ACO-Based Cascaded Adaptive Routing for Traffic Balancing in NoC Systems

En-Jui Chang, Chih-Hao Chao, Kai-Yuan Jheng, Hsien-Kai Hsin, and An-Yeu Wu
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan
Email: enjui@cc.ee.ntu.edu.tw

***Abstract*—Ant Colony Optimization (ACO) is a bio-inspired algorithm extensively applied in optimization problems. The performance of Network-on-Chip (NoC) is generally dominated by traffic distribution and routing. With more precise network information for path selection by using pheromone, ACO-based adaptive routing has higher potential to overcome the unbalance and unpredictable traffic load. On the other hand, the implementation cost of ACO is in general too high to store network information in pheromone memory, which is a routing table of all destination-channel pairs. We propose an ACO-based Cascaded Adaptive Routing (ACO-CAR) by combining two features: 1) table reforming by eliminating redundant information of far destinations from full routing table, and 2) adaptive searching of cascaded point for more precise network information. Our experimental results show that ACO-CAR has lower latency and higher saturation throughput, and can be implemented with 19.05% memory of full routing table.**

Keywords- Network-on-Chip (NoC); Ant Colony Optimization (ACO); Adaptive Routing; Selection Function; Traffic Balancing;

I. Introduction

With the advances of the semiconductor technology, the increasing complexity and delay of interconnection dominate the performance of System-on-Chip (SoC). To provide more efficient interconnections and accommodate data transfer requirements, Network-on-Chip (NoC) has been proposed as a flexible, scalable and reusable solution [1][2]. However, the performance of NoC is dominated by traffic distribution [3]. As the size of the system scales up, the traffic load becomes an unbalance and unpredictable distribution with the diverse applications. Hence, the traffic is prone to be congested due to that the packets are blocked and continue to hold all the channels. The increased latency brings about the problem of performance degradation in real-time application. Therefore, effective routing is an essential approach to deal with traffic congestion.

Routing algorithm determines a path that each packet is transferred from a source to a destination in NoC systems. Adaptive routing is composed of *routing function* and *selection function.* Firstly, the routing function returns a set of output channels based on the turn models [4]. Then, the selection function chooses an output channel of the candidate channels from the routing function according to the different network information. Obtaining more detailed network information can improve the selection efficiency and the balance of traffic distribution. Therefore, the selection function affects the performance critically for all adaptive routing algorithms [5].

This work is supported in part by the National Science Council, Taiwan R.O.C., under Grant NSC 97-2221-E-002-239-MY3.

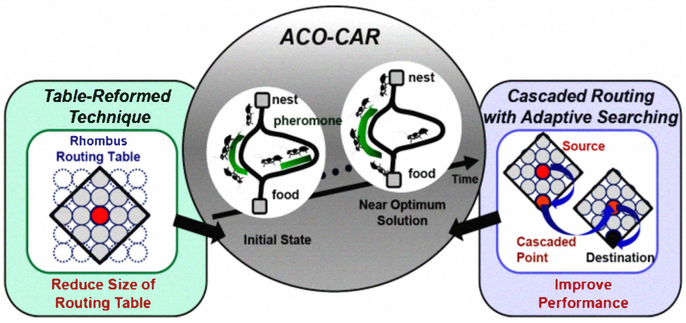

Figure 1. Scheme of ACO-CAR.

In order to make traffic more balance, *Ant Colony Optimization (ACO)-based adaptive routing* is proposed in NoC systems [6]. ACO is a bio-inspired algorithm that imitates the process of an ant colony that finds the shortest path from their nest to a food. Ants diffuse the pheromone on the path to communicate with each other, and the pheromone is accumulated faster on the shorter path. In the long run, ants can reach to the food with the shortest path. ACO is extensively applied in optimization problems [7][8]. ACO can also enhance selection efficiency with the current and the historical local buffer state for traffic balancing.

The implementation cost of ACO-based adaptive routing, however, is too high to store the network information in pheromone memory, which is a table of all destination-channel pairs. With the increased size of NoC scale, it becomes infeasible since the memory cost and the table access time grows significantly. Besides, the power consumption also grows with the table size. The challenge of this paper is to implement ACO-based adaptive routing with less memory and to balance the traffic load.

In this paper, we propose an *ACO-based cascaded adaptive routing (ACO-CAR)* for cost reduction and traffic balancing is shown as Fig. 1. The contributions are listed as following:

- *Table-reformed technique* diminishes the size of routing tables and ensures the reliability of network information.
- *Cascaded routing with adaptive searching* of cascaded point for more precise network information to balance traffic.

From our experiments, the cost of routing table can be reduced by 80.95% with proposed table-reformed technique. Moreover, ACO-CAR has higher saturation throughput, with an improvement from 16.67% to 40% compared to other selection functions. It can spread the traffic load and make network load more balanced.

II. Review of Adaptive routing

In this section, first we review the adaptive routing. Then we introduce the ACO-based adaptive routing [6][7] and the relation between the traffic balancing and implementation cost. Moreover, the problem of cost of ACO-based adaptive routing is formulated as a mathematical programming model.

A. Adaptive Routing

In adaptive routing, the routing function provides the path diversity for the selection function. A good selection function helps to spread the traffic and make network load more balanced according to the detailed network information. The network information can be classified as spatial and temporal information. Spatial network information is the buffer state of the different directions. Local information and regional information are the output buffer length of local router and neighbor routers respectively. Temporal information consists of current and historical information. Selection function bases on the present condition or the past experience to choose a path. Therefore, different selection functions make use of the different combinations of network information.

Adaptive routings can be classified as congestion-oblivious and congestion-aware according to whether selection functions consider the output link demand or not [9]. The congestion-oblivious adaptive routing does not take account of the output link status. *Random selection function* randomly chooses an output channel from the candidate channels [10], which is simple to implement, but it can not balance traffic load. The congestion-aware adaptive routing [9] takes account of the output link status. Therefore, these selection functions can adjust path selection with the time-variant congestion status. *Output buffer length (OBL) selection function* chooses an output channel with the shortest occupied buffer length [11]. OBL makes a routing decision with the current local network information. *Neighbors-on-Path (NoP) selection function* chooses an output channel that has the shortest occupied buffer length of neighbor and local router on routing path [11]. That is making a routing decision with the current regional network information. Using only current spatial network information in making routing decisions is efficient for local traffic balance, but it cannot guarantee global balance [14].

B. ACO-based Adaptive Routing

ACO-based adaptive routing can be defined as an adaptive routing with *ACO-based selection function* is shown as Fig. 2. It has two types of packet: ant packet and data packet. ACO-based routing uses ant packet for training and data packet for normal payload. The difference between the ant packet and data packet is the head flit. The head flit of ant packet consists of ant index and the information for routing. Besides, ant packets also carry the payload for transferring data. Ant packets and data packets have the same priority that makes ant packets have similar network experience as data packets.

ACO-based selection function chooses a better output channel according to the pheromone of candidate channels. The pheromone is derived from the current and the historical local network information and it is called as *the state transition rule* is shown by (1), where *j* is the channel index (North, East, South, and West) and *d* is the destination index.

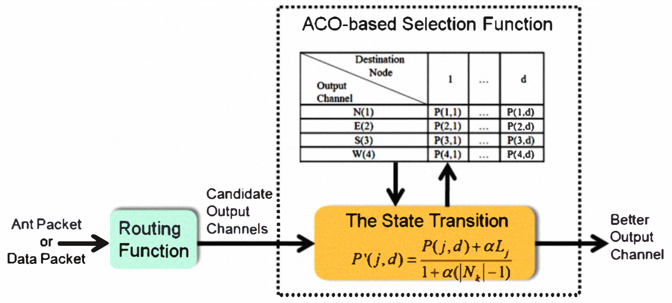

Figure 2.ACO-based adaptive routing.

The normalized pheromone *P'(j,d)* can be viewed as the probability of selecting channel *j* for destination *d.* The old pheromone *P(j,d)* is the historical local network information. L_j is the current local free buffer length of channel *j*, and N_k is the number of channels of router *k* .

$$P'(j,d)=\frac{P(j,d)+\alpha L_j}{1+\alpha(|N_k|-1)} \tag{1}$$

α is weighting coefficient for the current and the historical local network information. ACO needs a table for storing the historical local network information of all destination-channel pairs, which is shown in Table I.

TABLE I. Routing table of ACO-based Adaptive Routing.

Destination Node / Output Channel	1	2	...	d
N(1)	P(1,1)	P(1,2)	...	P(1,d)
E(2)	P(2,1)	P(2,2)	...	P(2,d)
S(3)	P(3,1)	P(3,2)	...	P(3,d)
W(4)	P(4,1)	P(4,2)	...	P(4,d)

At the training phase of the table, there are two equations for updating pheromone. For each router, the updating is based on the destination index *d* in the head flit of ant packet. *d* determines the column chosen to update. The pheromone of selected channel *j* in column *d* is increased by (2) and the other pheromones in column *d* are decreased by (3). *r* is the reinforcement factor indicating the congestion.

$$P(j,d)\leftarrow P(j,d)+r(1-P(j,d)),\quad j\in seleted\ channel \tag{2}$$

$$P(j,d)\leftarrow P(j,d)-rP(j,d),\quad j\notin seleted\ channel \tag{3}$$

ACO-based selection function has the potential to make a better decision than other congestion-aware selections. Because the historical network information can stand for the time-variant statistic traffic distribution in predicting the neighbor traffic status and the current network information can adjust the selection dynamically according to the local buffer state. The major problem of ACO-based adaptive routing is that the size of table increases tremendously as the scale of NoC grows. The column number of table is equal to the router number of NoC except local router, and row number is equal to the channel number of each router. Besides, every router has the table, so it is duplicated by the number of routers. Therefore, the cost of routing table is too high to implement in NoC systems.

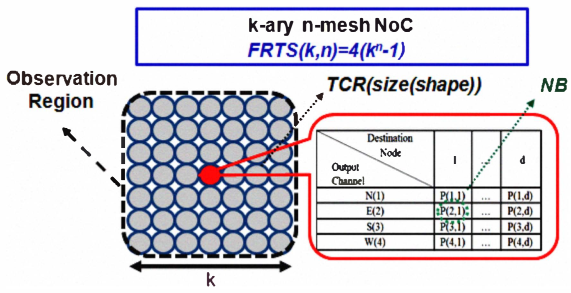

Figure 3. Area cost of routing table.

C. *Formulation of The Cost of ACO-based Adaptive Routing*

Given a *k*-ary *n*-mesh network topology *T(k,n)*, an application graph [3] and a routing function *R(i, Src, Dst)* at local router *i* for all packets are transferred from source router *Src* to destination router *Dst*. Use ACO-based selection function *ACOSel(ChSet,Info)* to select a better path $P_{sel}(i)$ of local router *i* from the candidate channel sets *ChSet* with the network information *Info*;

Find the minimum implementation cost of routing table with full routing table size *FRTS(k,n)*, the table compression ratio *TCR(size(shape))* and the number of bits *NB* is shown as Fig. 3. *TCR* is a function of the size of specific shape of observation region. And it is derived from the reformed table size divided by full table size;

Such that the path average latency $L(P_{sel}(i))$ with table-reformed technique is less than or equal to $L_{full\ routing\ table}$.

The cost problem of ACO-based adative routing can be formulated as follows:

$$\min\{FRTS(k,n)\times TCR(size(shape))\times NB\} \tag{4}$$

$$subject\ to \quad \sum_{i\in routers\ on\ the\ path} L(P_{sel}(i)) \le L_{full\ routing\ table}, \tag{5}$$

$$P_{sel}(i) = ACOSel(ChSet, Info(TCR, NB)), \tag{6}$$

$$ChSet = R(i, Src, Dst). \tag{7}$$

Intuitively, the objective function (4) minimizes the product of full table size, table compression ratio and the number of bits. Constraint (5) sets the bound for the table-reformed average latency is less than or equal to the average latency with full routing table. Constraint (6) shows a relation between ACO-based selection function and the path. And Constraint (7) shows a relation between the routing function and the candidate channels. In brief, we can reduce the *TCR* and *NB* to minimize routing table cost. Moreover, we use the cascaded routing with *adaptive searching* to ensure the reliability of *Info* for a better routing decision and conform to the bound of average latency.

III. Proposed ACO-Based Cascaded Adaptive Routing

ACO-CAR which considers the implementation cost and balances the traffic distribution. The table-reformed technique is proposed to reduce the *TCR* to minimize routing table cost. Additionally, to search cascaded point for cascaded routing, an *adaptive searching algorithm* is also proposed.

A. *Table-Reformed Technique*

In order to reduce the size of table, we propose table reforming technique to eliminate the redundant information from table. The state transition probabilities are stored in the table that includes the information of all destinations. However, the state transition probability of far destinations is derived from traffic status of local router. The information becomes an unnecessary and ineffective index in making a routing decision for local router. Hence, we can eliminate the information of far destinations from our table to diminish memory cost and try to keep the performance with negligible degradation.

Original observation region of routing table is all destinations in NoC and shape of observation region is square as shown in Fig. 3. We reform the observation region shape of routing table from square to rhombus which is shown as Fig. 4. Rhombus routing table can keep the same observation window to obtain the reliable information. And the hop counts from the boundary routers to the center router are the same, which is similar to the real ants diffusing pheromone toward every direction.

We define that the observation window size W_{ob} as the hop counts between the center and vertices of rhombus. And the definition of minimum observation window size $W_{ob,\ min}$ with the *k*-ary *n*-mesh network topology is as follows:

$$W_{ob,min} = \begin{cases} \left\lfloor \dfrac{nk}{6} \right\rfloor, & k\ is\ even \\ \left\lfloor n\left(\dfrac{k}{6} - \dfrac{1}{6k}\right) \right\rfloor, & k\ is\ odd \end{cases} \tag{8}$$

This concept stands for maximum hop counts between the opposite vertices of rhombus routing table, which is average hop counts over all source-destination pairs. In brief, the observation region can reserve the reliable pheromone within this range to make a routing decision with negligible degradation of performance. Moreover, *TCR* of rhombus in *k*-ary 2-mesh is given by

$$TCR(size(rhombus)) = \frac{(W_{ob})^2 + (W_{ob}+1)^2 - 1}{(k)^2 - 1} \tag{9}$$

Hence, the table cost is *TCR* multiplied by full table cost. Besides, the value of *TCR* value is always less than one. For example, given a 8-ary 2-mesh network topology. By (8) and (9), $W_{ob,\ min}$ is 2 and *TCR* is approximately equal to 0.19. Therefore, we reduce 81% cost of table size with 0.38% degradation of performance. The results are verified in section IV.

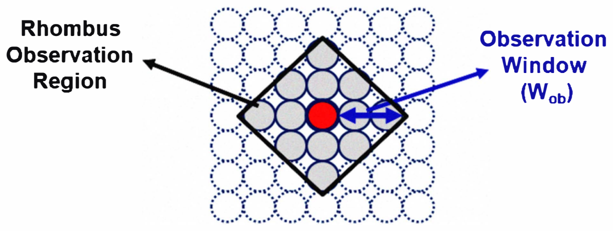

Figure 4. Observation region is rhombus.

B. Cascaded Routing with Adaptive Searching of Cascaded Point

Since some information of far destinations is not kept in our table by table-reformed technique; we cannot obtain the pheromone information directly from the observation region of local router. The routing algorithm does not discover a routing path in one-step for the far destination. Therefore, we propose the cascaded routing to overcome this problem. Cascaded point is a temporary destination between the local router and the destination. The packets are passed through the cascaded point and then routed to the destination. The flow and example of cascaded routing are shown as Fig. 5 and Fig. 6. The detailed executions of this flow and example are described as follows:

- *Step1:* We can check whether the destination is in the observation region.
- *Step2:* If the destination does not belong to the observation region, we need to search a cascaded point and route a packet to it.
- *Step3:* Then we can check whether the destination is in the observation region again. If the destination belongs to the observation region, the packet is routed to the destination directly and this process is terminated.

The challenge of cascaded routing is how to determine cascaded points at ant training phase. Intuitively, the *fixed* and *random searching* is used for search the cascaded point is shown as Fig. 7. *Fixed searching* chooses the cascaded point that is a vertex near the destination. *Random searching* chooses the cascaded point from the candidate points are between the source and destination with the minimal path. However, these searching methods do not consider the traffic status. The packets that might be forwarded to the busy cascaded point that brings about the traffic congestion. In order to search a better cascaded point, *adaptive searching* determines the cascaded point that has minimum number of ant packet being passed through a router in the observation region. The design concept based on the load-balancing and high degrees of adaptivity. It balances the number of packets being route through a router at the same time and increases adaptivity for the searching of cascaded point.

The number of training for routing table can affect the performance. By *adaptive searching*, we can balance the number of training for every destination-channel pair. Therefore, the reliable network information is constructed in the routing table. To summarize, cascaded routing can make a better routing decision with *adaptive searching* of cascaded point for more precise network information and conform to the bound of average latency which is listed as (5).

IV. PERFORMANCE EVALUATION

In order to minimize the implementation cost function as shown in (4), we can reduce the table compression ratio *TCR* and the number of bits *NB*. Firstly, we analyze the relation between the *TCR* of rhombus and performance in Experiment 1. Then the analysis of *NB* and performance are discussed in Experiment 2. In Experiment 3, the performance of *adaptive searching* is compared with *fixed searching* and *random searching*. In Experiment 4, the performance of table-reformed ACO-CAR is compared with the different selection functions, such as random, OBL and NoP under the condition of coupling with the same routing function. Finally, we show the statistical traffic load distribution (STLD) [12] of different selection functions in Experiment 5. According to the distribution, we can predict whether the traffic loads of a network are balanced or not.

The experiments are evaluated by the NoC simulator, Noxim [13]. We construct a 8-ary 2-mesh network topology. The system runs with wormhole switching mechanism [14] and round-robin arbitration. And we use the *odd-even* routing function [4] in our experiments. Each channel has an input queuing buffer with size of 4 flits and each packet has 8 flits. Besides, our experiments uses the *uniform distribution* and *transpose1 distribution* to evaluate performance. The maximum channel load of *transpose1 distribution* is more serious than *uniform distribution* [14]. Therefore, we will focus on the simulation results of *transpose1 distribution* in our experiments. And the simulation results of *uniform distribution* are conformed the performance of expectation. The time distribution of traffic is Poisson distribution. For each run of simulation is 52,000 cycles and the first 2,000 cycles is warm-up time of NoC system. The average latency and the saturation throughput are used as performance metric for our experiments. The definition of saturation throughput is where average latency is equal to twice of the zero-load latency [15].

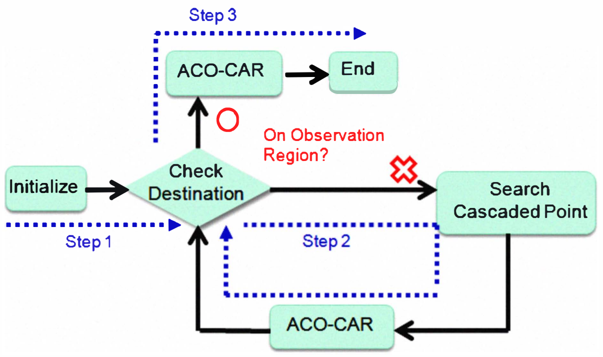

Figure 5. The flow of cascaded routing.

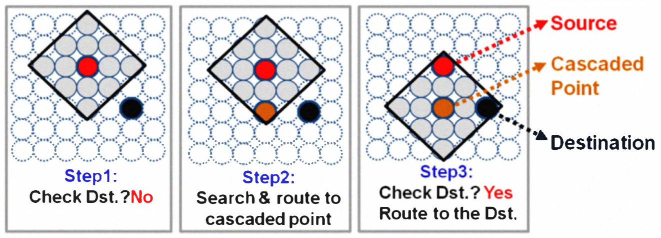
Source
Cascaded Point
Destination
Step1: Check Dst.? No
Step2: Search & route to cascaded point
Step3: Check Dst.? Yes Route to the Dst.

Figure 6. The example of cascaded routing.

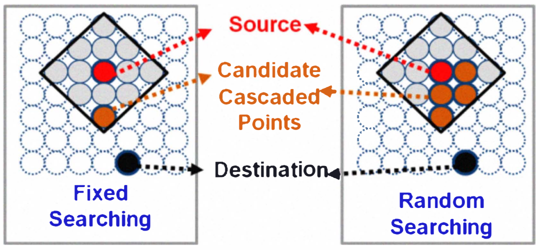

Figure 7. Fixed searching and random searching.

Experiment 1: Analysis of Table Compression Ratio

In order to reduce *TCR* to minimize the implementation cost, we analyze the relation between the *TCR* of rhombus and saturation throughput of a 8 × 8 mesh. The traffic model is *transpose1 distribution.* The process of experiment is to change the value of observation window size and record each saturation throughput. Fig. 8 shows the different cases of observation window size in 8 × 8 mesh. We can find the minimum *TCR* with 0.38% performance degradation, when W_{ob} is 2 that equal to $W_{ob,min}$ derived by (8). And *TCR* is 0.1905, it is means that the size of routing table can be reduced by 80.95%.

Experiment 2: Analysis of Number of Bits

In order to reduce *NB* for the minimization of implementation cost, we analyze the relation between the number of bits and saturation throughput for 8 × 8 mesh. The traffic model is *transpose1 distribution.* The process of experiment is to change the number of bits and record each saturation throughput. Fig. 9 shows the different cases of *NB* in 8 × 8 mesh. We can find that the saturated number of bits is 5 with 0.384% degradations of performance. In fact, we can choose 4 as our number of bits to reduce implementation cost with 2.29% performance degradation. However, the number of bits is 3 that bring about the 10% performance degradation. It is not feasible to NoC design. We choose 4 bits to implement our system in 8 × 8 mesh.

Experiment 3: Taffic of Different Searching Methods of Cascaded Routing

In order to make a better routing decision and construct the reliable network information in the routing table, we analyze the relation between the different searching methods and average latency for 8 × 8 mesh. We choose *TCR* to be 0.1905 and *NB* to be 4 according to the experiments in Experiment 1 and Experiment 2. The traffic model is transpose1 distribution. Fig. 10 shows the average latency about *fixed*, *random*, and *adaptive searching*. We can verify that *adaptive searching* performs better than other searching methods. Because the *fixed searching* always sends packets to the same cascaded point of vertex. It brings about hotspot traffic load on this cascaded point. And *random searching* does not consider the traffic load of cascaded point and packets that might be forwarded to the busy cascaded point with degradation of performance. To summarize, the improvements of *adaptive searching* to *fixed searching* and *random searching* are 39.6% and 12% in terms of saturation throughput respectively.

Experiment 4: Taffic of Different Selection Functions

The performance of ACO-CAR is compared with the random, OBL and NoP selection functions that are coupled with odd-even routing function. Besides, we compare the ACO-CAR and ACO selection with full routing table to ensure the relation of average latency is conformed to (5). The topology is 8 × 8 mesh and we choose *TCR* to be 0.1905 and *NB* to be 4 according to the experiments in Experiment 1 and Experiment 2. Fig. 11 shows the average latency of different selection functions with the traffic of *transpose1 distribution.* ACO-CAR has lower average latency and higher saturation throughput than other selection functions. The improvements to random, OBL, and NoP are 40%, 33.33% and 16.67% in terms of saturation throughput respectively. Fig. 12 shows the average latency of ACO-CAR and ACO selection with full routing table. And the average latency of ACO-CAR is less than or equal to the average latency of ACO selection with full routing table. To summarize, we minimize routing table cost by the reduced *TCR* and *NB,* and conform to the bound of average latency.

Experiment 5: Statistical Traffic Load Distribution

STLD represents the number of packets of all source-destination pairs passing through the corresponding routers [12]. According to the STLD, we can analyze whether the traffic loads of a network are balanced or not. Moreover, we use the standard deviation (σ) of traffic load to estimate the network performance. We consider the *center hotspot traffic distribution* that 4 hotspot nodes on center of 8 × 8 mesh with 20 % hotspot traffic load. Fig. 13 shows the traffic load distribution of NoP selection and ACO-CAR with odd-even routing function. The deep and light colors in Fig.13 stand for the heavy and light traffic load. Hence, there are dense loads on the center of mesh for NoP selection. In contrast, ACO-CAR results in balanced statistical traffic loads not only around the hotspot nodes but also on the whole network. Moreover, standard deviations of loads are 603 and 432 of NoP and ACO-CAR respectively. ACO-CAR can spread traffic load into the different paths evenly for traffic balancing.

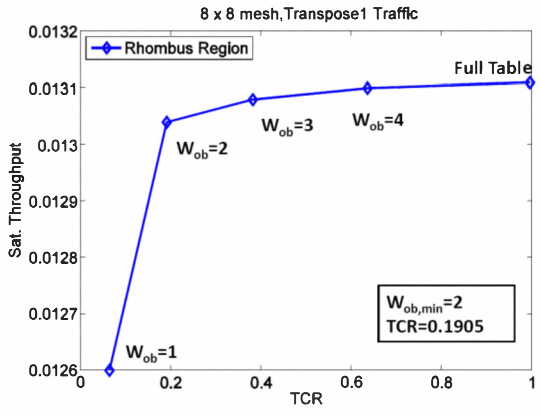

Figure 8. The saturation throughputs versus TCR in 8 × 8 mesh.

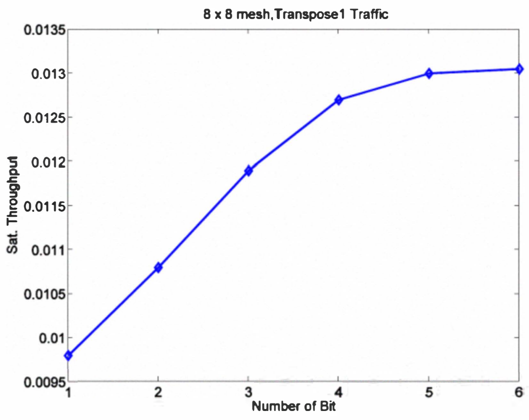

Figure 9. The saturation throughputs versus the number of bits.

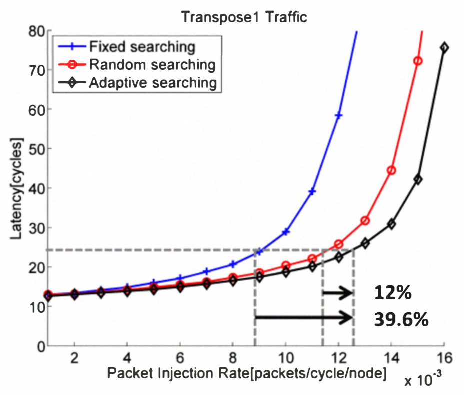

Figure 10. The average latency versus packet injection rate about fixed, random and adaptive searching.

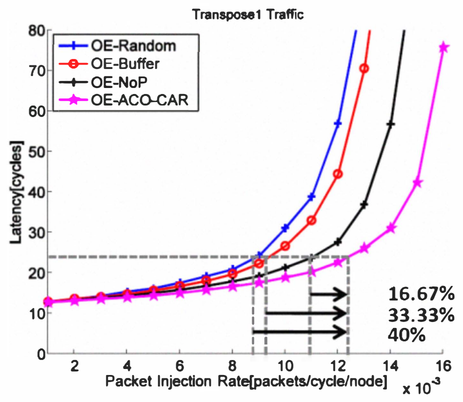

Figure 11. The average latency versus packet injection rate of random, OBL, NoP and ACO-CAR.

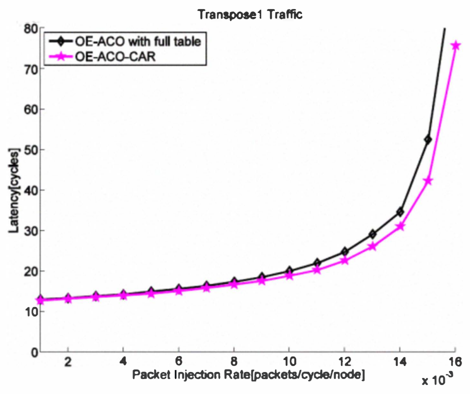

Figure 12. The average latency versus packet injection rate of ACO with full table and ACO-CAR.

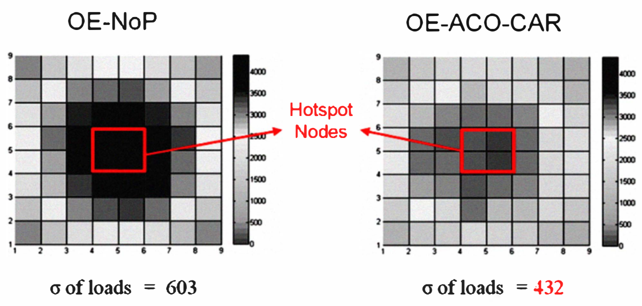

Figure 13. Statistical traffic load distribution of NoP and ACO-CAR.

V. Conclusions

In this paper, we propose an ACO-based Cascaded Adaptive Routing (ACO-CAR) for balancing the network load and reducing the implementation cost of routing table. The experiments have shown that the cost of routing table can be reduced by 80.95% with proposed table-reformed technique. Moreover, ACO-CAR has higher saturation throughput and gives 16.67% to 40% improvement compared to the random, OBL, and NoP selection function. According to the STLD, ACO-CAR can spread traffic load into the different paths evenly for traffic balancing.

References

[1] W.J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," *Proc. ACM/IEEE Design Automation Conf.*, pp. 684-689, 2001.

[2] L. Benini and G.D. Micheli, "Network on chip: a new paradigm for systems on chip design," *Proc. of the Conf. on Design, Automation and Test in Europe Conf. and Exhibition*, pp.418-419, 2002.

[3] R. Marculescu, U.Y. Ogras., L.-S. Peh, N.E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems,* vol. 28, no. 1, pp. 3-21, Jan. 2009.

[4] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans. on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729-738, July 2000.

[5] J.C. Martı´nez, F. Silla, P. Lo´pez, and J. Duato, "On the influence of the selection function on the performance of networks of workstations," *Proc. International Symp. High Performance Computing,* pp. 292-299, 2000.

[6] M. Daneshtalab, A. Sobhani, "NoC hot spot minimization using antnet dynamic routing algorithm," *IEEE International Conf. on Application Specific Systems, Architectures and Processors*, 2006

[7] M. Dorigo and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the travelling salesman problem, " *IEEE Trans. on Evolutionary Computation*, pp. 53-66, 1997

[8] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," *IEEE Trans. on Systems, Man, and Cybernetics B*, vol. 26, no. 2, pp. 29–41, 1996.

[9] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," *International Symp. on High-Performance Computer Architecture*, 2008.

[10] W.J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. on Parallel and Distributed Systems*, pp.466–475, April 1993.

[11] G. Ascia, V. Catania and M. Palesi, "Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip, " *IEEE Trans. on Computer*, vol. 57, no. 6, June. 2008.

[12] S.-Y. Lin, C.-H. Huang, C.-H. Chao, K.-H. Huang, and A.-Y. Wu, "Traffic-balanced routing algorithm for irregular mesh-based on-chip networks," *IEEE Trans. on Computers*, vol. 57, no. 9, pp. 1156–1168, September 2008.

[13] "Noxim: Network-on-Chip Simulator," http://sourceforge.net/projects/noxim , 2008.

[14] W.J. Dally and B. Towles, *Principles and practices of interconnection networks*, Morgan Kaufmann, 2004.

[15] L. Shang, L.-S. Peh, and N.K. Jha," Powerherd: a distributed scheme for dynamically satisfying peak-power constraints in interconnection networks," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 1, Jan. 2006.