COMPUTER NETWORK LOAD-BALANCING AND ROUTING BY ANT COLONY OPTIMIZATION

Ying-Tung Hsiao, Cheng-Long Chuang, and Cheng-Chih Chien

Department of Electrical Engineering
Tamkang University, Taipei, Taiwan, 251, R.O.C.
Tel: +886-2-26215656 Ext. 2786, Fax: +886-2-26209814
E-mail: clchuang@ee.tku.edu.tw

Abstract—**A high efficient design of computer network is an important issue for the high transmission speed requirement of today. In computer network, the data packages have to be transmitted to the destination with a minimum delay for ensuring the quality of service guarantees. This work presents an algorithm to perform a dynamic load-balancing for transmitting the data packages with near minimum delays in the interconnection networks. The proposed algorithm is based on the ant colony optimization algorithm inspired by the simple behavior of biological ants. This work utilizes a cube topology network to evaluate the performance of the proposed algorithm. From the comparing results, the proposed algorithm can achieve good network utilization by the low rate of the bandwidth blocking.**

I. INTRODUCTION

Typically, the links, routers, and servers are the main cause of the delay for transmitting data in a computer network. As a network is growing bigger and complex, it is a real challenge to deal with the problem of load-balancing and routing. The needed of intelligence methods to solve these problems is necessary important. A shortest link path generated by a routing table is unable to meet the requirements regarding the delay problem when the transmission rate of the critical link is exceeded its maximum capacity. To reduce the transmission delay, the data packages have to avoid to be routed to a heavily loading link to save transmission time.

Minimize the transmission delays is necessary for minimizing the congestion. A good routing technique is also important to increase the system throughputs. When two or more data packages are routed to be transmitting by the same link at the same moment, a hot-spot could be produced. A hot-spot is a region that the network link capacity is saturated, and requires more bandwidth. The packages entered this region takes a very high transmission time. While the useful bandwidth resources of other regions are available for a package and it can be routed to other link by utilizing the available bandwidth resources, the package may be able to transmit to its destination node quickly. Therefore, the problem of the load-balancing and routing can be defined as that there is an disproportionately communication load distribution over the topology of the network, even the total network bandwidth requirement do not exceed the total limit of the network, and the disproportionately communication load distribution generates some nodes that the processing resources are exhausted as if the whole interconnection network were collapsed. This problem is produced when the buffers of the routers in the hot-spot region are exhausted while other routers still have free resources. The hot-spot problems may exist in many region of the network, and the throughput of the network is very worst because the blocked data package is occupying a large number of links resources in the network.

In this work, we formulate the problem of load-balancing and routing in computer network as an optimization problem. The problem consists of one performance index, i.e. the delay time. Typically, we employ average delay time to determine the performance of a routing protocol. In this study, the primary goal is to design a zero-administration distributed routing protocol that is capable to minimize the transmission delay time of data packages and to avoid generating hot-spot region over the topology of the network. Furthermore, we apply an ant colony optimization [1-5] based algorithm to achieve the design goal. ACO is a general-purpose optimization technique that has been recently developed and recognized as effective for combinatorial optimization problems [5] such as traveling salesman problem [6], quadratic assignment problem [7], graph coloring problems [8], and hydroelectric generation scheduling problems [9]. This paper focuses on ACO approaches in computer network load-balancing and routing. The feature of the presenting technique is that can be implemented easily and flexible for many different problem formulations. The ACO algorithm consists of many artificial ants, and these ants follow simple procedures that simulate the laying and sensing of pheromone to find the optimal solution of the problem. In the computer network routing problem, the objective of ACO algorithm is to find the fastest link for packages to be transmitted to their destination nodes.

0-7803-8783-X/04/$20.00 © 2004 IEEE

This paper is organized as follows. Section II briefly introduces the load-balancing and routing problems. Section III introduces the ACO technique. Section IV applies ACO for load-balancing and routing protocol. Section V discusses experimental results as illustrations. Finally, the conclusions are discussed in the last section.

II. Problem Formulation

The load-balancing and routing are the methods of creating alternative source-destination paths in the computer network. The load-balancing method distributes every data package load over several paths of the network. The goal of the load-balancing routing technique is to uniform the traffic loads of entire computer network, and maintain low data package transmission delay and avoid the generation of hot-spots. The uniformed traffic loading of the computer network ensures that the total bandwidth requirement does not exceed the total limit of the network capacity. By applying the load-balancing routing technique, the data packages are distributed to less loading paths instead of a heavily loaded path.

For a given network $N(\Gamma,\Lambda)$, where Γ is the set of nodes and Λ is the set of links, and let $M=\|\Gamma\|$ and $L=\|\Lambda\|$. Π denotes the set of all possible free paths, and $\Pi_{(s,d)}$ denotes the set of all possible paths between the source node s and the destination node d. Let π is a generic path, and $\pi_i(s,d)$ is the i-th path from source node s to destination node d. Let Π^l be the set of all possible paths sharing the same link l. $\Pi^l(s,d)$ is the set of all paths from source node s to destination node d that is sharing the same link l.

Let $\Phi(\pi)$ be a generic cost function associated to the path π, related to the network topology. $\Phi(\pi)$ can be a function of the delay time of π, or of the number of hops $n(\pi)$, or a function of both of them. For two paths $\pi_1(s,d)$ and $\pi_2(s,d)$, if $\Phi(\pi_1)=\Phi(\pi_2)$, then the two paths are considered as Φ-*equivalent*.

We also have to define the traffic matrix $T=[t_{s,d}]$, where $t_{s,d}$ is the indicator that the link between originated source node s and the routed destination node d has been used or not (in this case, node s and node d are directly connected neighborhood nodes). Let B_s is the offered load of node s. For the matrix $E=[e_{s,d}]$, where $e_{s,d}$ is the probability that a source node s tries to send a package toward a destination node d. We shall denote $\rho(s,d)$ as another cost function, derived from the bandwidth requirement of the connection originated at the source node s to the routed destination node d. According the definition, we have $\rho(s,d)=e_{s,d}B_s$, its representing the load that a node offered to the network.

With the above notation, we can compute the cost function $\Phi(\pi)$ by following formula:

$$\Phi(\pi)=\frac{1}{C_l}\sum_{s,d|\pi(s,d)\in\Pi^l}\rho(s,d) \qquad (1)$$

where C_l is the offered bandwidth of link l. In this work, we are targeting to route the data packages to the minimum cost function path. The detail route technique is introduced in Section IV.

III. Ant Colony Optimization Algorithm

In this section, the Ant Colony Optimization (ACO) method is introduced. ACO is a graph representation based evolutionary meta-heuristic algorithm. ACO has been successfully employed to solve many different combinatorial optimization problems. The main idea of ACO is to model the problem as a minimum cost path searching problem in a graph. ACO consists of many artificial ants, and these artificial ants walk through the graph to find the shortest path of the graph. An artificial ant has a very simple behavior, and all ants have no knowledge about which path is the shortest, so typically, it only be able to find a very poor-quality path by itself. Better paths are found by the global cooperation among all artificial ants in the colony.

The behavior of artificial ants is modeled from biological ants. In real world, biological ants act in a very simple behavior as well. When they walk through a place, they lay pheromone trails on the path. The moving direction of each ant is decided by the consistency of the pheromone that lies on the path. Ants prefer to go through a path with a relatively high among of pheromone on it. In addition, artificial ants have some extra feature that do not find in biological ants. In particular, they live in a discrete world. Their moves consist of transitions from nodes to nodes. Their moves usually associated with their pervious action that stored in the memory with a specific data structure.

The pheromone consistencies of all paths are updated only after the ant finished its trip from the original node to destination node and not during the ant is still walking. Every artificial ant has a constant amount of pheromone stored in it when the ant proceeds from the original node. When the ant has finished the trip to the destination node, the pheromone that stored in it will be distributed averagely on the path of its journey. By this pheromone distribution strategy, if an ant finished its journey with a good path, the average distribution amount of pheromone will be high. In the other hand, if an ant finished its journey with a poor path, then the average distribution amount of pheromone will be low. The amount of pheromone deposited is usually corresponding to the quality of the path. The pheromone of the routes progressively decreases the path. The pheromone of the routes progressively decreases by evaporation in order to prevent the artificial ants stick in local optima solution.

Fig. 1 depicts the ACO algorithm. At each generation, each ant generates a complete journey by choosing the nodes according to a probabilistic state transition rule. It is necessary to spread a little amount of pheromone on all paths at the initial of the algorithm. Every ant selects the nodes in the order in which they will appear in the permutation. For the selection of a node, ant uses heuristic factor as well as pheromone factor. The

procedure ACO
  Set parameters and Initialize pheromone trails
repeat
    for k=1 to m do construct an assignment X_k
    update pheromone trails using $\{X_1,\ldots,X_m\}$
  until best solution achieved
    or maximum cycle reached

Fig. 1. ACO algorithm

heuristic factor, denoted by $\eta_{(s,d)}$, and the pheromone factor, denoted by $\tau_{(s,d)}$, are indicators of how good it seems to have node d at node s of the permutation. The heuristic value is generated by some problem dependent heuristic whereas the pheromone factor stems from former ants that have found good solutions.

The rule for an ant to choose its next node is called Pseudo-Random-Proportional Action Choice Rule. With probability q_0, where $0 \le q_0 < 1$ is a parameter of the algorithm, the ant chooses a node d from the set Γ of nodes that have not been selected so far which maximizes

$$[\tau_{(s,d)}]^{\alpha}[\eta_{(s,d)}]^{\beta} \tag{2}$$

where $\alpha \ge 0$ and $\beta \ge 0$ are constants that determine the relative influence of the pheromone values and the heuristic values on the decision of the ant. With probability $(1-q_0)$ the next node is chosen from the set Γ according to the probability distribution that is determined by

$$p_{(s,d)} = \frac{[\tau_{(s,d)}]^{\alpha}[\eta_{(s,d)}]^{\beta}}{\sum_{h\in\Gamma}[\tau_{(s,h)}]^{\alpha}[\eta_{(s,h)}]^{\beta}} \tag{3}$$

Therefore the transition probability is a trade-off between the heuristic and pheromone factor. For the heuristic factor, the close nodes, that mean the paths with low cost, should be chosen with high probability, thus implementing greedy constructive heuristic. As to the pheromone factor, if on edge (s,d) there has been a lot of traffic then it is highly desirable, thus implementing the *autocatalytic process*. If the selection results a $P_{(s,d)}$, carry out the rule (3). In (3), the heuristic factor $\eta_{(s,d)}$ is computed according to the following rule

$$\eta_{(s,d)} = \frac{1}{f(X_d)} \tag{4}$$

where $f(X)$ represents the cost function of X. In (3), it is favor that the choice of edges which are shorter, which mean the path with low cost, and which have a greater amount of pheromone. Along the path from s to d the ant lays a trail substance so defined

$$\Delta\tau^{k}_{(s,d)} = \begin{cases} \dfrac{Q}{\Phi(\pi_k)} & \text{if } k\text{-th ant uses edge } (s,d) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where Q is a constant related to the quality of pheromone trail laid by ants and $\Phi(\pi_k)$ is the cost of the journey performed by the k-th ant. In other word, pheromone updating is intended to allocate a greater amount of pheromone with low cost, which means shorter journey. This value is evaluated when the ant has completed a journey and consisting of a cycle of n iterations (generations). Then, it is used to update the amount of substance previously laid on the trail, on the following rules

$$\tau_{(s,d)}(t+n) = \sigma\cdot\tau_{(s,d)}(t) + \Delta\tau_{(s,d)}(t) \tag{6}$$

$$\Delta\tau_{(s,d)}(t) = \sum_{k=1}^{m}\tau_{(s,d)}(t) \tag{7}$$

where m denotes the number of ants, σ, $\sigma\in(0,1)$, is a coefficient of persistence of trail during a cycle such that $(1-\sigma)$ represents the evaporation of trail between generation n_g and n_g+1. The pheromone-updating rule was meant to the addition of new pheromone deposited by ants on the visited edges and to pheromone deposited by ants on the stops iterating either when an ant found a solution or when a maximum number of generations have been performed.

IV. Apply ACO to Load-Balancing Routing

In this section, the detailed ACO routing procedure is discussed. The whole procedure to establish a network based on ACO routing protocol is as follow

1) Create routing information.
2) Update the pheromone level table for operate and adapt.

A. Create Routing Information

In traditional routing algorithms, each node depends on the routing information provided by its neighboring nodes to construct a complete routing table of itself. The neighboring nodes depend on the routing information of their neighboring nodes, which means that every node in turn depend on others routing information of other nodes.

In ACO, the routing table of each node dose not depends on other nodes to construct. The routing path from a source node to a destination node is found independently and in parallel. Ants travel independently from a source node to discover several paths and according to the cost function these paths; we can determine the shortest path and spread the pheromone on the paths. As soon as the ant arrives the destination node, the pheromone for its path is updated. Therefore, the pheromone level of each link of a node can be updated independently. When the system is operating, each node of the network can use the information of pheromone level table in each node to route

the data packages to the correctly link of the node. Simultaneously, when the data package reaches the destination node, the pheromone level table of each node is updated by the acknowledge package that contents the cost of the original request data package.

The routing information of the ACO algorithm is discovery by release many ants to tour the network topology. The following procedures present the method of constructing the routing information for each node in network.

Step 1. Initialization.
An initial population of ant colony individuals X_i, i=1, 2,…,m, is initialized in this step. The maximum node transport action number of each ant is *MaxT*. Format the *journey nodes lists* of each ant to provide the following step to record the ant journey history. Randomly generate a pheromone consistency value and set the value as initial condition for each route. Set time counter t=0 and generation counter n_g=0.

Step 2. Starting to tour.
For i=0 to M
For j=0 to M
For ant counter k=1 to m
Place the starting node of the k-th ant in source node i, and set the destination node as f.

Step 3. Searching neighborhood.
Repeat until the *journey nodes list* is full
For k=0 to m **do**
If ant X_k is already reached the desired destination node f
Then Break and sim next ant X_{k+1}.
Choose the node j to move to, with probability $P_{(s,d)}$ given in (3).
If node j already exists in the *journey nodes list*
Then Repeat last procedure.
If the ant X_k already stuck at a dead end or reach the maximum transport action number *MaxT*.
Then suspend ant X_k from simulation.
Move the k-th ant to the node j.
Insert node j into the *journey nodes list* of ant X_k.

Step 4. Calculate the path cost $\Phi(\pi)$ for each ant's journey.
For k=0 to m **do**
Compute the cost $\Phi(\pi_k)$ of the journey node list of ant X_k.
For every route (i, j)
For k=1 to m **do**
Calculate the pheromone trail of each edge according to (5).

Step 5. Update the pheromone level table.
For every edge (i,j) update the pheromone value according to the update rule (6) and ~~(7),~~
For every edge (i,j) update the pheromone value by multiply it by σ.
Let t=t+1; n_g=n_g+1
For every edge (i,j), reset $\Delta\tau_{(i,j)}$=0

Step 6. Check the stop criterion.
If (n_g<n_{gmax}) or (not user stop)
Then
Clear the *journey nodes list*, and **Goto** Step 2.
Else
Stop.

B. *Operate and Adapt*

After the routing information has been created, the pheromone level table of every node is updated and operational. During the network system operating, the load distribution for each region may be changed. Hence, the network has to adapt the loading condition at the time to prevent generating the congestion region.

An optimal path to transmit a package to its destination may be changed. This issue occurs may lead to:

1) Optimal path dramatically become non-optimal.
2) New congestion nodes or hot-spots generated.
3) Some node suddenly disconnected from the network.
4) Another optimal path has been found and replaced the former optimal path.

In order to make the network system able to adapt the loading condition changes and network topology changes, the following procedures present the method of updating the routing information for each node in network.

Proc 1. Package arrives the destination.
If data package reaches the destination node, calculate the cost $\Phi(\pi)$ for transmitting the package, create a acknowledge package that content the cost $\Phi(\pi)$ and send it back to the original source node.

Proc 2. Acknowledge package arrives the original source.
Extract the cost of original request package $\Phi(\pi)$, and calculate and update the pheromone level table of the original source node.

Proc 3. Pheromone evaporation.
For every edge (i,j) update the pheromone level value according to the update rule (6) and (7).

Proc 4. Select the link to route the package.
For every link of the node
Calculate the sum of all pheromone level of each link
Choose the link to route the data package by probability $P_{(s,d)}$ given in (3).
If link does not exist or route fails
Then delete the routing information of the link and repeat the last for loop.

V. Simulation Results

Simulations were performed to compare the difference between the ACO-based routing algorithm and the round-robin routing algorithm. The simulations topologies were shown in fig. 2 and fig. 3, which are CAIRN topology and cube topology, respectively. The cube topology is a network that its internal interconnected nodes and edges correspond to the vertices and edges of a three-dimensional cube. The CAIRN topology is a real network, more detailed information could be found on the web (http://www.cairn.net).

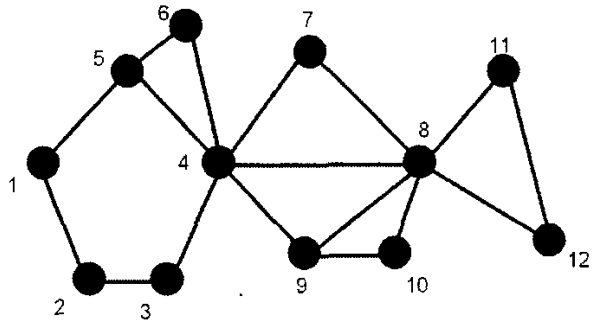

Fig. 2. CAIRN Topology

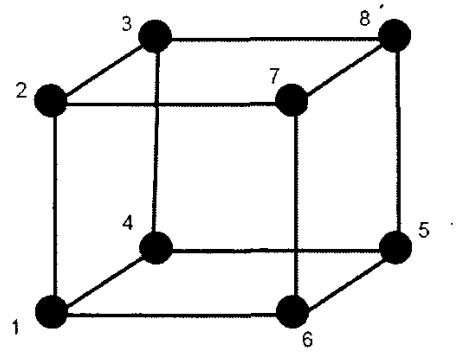

Fig. 3. Cube Topology

Servers of a network can be assigned on each node of the network, or we can specially designate some nodes as servers. The servers request for service and send data packages to every node of the network. When the destination node of a data package is achieved, the destination node generates a acknowledge package and sends it back to the source node that is the original service requesting server of the original data package. Note that the algorithm also consider the transmission cost of the acknowledge package to refine the parameters and tables of the network system.

For simplicity and easily comparison, we used a stable topology in all simulations, which means that the connections or internal nodes will not fail during the simulation process. Some parameters and constraints are applied in the simulations as follow:

A. Offered Link Capacity

The offered link capacity is the maximum data transmission rate of a link, modeled as an M/M/1 queue. All links have been assigned the same maximum capacity. In this paper, we set all maximum link capacity as10Mb/s.

B. Propagation Delay Time

The propagation delay time is a fixed delay added between the transmission and reception of a package. The propagation delay time is set as 10μs.

C. Average Request Rate

The average request rate is the rate at which packages enter the network system. In this paper, all package length is assumed to be 1250 bytes.

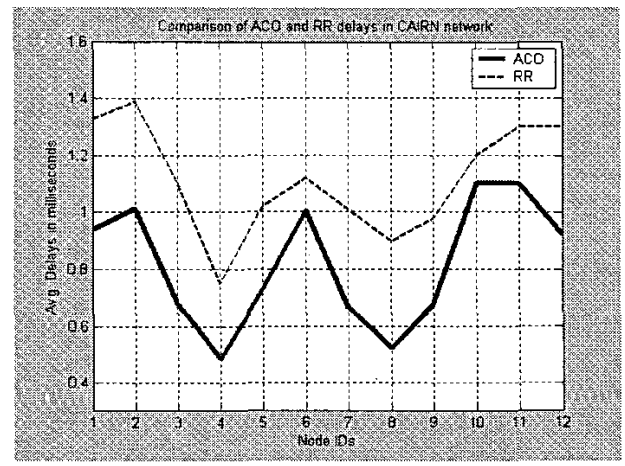

Fig. 4. CAIRN network topology. (Average delays for requests at each node. Servers are assigned at every internal node.)

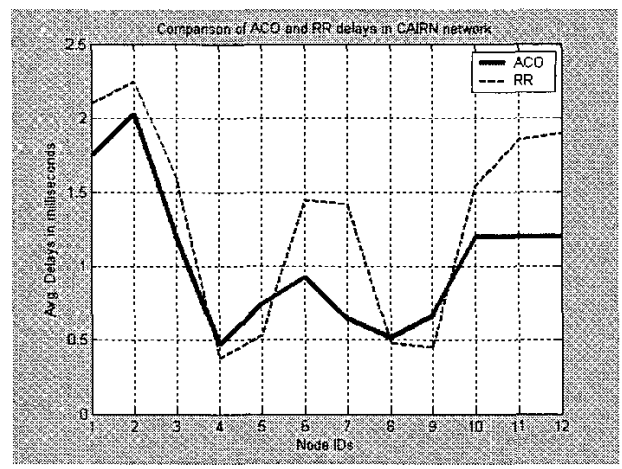

Fig. 5. CAIRN network topology. (Average delays for requests at each node. Servers are assigned at nodes 4, 5, 8 and 9.)

D. Average Processing Delay Time

The average processing delay time is the time requirement for every server to process a single received package. The servers are modeled as an M/M/1 queue as well. The average processing time is defined as 0.1ms in this paper.

The request rate is maintained to be steady at the server nodes. In the result graphs, the plots for the round-robin scheme are labeled "RR" and the plots for ACO-based algorithm are labeled "ACO". The X-axis is labeled with the node ID. The Y-axis is labeled with the average delay time of each node.

Fig. 4 and fig. 5 show the results for CAIRN topology. Fig. 6 and Fig. 7 are the simulation results for Cube topology. In these cases, there is a server at every node and the same level of input traffic at these nodes. We can see that the load-balancing algorithm diverting traffic to nearby nodes, delay time of some nodes may increased, but it is indeed successfully improved the entire system performance. The ACO-based load-balancing routing algorithm shows lower delays and outperforms than the round-robin scheme.

VI. Conclusion

The load-balancing routing is a technique to minimize both of the delay time of routing and hot-spot generation. In this work, we presented and simulated a novel approach to perform load-balancing routing algorithm based on the ACO. ACO-based load-balancing routing algorithm supports load balancing in computer networks in which all computations are distributed on various nodes, using the delay time of each transmitted data package, we can calculate the cost of the package transmission and refine the pheromone level table of each node. By the persist updating of the pheromone, the presented method can get a well total system performance and prevent the generation of hot-spots. Finally, we compare the proposed routing algorithm with the round-robin routing scheme, which is a protocol for load-balancing of communication links. By simulation results, we can see that the ACO-based algorithm can outperform the comparison scheme, and provide a better performance and more reliable than the round-robin routing algorithm. This may be the first step to apply the ACO in the computer network routing problem, and it's a practical method for implementation and research for discovering more potentials of the ACO technique.

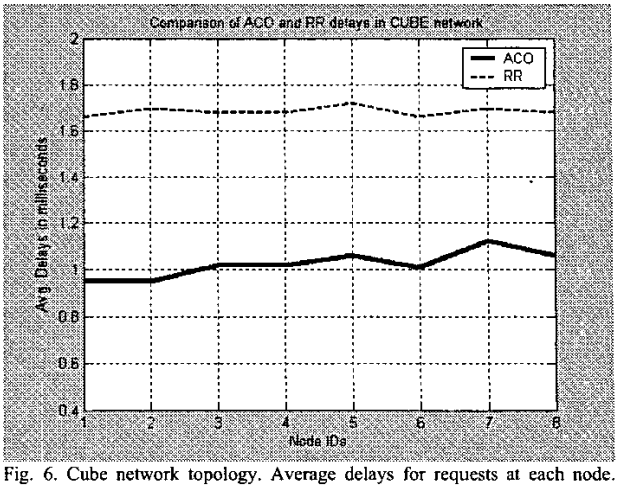

Fig. 6. Cube network topology. Average delays for requests at each node. Servers are assigned at every internal node.

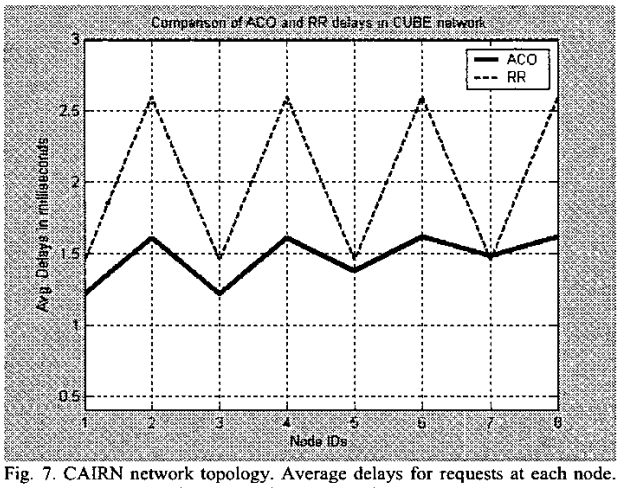

Fig. 7. CAIRN network topology. Average delays for requests at each node. Servers are assigned at nodes 1, 3, 5 and 7.

References

[1] M. Dorigo, G. D. Caro, and L. M. Theraulaz, "Inspiration for optimization from social insect behavior," *Nature*, vol. 406, pp. 39-42, July 2000.

[2] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy," *Future Gener. Comput. Syst.*, vol. 16, no. 8, pp. 851-871, 2000.

[3] V. Maniezzo and A. Carbonaro, "Ant colony optimization: An overview," *in Metaheuristic Int. Conf.*, Angra dos Reis, Brazil.

[4] M. Dorigo and T. Stutzle, "The ant colony optimization metaheuristic: Algorithms, applications, and advances," *in Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Norwell, MA: Kluwer.

[5] M. Dorigo, G. D. Caro, The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, F. Glover (Eds.), *New Ideals in Oprimization*, McGraw-Hill, 1999.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents", *IEEE trans. on System, Man, and Cybernetics-Part B: Cybernetics*, vol. 26, no. 1, pp. 29-41, Feb. 1993.

[7] D. Costa and A. Hertz, "Ants can color graph," *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 295-305, Mar. 1997.

[8] M. Dorigo, T. S. Zel, *Ant colony optimization*. Bradford, 2004.

[9] R. S. Parpinelli, H. S. Lopes, A. A. Freitas, "Data mining with an ant colony optimization algorithm," *IEEE Trans. Evolutionary Computation*, vol. 6, no. 4, pp. 321-332,Aug. 2002.

[10] S. Vutukury and J. J. Gurcia-Luna-Aceves, "A simple approximation to minimum-delay routing," *Proc. ACM SIGCOMM '99*, Cambridge, Massachusetts, Sept. 1999.

[11] A. Orda and R. Rom, "Routing with packet duplication and elimination in computer networks," *IEEE trans. Commmun.*, vol. 36, pp. 860-866, July 1988.

[12] W. T. Zaumen, S. Vutukury, and J.J. Garcia-Luna-Aceves, "Load-balanced anycast routing in computer networks," *in proceedings of Fifth IEEE Symposium on Computers and Communications*, pp. 566-574 , 3-6 July 2000

[13] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *IEEE trans. on System, Man, and Cybernetics-Part A: Systems and Humans*, vol. 33, No. 5, Sept. 2003.