{\mu} - \varepsilon) f(\sigma, \mu) \qquad (4)$$

with f being S or U according to the case. The value of r' is finally reported on a more compressed scale through a power law, $r' \leftarrow (r')^h$ and bounded in the interval [0,1]. The obtained value r' is used by the current node k to define an increment, r_+, for the node f, which the packet arrived from, and a decrement, r_-, for the other neighboring nodes n:

$$r_+ = (1 - r')(1 - P_{df}) \qquad (5)$$

$$r_- = -(1 - r')P_{dn}, \quad n \in N_k, \; n \neq f \qquad (6)$$

where, P_{df} and P_{dn} are the last probability values assigned to neighbors of node k for destination d. These probabilities are then modified by

$$P_{df} \leftarrow P_{df} + r_+, \quad P_{dn} \leftarrow P_{dn} + r_- \qquad (7)$$

The constants used in the previous equations have been set to the following values by simulation tuning:
$c = 2$, $a = 10$, $a' = 9$, $\varepsilon = 0.25$, $h = 0.04$, $t = 0.5$.

ACKNOWLEDGMENTS

Many thanks to Dr Peter Blanchfield, the head of the Computer Science and Information Technology at the University of Nottingham Malaysia Campus, for his English editing of this article.

REFERENCES

[1] M. Steenstrup, *Routing in Communications Networks*. Englewood Cliffs, New Jersey: Prentice-Hall, 1995.

[2] M. Dorigo, V. Maniezzo and A. Colorni, "The Ant System: Optimisation by a Colony of Cooperating Agents". *IEEE* Transactions *on Systems, Man, and Cybernetics-Part B,* Vol. 26 No. 1, 1996, pp. 1-13.

[3] M. Dorigo and L. M. Gambardella, "Ant Colony System a Cooperative Approach to the Travelling Salesman Problem". *IEEE Transactions on Evolutionary Computation*, Vol. 1 No. 1, 1997, pp. 53-66.

[4] G. Di Caro. and M. Dorigo, "Mobile Agents for Adaptive Routing", in *Proceedings of the 31st Hawaii International Conference on System, IEEE Computer Society Press*, Los Alamitos, CA, 1998, pp. 47-83.

[5] E. Bonabeau, F. Henaux, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz, "Routing in Telecommunication Networks with 'Smart' Ant-Like Agents". *Technical Report*, no. 98-01-003, Santa Fe Institute, 1998.

[6] R. Schoonderwoerd, O. Holland and L. Rothkrantz, "Ant-Based Load Balancing in Telecommunication Networks". *Adaptive Behavior* Vol. 5 No. 2, 1997, pp. 169-207.

[7] M. Saleh, A. A. Ghani, M. Y. Saman and A. K. Ramani, "A Genetic Optimisation Model for Network Balancing and Adaptive Routing in Telecommunication Networks", in *Proceedings of the 8th International Parallel Computing Workshop*, National University of Singapore, 1998, pp. 289–292.

BIOGRAPHY

Mohammad Saleh holds PhD and MSc. (Computer Networks), BSc. (Mathematics/Computer Science). He is an Assoc. Professor in the Division of Computer Science and Information Technology at University of Nottingham (Malaysia Campus). His research interest is in the area of Network Performance Evaluation.

Abdul Azim Abdul Ghani holds PhD and MSc. (Software Engineering), BSc. (Computer Science). He is an Assoc. Professor and the Dean of the Faculty of Computer Science and Information Technology, University Putra Malaysia. His research interest is in the area of Software Engineering and in particular OO Program Slicing, OO Metrics, and Web Software Architectures.