Optimized Q-learning Model for Distributing Traffic in On-Chip Networks

Fahimeh Farahnakian , Masoumeh Ebrahimi, Masoud Daneshtalab, Juha Plosila, Pasi Liljeberg
Department of Information Technology, University of Turku
Turku, Finland
{fahfar,masebr,masdan,pakrli,juplos}@utu.fi

Abstract— **Many adaptive routing protocols have been developed for Networks -on-Chip to improve the network performance by traffic reduction. In this paper, we present an adaptive routing algorithm based upon the Q-routing, which distributes traffic by a learning method in the entire network. The learning method utilizes local and global traffic information and can select the minimum latency path to the destination. Since the routing table sizes become one of the main sources of area consumption in the Q-routing algorithm, we propose a clustering approach in order to reduce the area overhead. Furthermore, this approach improves the observability of the traffic condition. Experimental results for different traffic patterns and network loads show that the proposed method achieves significant performance improvement over the Q-routing, C-routing, DBAR and Dynamic XY algorithms.**

Keywords—Network-on-Chip, Adaptive routing algorithm, Congestion-aware routing algorithm, Q-routing

I. INTRODUCTION

As technology scales further and chip integration density grows, on-chip communication is playing an increasingly important role in System-on-Chip (SoC) design [1][2]. Network-on-Chip (NoC) has emerged as a dominant communication infrastructure in many core architectures to address many of the on-chip communication design issues such as scalability, reliability, and throughput [3][4][5][6][7]. In a mesh NoC architecture, each Processing Element (PE) is connected to its switch by a local network interface. Also, each switch is connected to its neighboring switches through bidirectional links [3][4]. PEs can communicate with each other by propagating packets through switches in the network. The performance and the efficiency of NoCs strongly depend on the routing algorithms that allow packets to be forwarded from any source to any destination. Generally, routing algorithms can be classified into deterministic and adaptive [8][9]. Deterministic routing algorithms [10] route packets in a fixed path between the source and destination switches. For example, in the XY routing algorithm, packets first transfer along the X direction, then along the Y direction. Implementations of the routing unit when utilizing deterministic algorithms are simple. However, when the packet injection rate increases, deterministic algorithms cannot balance the load across the links. In contrast, adaptive routing algorithms [11][12] can avoid congested links by choosing paths with low latencies. Therefore, they can decrease the probability of passing packets through congested areas.

In our method, we use a strong approach of machine learning and artificial intelligence — Reinforcement Learning (RL). In this approach, a system obtains an optimal solution without prior knowledge when the system offers alternative choices. One of the most important breakthroughs in RL is known as Q-learning [13] that has been applied in many different areas. In this algorithm, an agent first learns a model of the environment on-line and then utilizes this knowledge to find an effective control policy for the given task. So, Q-learning provides a learning algorithm for self-optimizing controller design which does not need a model of its environment [14]. The Q-learning algorithm was used to create an adaptive routing algorithm named Q-routing [15]. In Q-routing, each switch makes use of a table of Q-values (latency information), called Q-table. These values are updated whenever the switch sends a packet to one of its neighbors. As the switch routes packets, its Q-values gradually incorporate more global information.

The main contribution of this paper is to present an intelligent routing algorithm which is able to learn from the frequently changing conditions of the network to be used in adaptive routing packets. Since Q-routing imposes unacceptable area overhead to NoCs, we focus on keeping the area overhead small while improving the performance by utilizing appealing features of the clustering approach. The proposed algorithm, named Clustered Q-routing (CQ-routing), leads to a better distribution of traffic over the network.

The paper is organized as follows. Section II reviews the related work. Some background information on Q-routing and C-routing techniques is discussed in Section III. Section IV the proposed adaptive routing algorithm is explained. The results are reported in Section V while conclusion is given in the last section.

II. RELATED WORK

When the network size increases, traffic becomes a major limiting factor on the NoC performance. Adaptive routing algorithms partially address the network traffic problem by considering paths with low latencies [16][17]. These methods can distribute traffic to improve the reliability and communication efficiency in NoCs. Adaptive routing algorithms, depending on the degree of adaptiveness can be classified as partially adaptive or fully adaptive. Partially adaptive routing algorithms use some of the shortest paths between the sender and the receiver, but not all packets are allowed use every shortest path. For example the turn model routing is based on prohibiting certain turns during the packet routing to prevent deadlock [18]. Fully adaptive routing algorithms allow to route packets on any shortest path. To make the routing algorithm fully adaptive, there is a need of virtual channels in a wormhole switching network [19][20]. Most common implementations of routing algorithms, e.g. CATRA [11], HARAQ [21], NoP [22], RCA [23], DBAR [24], have focused on collecting local or global traffic information to get an estimation of the congested areas in the network. Some routing algorithms, like ANTNET [25] and QoS routing [26], not only include adaptively schemes but also using distributed learning methods. However, they are very expensive to be integrated inside the chip.

The Q-routing allows a network to be constantly adapted to the changing congestion condition and traffic flows. In this algorithm, each switch makes its routing decision based on the latency information, represented as a table of Q-values which estimate the quality of alternative routes. These values are updated each time a switch sends a packet to one of its neighbors. Depending on traffic patterns and load levels, only few Q-values are updated regularly while most of the Q-values in the network remain undated and thus unreliable. Some proposals have been presented to address this issue such as CQ-routing [27], PQ-routing [28], and DRQ-routing [29].

RL approaches have rarely been investigated in the context of NoCs. The algorithm in [30] is proposed to handle communication among modules which are placed on a reconfigurable NoC dynamically. Fault-tolerant deflection routing algorithm (FTDR) is presented in [31] which is inspired by Q-learning techniques for tolerating faults in NoC. In another approach [32], Q-routing is used to provide different levels of Quality-of-Service (QoS) such as Best Effort (BE) and Guaranteed Throughput (GT) in NoC. In this approach, a novel scheme is presented which contrasts the performance of Q-routing with the XY routing strategy in context of QoS in NoC. A Q-learning based Congestion-aware Algorithm [33] is presented to alleviate congestion condition in NoCs. It estimates and predicts the congestion condition of the network as close to the actual values as possible. Moreover, the Dual Q-routing Adaptive learning Rate [34] algorithm has enhanced Q-routing performance in NoC while the network becomes congested. It presents a congestion detection technique to update information dynamically according to changing traffic conditions. C-routing [35] is an adaptive routing which reduces the Q-table size compared to Q-routing algorithm. It chooses between XY and Q-routing algorithm by considering the source and destination locations.

Q-routing methods suffer from large area overhead in NoCs due to using large table sizes in each switch. The goal of this work is on gaining the performance out of learning methods while keeping the area as minimal as possible. In our proposed method, the network is divided into several clusters, each cluster maintaining a routing table instead of each switch. This approach can reduce the table sizes by the factor of four. It might be thought that this improvement is at the cost of degrading the performance. However, as shown in the result section, for instance, the performance is improved up to 25% over DyXY method in uniform traffic in 8×8 mesh network.

III. Background

In this section, we will give a brief introduction to the Q-routing, which is one of the most popularly adaptive routing algorithms. In addition, we describe the C-routing algorithm that perform Q-routing algorithm.

A. Q-routing

Q-routing is a routing algorithm taking advantage of the reinforcement learning aspect. The Q-values are stored in the Q-tables that indicate the estimated latencies of alternative routes. These values are updated each time a switch sends a data packet to one of its neighbors. In general, Q-routing dynamically learns a representation of the network state in terms of Q-values and uses these values to make routing decisions. Assume that the switch *x* sends a packet to one of its neighboring switches *y*, destined for the switch *d* (Fig. 1). The maximum amount of time it takes for a packet to reach its destination from the switch *x* is bounded by the sum of three quantities:

1. The waiting time in the packet queue of the switch *y* (q_y).
2. The transmission delay over the link from switch *x* to *y* (δ).
3. The time it would take for the switch *y* to send this packet to its destination via any of the switch *y*'s neighbors, assumed *z*. It can be formulated as:

$$Q_y(z,d) = \min_{n \in N(y)} Q_y(n,d)$$

Where *N(y)* is a set of the *y*'s neighboring switches. Thereby, the total latency can be calculated by:

$$Q_x(y,d)_{est} = Q_y(z,d) + q_y + \delta$$

$Q_x(y, d)_{est}$ is the switch *x*'s best estimated delay it would take for a packet to reach its destination switch *d* from the switch *x* when sent via its neighboring switch *y*. To update the latency estimation of switch *x*, the best estimate of switch *y*, $Q_x(y,d)_{est}$, sends back to the switch *x*. Upon receiving the estimation value, the switch *x* computes the new estimation for $Q_x(y,d)$ as follows:

$$Q_x(y,d)_{new} = Q_x(y,d)_{old} + \gamma\left(Q_x(y,d)_{est} - Q_x(y,d)_{old}\right) \quad (1)$$

In this way, learning is performed by updating the Q-values. Learning rate, γ, determines in which rate the newer information overrides the older one. Learning rate can take a value between zero and one; the value of zero means no learning is made by the algorithm; while the value of one indicates that only the most recent information is used.

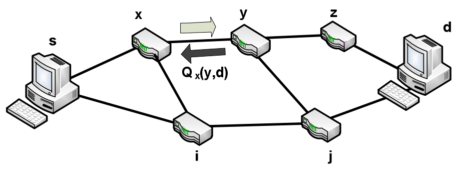


Fig. 1.An example of Q-routing method

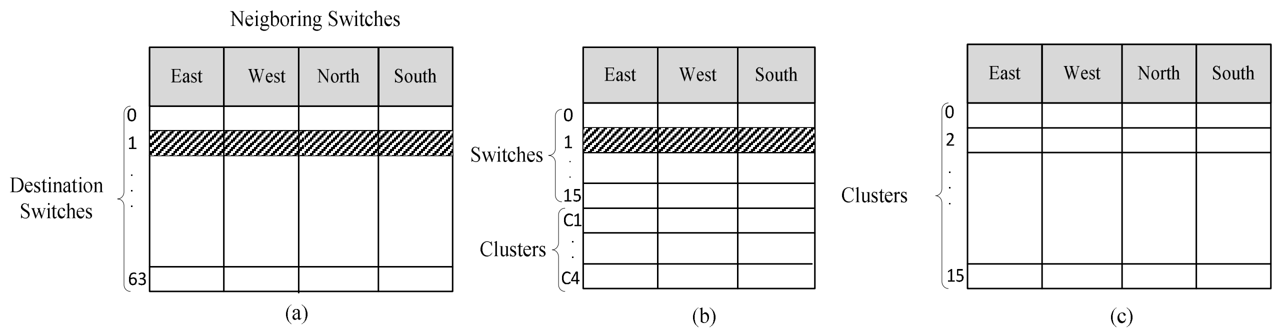


Fig. 2. Formats of different Q-tables in 8×8 mesh network (a) Q-table at switch 1, (b) CRotuing-table at switch 1, (c) CQ-table at cluster 1

B. C-routing

The C-routing algorithm is a combination of a deterministic routing algorithm (XY) and a Q-routing based adaptive routing algorithm. Depending on the location of source and destination switches, one of the routing algorithms is invoked. Similar to Q-routing method, in C-routing algorithm, each switch maintains a table of Q-values while a clustering approach is utilized to reduce the size of Q-table. For instance, this algorithm could reduce the table size by 75% compared with Q-routing in 8×8 mesh network.

IV. Clustered Q-routing

Our approach is able to balance traffic load by forwarding packets on less traffic routes in the network. We name the proposed algorithm Clustered Q-routing (CQ-routing) because it uses the Q-routing and a clustering approach together. The first step to implement CQ-routing is to form clusters. The number of clusters is defined according to the size of the network and a Q-table is associated with each cluster. In the second step, we explain the format of data packet (propagating data information) and learning packet (for carrying the Q-routing information). Finally, the routing method is extensively explained supported by an example.

A. CQ-table

Using Q-routing method in $n\times n$ 2D mesh, each switch maintains a Q-table with $n\times m$ entries; where n represents the number of switches in the network and m is the number of neighboring switches. Fig. 2(a) shows a Q-table of switch 1 in a 8×8 2D-mesh network. In this table, each row corresponds to a destination; each column relates to an output port; the contents of the table are the estimated latencies; and the next switch would be the neighboring switch connected to the minimum latency value for a corresponding destination. Since a separate row should be dedicated to each destination in the network, the area overhead of Q-tables becomes problematic in *NoCs.*

The size of the Q-table is reduced by using the CRouting-table (Fig. 2(b)). C-routing divides a network into C clusters and the Q-table sizes decrease to $((n/C)+C)\times(m-1)$. As an example, in a 8×8 mesh network, the table size is reduced by 75% compared with Q-tables.

The size of the Q-table can be further decreased by our clustering approach such that it can be employed in NoCs. The Q-table in this approach is called Cluster Q-table (CQ-table). An idea is that a CQ-table is maintained for each cluster instead of each switch. In this model each cluster maintains information about the routing cost when sending a packet to the possible destination clusters. Therefore, the CQ-table contains $C-1$ entries, where C is the number of clusters in the network. For example in a 8×8 mesh, the network is divided into 16 clusters. The number of switches within each cluster is considered to be four as the traffic condition within each cluster is roughly similar. As shown in Fig. 2(c), reduces the number of rows to 15. Each row of the table has four fields indicating the latency values to reach the destination cluster through each of the neighboring clusters. The CQ-table is referred by the switch whenever the destination cluster is not equal to its cluster number. Otherwise the XY routing is utilized. The area occupied by the Q-tables in the whole network for Q-routing, C-routing and CQ-routing can be calculated using Equation (2), (3) and (4), respectively as follow:

$$T_{Q-routing} = n\times(n\times m) \tag{2}$$

$$T_{C-routing} = n\times((D+C)\times(m-1)) \tag{3}$$

$$T_{CQ-routing} = \frac{n}{D}\times((C-1)\times m)) \tag{4}$$

In these equations, n refers to the number of switches in the network, m is the number of directions, C is the number of clusters, and D indicates the number of switches within each cluster. The required area of each method in 8×8 mesh is given in Table 1. As can be observed from the table; the size of Q-tables has been considerably reduced by our approach.

Table 1.The area overhead in a 8×8 mesh

Routing method	**No.of Clusters**	**No. of Tables**	**Occupied area in network**
Q-routing	0	64	64×64×4= 16384 bits
C-routing	4	64	64×20×3= 3840 bits
CQ-routing	16	16	16×(15×4)= 960 bits

B. Routing Packets

There are two types of packets that are propagated inside the network: data packets and learning packets. The header flit of data packet contains routing information such as destination and source addresses.

```
1.  Determine the current cluster(C2);
2.  Determine the destination cluster(Cd);
------------------------------------------------
3.  if  C2 = Cd  then
4.      Use xy_routing_algorithm;
5.  else
6.      Use learning_routing_algorithm;
7.  end if;
------------------------------------------------
8.  function learning_routing_algorithm
9.  --Extracting information from data massage header
10.         Cupstream  ← UpstreamCluster_ID;
11.         QTime ← QueueTime;
12. --Determining the neighboring cluster (C3) with minimum traffic load
13.         $C3 <= \min_{Cm \in N(C2)} Q(Cm, Cd)$
14. --Measuring total waiting time in input buffers as long as the packet traveling inside cluster C2
15.   Do
16.           QueueTime ← WaitingTimeinInputBuffer + QTime;
17.           Forward message to the neighboring node;
18.  Until (message is going to leave the cluster C2);
19. --Upon leaving the message, generate learning packet and send it back to cluster C1
20.      LocalLatency ← QTime/NumberOfHops;
21.      GlobalLatency ← QC2(C3,Cd);
22.      NewEstimatedLatency ← LocalLatency + GlobalLatency;
23.     RecivingCluster_ID ← C1;
24.    DestinationCluster_ID ← Cd;
25. --Updating the data message header before delivering packet into cluster C3
26.     UpstreamCluster_ID ← C2;
27.    QTime ← 0;
28. End function
```

Fig. 3. CQ-routing algorithm (state 1)

The learning packet carries local and global routing information to be used for updating the CQ-tables. In CQ-routing, the data packet format is modified by adding six bits into the header flit. These additional fields can be described as follows:

- *UpstreamCluster-ID:* determines the sending cluster ID of the data packet. Since each cluster is connected to at most four neighboring clusters, two bits are enough to encode the neighboring cluster IDs.
- *QTime:* determines the total waiting time of a packet in the input buffers of a cluster from when it enters the cluster until leaves the cluster. We have assigned four bits for this field.

The learning packet consists of three fields as follows:

- *ReceivingCluster-ID*: contains the cluster ID used to forward a learning packet. Four bits are considered for this field.
- *NewEstimatedLatency:* It is obtained by adding the local latency and global latency values. Local latency is measured by dividing the sum of waiting times into the number of hops taken by a packet within a cluster. In the Equation (1) the term q_y is the local latency. Global latency determines the expected latency of sending a packet from the next cluster to the destination cluster. The term $Q_y(z,d)$ in Equation (1) indicates the global latency. The latency value is determined with four bits.
- *DestinationCluster-ID:* determines the destination cluster ID to be used in updating the CQ-tables. Four bits are enough for this field.

C. *Routing algorithm*

In this subsection, we describe the functionality of the routing algorithm, and how a packet is routed from the source switch to the destination switch. In CQ-routing, every switch first determines the destination switch of the received packet. If the destination switch is located in the same cluster as the current switch, it uses the XY routing algorithm because the destination switch is close to the current switch and it does not need a complex routing algorithm to cope with. Otherwise; if the destination switch is located in another cluster than the current switch, CQ-routing chooses a neighboring cluster which has a lowest traffic condition. Similar to Q-routing, the proposed algorithm can be summarized in two states:

State 1- the functionality of CQ-routing algorithm is described in Fig. 3 when a switch has just received a packet in cluster *C2*. This packet has been originated by a switch in cluster *C1* and is going to be sent to a switch in the destination cluster *Cd*. After receiving a data packet by a switch in cluster *C2*, the switch compares the current and destination cluster IDs. If cluster IDs are the same, XY routing is employed, otherwise learning method is utilized (step 1-7). In the learning method, at first, *UpstreamCluster_ID* and *QTime* values are extracted from the

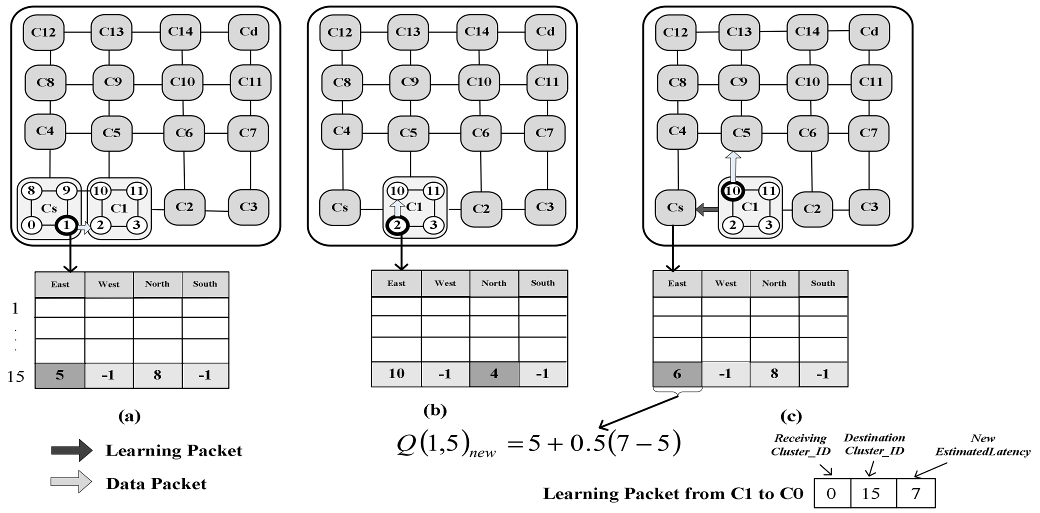


Fig. 4. An example of CQ-routing

data packet header (step 8-11). Then, a neighboring cluster with a minimum estimated latency is selected from the CQ-table (step 12-13). As long as the packet is traveling within a same cluster, the waiting time in the input buffers are summed together and carried by the packet header flit (step 14-18). Upon leaving the current cluster, the learning packet is generated and delivered to the previous cluster *C1*. The learning packet should carry the new estimation of the latency which consists of the local and global latencies. The local latency is obtained by dividing the sum of waiting times to the number of hops taken by a packet inside cluster *C2*. The global latency is extracted from the CQ-table corresponding to the cluster C2. This value shows the estimated latency of the packet in the remaining path from the next cluster *C3* to the destination cluster. These values are used to update the corresponding entry of the CQ-table in the cluster C1. The row of the table is determined by a destination cluster ID while the column is a direction in which the data packet has been already delivered from (step 19-24). Before sending the data packet to the next cluster, some modifications should be made in the header flit such as setting the upstream cluster to current cluster *C2* and resetting the waiting time value to zero (step 25-28).

State 2- describes the functionality of a switch in the cluster first receiving the learning packet.

As shown in Fig. 5, the switch takes the new estimated latency, column and row numbers out of the learning packet (step 1-4) and updates the CQ-table using the old and new estimation values (step 5-8).

```
1.  --Extracting information from learning massage
2.     column <= direction(ReceivingCluster_ID);
3.     row ← DestinationCluster_ID;
4.     Qest ← NewEstimatedLatency;
5.  --Update the corresponding entry of the CQ- routing table
      using Formula (5) as follow:
6.     Qold ← CQ_routing(row,column);
7.     Qnew ← Qold + 0.5 (Qest + Qold);
8.     CQ_routing(row,column) <= Qnew;
```

Fig. 5. CQ-routing algorithm (state 2)

Let us explain the CQ-routing algorithm via the example shown in Fig. 4. In this example, a system consisting of 64 IP blocks mapped onto mesh-based NoC architecture. A packet is generated at the source switch 1 for the destination switch 55. Since the CQ-routing is a minimal routing algorithm, the non-minimal paths values are set to -1. Therefore, two output channels (East and North) can be selected at switch 1 for forwarding the packet to the destination cluster (C15). As illustrated in Fig. 4(a), the east direction has the lowest latency to reach the destination cluster, so it is chosen as the next cluster. Using CQ-table information at switch 2, among north and east directions, the north cluster has the lowest estimated latency (Fig. 4(b)). Thus the switch 10 is selected for sending the packet to the switch 54. As indicated in Fig. 4(c), at switch 10, the north output channel is used to forward the packet. At this time, the learning packet is generated to return the local and global latencies back to C0. Local information is calculated by summing up the waiting time of the packet in the input buffers of switches 2 and 10 divided by the number of hops (2). This value is assumed to be 3 in our example. The minimum estimated latency of routing packet from the switch 10 to the C15 via the north cluster is considered as the global latency. This value is extracted from the CQ-table in C1 (i.e. global=4). The sum of the local and global values is new latency estimation from cluster C1 to the C15 which is put in the learning packet. Finally, the table C0 is updated whenever the learning packet is received. As shown in Fig. 4(c), the corresponding entry of the CQ-table at the C0 is updated taking an average of the new estimated value and an existing estimation by using Equation (5). A similar process is done until the packet reaches the C15. After that, CQ-routing utilizes XY routing in C15 for forwarding the packet to the destination switch 55.

V. Simulation Results

In this paper, we used a modular NoC simulator based on OMNeT++ framework to measure the end-to-end latency [36]. OMNeT++ is an open-source, extendible, easy traceability and C++-based simulator for modeling distributed and parallel system.

The performance of the routing scheme is evaluated through latency curves. For a given packet injection rate, a simulation is conducted to evaluate the average latency. It is assumed that latency is the duration from the time when the first flit is created at the source core, to the time when the last flit is delivered to the destination core. To propagate data packets, two virtual channels are used along the y and x dimensions, while for learning packets, one virtual channel is utilized. The buffer size per virtual channel is set to 8 flits. The simulator is warmed up for 3,000 packets and then the average performance is measured over 10,000 data packets. In each simulation time step (nanosecond), these packets are stored in the queue of the source switches. The amount of injected packets depends on the network offered load. The network offered load is calculated as follow:

Network offered load = flit-size/flit arrival delay

flit_size represents the size of the flit. It is assumed that the data packets have a fixed length of 8 flits with the flit width of 32 bits.

Flit-arrival-delay represents the delay time between previous flit generation and next flit generation.

A. Performance Evaluation under Traffic Load Profiles

The simulations were conducted on the 8×8 and 14×14 2D-mesh under various traffic patterns. Three synthetic traffic profiles including random, transpose and hot-spot traffics are used.

Random traffic: in random traffic, each core sends a packet to another core with a random probability. The destination of different packets in each switch is determined randomly using a uniform distribution. In Fig. 6, the average latency as a function of the offered load is plotted. As observed from the results, the proposed routing scheme achieves better performance compared with the other schemes. The CQ-routing method can alleviate the congested areas and performs considerably better than other schemes.

Transpose traffic: transpose traffic means that a switch with the coordinates *(i,j)* sends packets to switch with the coordinates *(n−j, n−i)* in an $n\times n$ mesh. Fig. 7 shows the CQ-routing behaves as efficiently as other routing algorithms. CQ-routing leads to the lowest latency due to the fact that it can distribute traffic more efficiently than the other four schemes. Using minimal routes along with the intelligent selection policy reduces the average network latency of CQ-routing for both mesh sizes.

Hotspot traffic: under the hotspot traffic pattern, one or more switches are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. This traffic represents a more realistic traffic pattern. In simulations, given a hotspot percentage of H, a newly generated packet is directed to each hotspot switch with an additional H percent probability. We simulate the hotspot traffic with a single hotspot switch at (4, 4) and (7, 7) in the 8×8 and 14×14 2D-meshes, respectively. The average latency of each network with H=10% are illustrated in Fig. 8. As observed from the figure, the Q-routing schemes (C-routing and CQ-routing) behave as efficiently as DBAR and DyXY especially in medium and high loads. As load increases, DBAR is unable to tolerate the high load condition, while the Q-routing schemes learn an efficient routing policy. CQ-routing method can alleviate the congested areas and perform considerably better than other schemes.

B. Area overhead

In C-routing method, an area-utilization (*AU*) metric is defined to compare the table sizes of Q-routing ($T_{Q\text{-}routing}$) and C-routing ($T_{C\text{-}routing}$) methods. AU is calculated by dividing the C-routing table size over the Q-table size. Similarly, AU is used to compare the table size of our method ($T_{CQ\text{-}routing}$) with the two other routing schemes. The area reduction of CQ-routing method over the Q-routing method ($AU_{Q\text{-}routing}$) is given by:

$$AU_{Q-routing}(\%) = (1 - \frac{T_{CQ-routing}}{T_{Q-routing}}) \times 100$$

The utilization of CQ-routing method over the C-routing method ($AU_{\text{C-routing}}$) can be expressed by:

$$AU_{C-routing}(\%) = (1 - \frac{T_{CQ-routing}}{T_{C-routing}}) \times 100$$

Table 2 shows the calculated AU metric in three techniques. For instance, our approach can reduce the area required for Q-tables by 94% and 75% over Q-routing and C-routing methods in 8×8 mesh network.

Table 2. Area reduction as compared CQ-routing to Q-routing and C-routing

Mesh Size	No. of Clusters	No. of Tables	AU Q-routing (%)	AU C-routing (%)
8×8	16	16	94%	75%
16×16	32	32	98%	83%
32×32	64	64	99%	93%
64×64	128	128	99%	93%

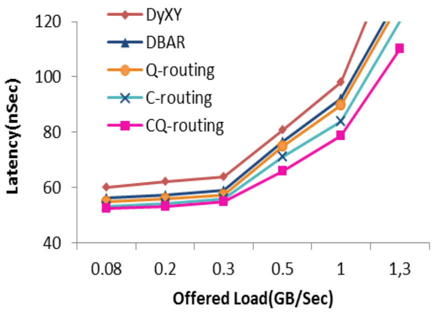
To estimate the hardware cost of our proposed method along with other routing schemes, the on-chip router of each scheme is implemented with VHDL and synthesized with Synopsys Design Compiler using the 65nm standard CMOS technology with a timing constraint of 1GHz for the system clock and supply voltage of 1V. The synthesized netlist is verified through post synthesis simulations. The layout areas of the three schemes are listed in Table 3. The area overhead of CQ-routing is smaller than Q-routing and C-routing methods.

Table 3. Hardware cost

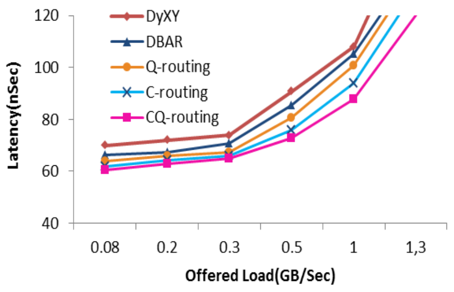
Routing method	Network Area (mm^2)
Q-routing	3.913
C-routing	3.187
CQ-routing	3.025

IV. Conclusion

In this paper, we proposed an optimized Q-routing algorithm named CQ-routing. The idea behind our algorithm is in clustering the network to monitor the traffic in the cluster (region) level rather than the switch level. Each cluster maintains a routing table that is updated dynamically when a switch receives a learning packet. This packet is provided local and global traffic information for each switch. By our proposed approach, not only the network performance can be improved, but also the area overhead of adding routing tables can be reduced by four times.

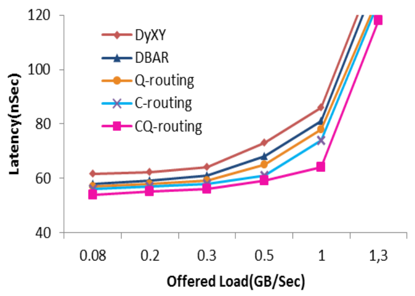


(a)

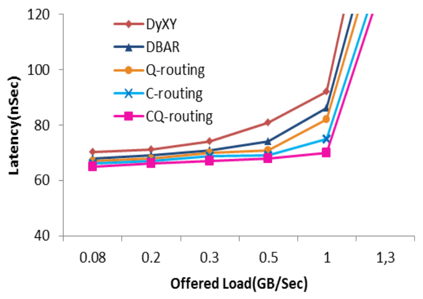


(b)

Fig. 6. Performance under different traffic models in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under the random traffic model

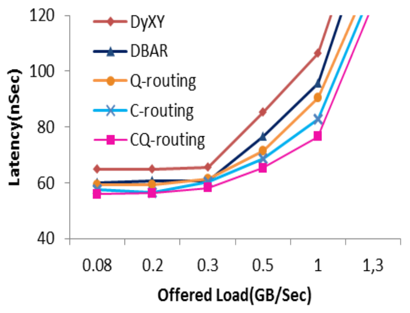


(a)

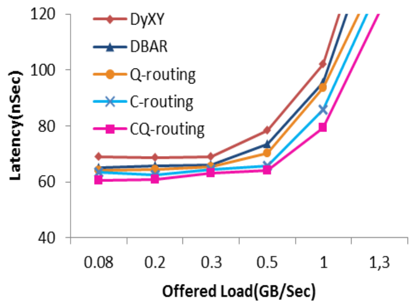


(b)

Fig. 7. Performance under different traffic models in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under the transpose traffic model



(a)



(b)

Fig. 8. Performance under different traffic models in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under the hotspot traffic model

REFERENCES

[1] D. Wu, B. M. Al-Hashimi, and M. T. Schmitz, "Improving Routing Efficiency for Network-on-Chip through Contention-Aware Input Selection", in Proc. of 11th Asia and South Pacific Design Automation Conference, pp. 36-41, 2006.

[2] T. C. Xu et al., "Explorations of Optimal Core and Cache Placements for Chip Multiprocessor," In Proceedings of the 29th IEEE Norchip Conference (Norchip), pp.1-6, 14-15 November 2011, Lund, Sweden;

[3] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks", in Proc. of DAC, pp. 684-689, USA, 2001.

[4] L. Benini and G. De Micheli, "Networks on chips: A new SoC paradigm", IEEE Computer, pp. 70-78, 2002.

[5] T. C. Xu, P. Liljeberg, and H. Tenhunen, "A Minimal Average Accessing Time Scheduler for Multicore Processors," In Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), LNCS 7017/2011, pp.287-299, 24-26 October 2011, Melbourne, Australia.

[6] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: A scalable, communication-centric embedded system design paradigm", in Proc. of VLSI Design, pp. 845-851, India, 2004.

[7] T. C. Xu, P. Liljeberg, and H. Tenhunen, "An Optimized Network-on-Chip Design for Data Parallel FFT," Procedia Engineering, Volume 30, March 2012, Pages 313-318.

[8] L. M. Ni, P. K. McKinley, "A survey of wormhole routing techniques in direct networks", Computer, pp. 62-76,1993.

[9] M. Ebrahimi et al., "MAFA: Adaptive Fault-Tolerant Routing Algorithm for Networks-on-Chip," in Proceedings of 15th IEEE Euromicro Conference On Digital System Design (DSD), pp. 201-206, Sept 2012, Turkey.

[10] E. Rijpkema et al., "Trade-offs in the design of a switch with both guaranteed and best-effort services for networks on chip", IEE Proceedings: Computers and Digital Techniques, pp. 294-302, 2003.

[11] M. Ebrahimi et al., "CATRA ─ Congestion Aware Trapezoid-based Routing Algorithm for On-Chip Networks," in Proceedings of 15th ACM/IEEE Design, Automation, and Test in Europe (DATE), pp. 320-325, Mar. 2012, Germany.

[12] T. T. Ye, L. Benini, G. De Micheli, “Packetization and routing analysis of on-chip multiprocessor networks”, Journal of Systems Architecture, pp. 81-104, 2004.

[13] C.J.C.H. Watkins and P. Dayan, “Q-Learning”, in Proc. Machine Learning, pp.279-292, 1992.

[14] R.S. Sutton and A.G. Barto, “Reinforcement Learning. An Introduction”, MIT Press, Cambridge, MA, 2000.

[15] J.A.Boyan, M. L .Littman, “Packet routing in dynamically changing networks:A reinforcement learning approach”, Advances in Neural Information Processing Systems 6., pp. 671-678,1994.

[16] D. Kim, S. Yoo, S. Lee, “A Network Congestion-Aware Memory Controller”, in Proc. of Fourth ACM/IEEE International Symposium on Networks-on-Chip, pp. 257-264, 2010.

[17] A. Sobhani et al., “Dynamic Routing Algorithm for Avoiding HotSpots in On-chip Networks,” in Proceedings of 2th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS), pp. 179-183, Sep 2006, Tunis.

[18] G.-M. Chiu, “The Odd-Even Turn Model for Adaptive Routing”, IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 7, 2000.

[19] Y. M. Boura and C. R. Das, “Efficient Fully Adaptive Wormhole Routing in n-Dimensional Meshes”, in Proceedings of the 14th International Conference on Distributed Computing Systems (ICDCS), 1994.

[20] A. A. Chien and J. H. Kim, “Planar-Adaptive Routing: Low-Cost Adaptive Networks for Multiprocessors”, Journal of the ACM, vol. 42, no. 1, 1995.

[21] M. Ebrahimi et al., “HARAQ: Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in On-Chip Networks,” in Proceedings of 6th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), pp. 19-26, May 2012, Denmark.

[22] G. Ascia, et al., “Implementation and Analysis of a New Selection Strategy for Adaptive Routing in Networks-on-Chip”, IEEE Transaction on Computers, v.57, I.6, pp. 809-820, 2008.

[23] P. Gratz et al., “Regional Congestion Awareness for Load Balance in Networks-on-Chip”, in Proc. HPCA, pp. 203-214, 2008.

[24] S. Ma, et al., “DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip”, in Proc. of ISCA, pp.413-424, 2011.

[25] M. Daneshtalab, A. Sobhani, M. D. Mottaghi, A. Afzali-Kusha, O. Fatemi, and Z. Navabi, “Ant Colony Based Routing Architecture for Minimizing HotSpots in NOCs,” in Proceedings of 19th ACM/IEEE International Symposium on Integrated Circuits and Systems Design (SBCCI), pp. 56 - 61, Sep 2006, Brazil.

[26] M. Mottaghi, M. Daneshtalab, “Using Routing Agents for Improving the Quality of Service in General Purpose Networks,” in Proc. of International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS), pp. 609-613, March 2011, Portugal.

[27] Shailesh Kumar, and Risto Miikkulainen, “Confidence-Based Q-routing: An On-Line Adaptive Network Routing Algorithm”, in Smart Engineering Systems: Neural Networks, Fuzzy Logic, Data Mining, and Evolutionary Programming, Vol. 8, pp. 147-152, 1998.

[28] S.P. Choi and D.-Y. Yeung, “Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control”, In Advances in Neural Information Processing Systems 8 (NIPS8), 945–951, 1996.

[29] S. Kumar and R. Miikkulainen, “Dual reinforcement Q-routing: An on-line adaptive routing algorithm”, in Proc. of the Artificial Neural Networks in Engineering Conference, pp. 231-238, 1997.

[30] M. Majer, et al. “Packet Routing in Dynamically Changing Networks on Chip”. in Proc. of the 19th International Parallel and Distributed Processing Symposium (IPDPS). Denver, CO, U.S.A. , 2005.

[31] C. Feng, Z. Lu, A. Jantsch, j. Li, and M. Zhang, “A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip”, in Proc of NoCArc, pp.11-16 , 2010.

[32] K. K. Paliwal, J. S.George, N. Rameshan, V. Laxmi, M.S.Gaur, V. Janyani, and R.Narasimhan, “Implementation of QoS Aware Q-routing Algorithm for Network-on-Chip”, pp. 370-380, 1C3 2009.

[33] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, “ Q-learning based Congestion-aware Routing Algorithm for On-Chip Network”, in Proceedings of 2th IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA), 2011.

[34] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, “An Efficient Q-Routing in Network-on-Chip Using Adaptive Learning Rate”, in Proceedings of IEEE of International Conference on Embedded Computer Systems (SAMOS), 2012.

[35] M.K. Puthal, V. Singh, M.S. Gaur,and V. Laxmi, ”C-routing: An Adaptive Hierarchical NoC Routing Methodology” , 19th International Conference on VLSI and System-on-Chip, pp. 392 – 397, 2011.

[36] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny, “NoCs simulation framework for OMNeT++”, in Proc. of NOCS, pp.265-266, 2011.