2012 Sixth IEEE/ACM International Symposium on Networks-on-Chip

HARAQ: Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in On-Chip Networks

Masoumeh Ebrahimi[1], Masoud Daneshtalab[1], Fahimeh Farahnakian[1],
Juha Plosila[1], Pasi Liljeberg[1], Maurizio Palesi[2], Hannu Tenhunen[1]

[1]University of Turku, Finland, [2]University of Kore, Italy

***Abstract*— the occurrence of congestion in on-chip networks can severely degrade the performance due to increased message latency. In mesh topology, minimal methods can propagate messages over two directions at each switch. When shortest paths are congested, sending more messages through them can deteriorate the congestion condition considerably. In this paper, we present an adaptive routing algorithm for on-chip networks that provide a wide range of alternative paths between each pair of source and destination switches. Initially, the algorithm determines all permitted turns in the network including 180-degree turns on a single channel without creating cycles. The implementation of the algorithm provides the best usage of all allowable turns to route messages more adaptively in the network. On top of that, for selecting a less congested path, an optimized and scalable learning method is utilized. The learning method is based on local and global congestion information and can estimate the latency from each output channel to the destination region.**

I. INTRODUCTION

Networks-on-Chip (NoC) has emerged as a solution to address the communication demands of future multicore architectures due to its reusability, scalability, and parallelism in communication infrastructure [1]. The performance and efficiency of NoCs largely depend on the underlying routing model which establishes a connection between input and output channels in a switch.

In minimal adaptive routing algorithms, shortest paths are used for transmitting messages between switches. In low traffic loads, minimal methods can achieve optimized performance, while they are very inefficient in avoiding hotspots when the network load increases. The reason is that they can deliver messages through at most two minimal directions and thereby they cannot reroute messages around congested areas. The routing policy (output selection) of minimal methods can be based on local, non-local, or mix of local and non-local congestion of the network. However, minimal routing algorithms suffer from a low degree of adaptiveness, which are inefficient in distributing the traffic over the network even if they have accurate knowledge of the network condition.

In wormhole routings, messages are divided into small flits traveling through the network in a pipelined fashion. This approach eliminates the need to allocate large buffers in intermediate switches along the path. However, a message waiting to be allocated to an outgoing channel may prohibit other messages from using the channels and buffers and thereby wasting channel bandwidth and increasing latency. Adding virtual channels can alleviate this problem, but it is an expensive solution. Non-minimal methods can partially overcome this blocking problem and reduce the waiting time of messages by delivering them via alternative paths. In contrast, performance can severely deteriorate in non-minimal methods due to the uncertainty in finding an optimal path as they may choose longer paths and meanwhile delivering messages through congested regions. Moreover, non-minimal methods can suggest minimal and non-minimal paths between a source and destination but this flexibility is at the cost of a more complex switch structure or additional virtual channels. On the other hand, an output selection should choose a single channel from a set of predetermined channels to forward a message to the next hop. This becomes one of the main challenges involved in designing an efficient non-minimal method to select a less congested path from a set of alternative paths. The decision for an output channel should not be based on local information as it may route messages through paths which are not only longer but also highly congested. On the other hand, even if a global knowledge of the network is provided, due to a large number of alternative paths, finding a less congested path is questionable which demands an intelligent method to cope with.

Reinforcement learning has attracted considerable attention in industry and academia because it provides an effective model for problems where optimal solutions are analytically unavailable or difficult to obtain. The learning methodology is based on the common-sense idea that if an action is followed by a satisfactory state, or by an improvement, then the tendency to produce that action is strengthened, i.e., reinforced. On the other hand, if the state becomes unsatisfactory, then that particular action should be suitably punished [2][3]. Q-Learning [4] is one of the algorithms in the reinforcement learning family of machine learning. In the Q-Learning approach, the learning 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. Q-Routing [5] is a network routing method based on Q-Learning models which learn a routing policy to minimize the delivery time of messages to reach their destinations [6]. Q-Routing methods are implemented by having each switch maintains a table of Q-values, where each value is an estimate of how long it takes for a message to be delivered to a destination, if sends via a neighboring switch [4]. The method requires a switch to update its routing table whenever a message is delivered to the next switch and the congestion information is returned. Q-Routing methods allow a network to be continuously adopted to load changes. Although these schemes make the most optimal routing decisions, due to employing large tables, they are not cost-efficient approaches for Networks-on-Chip.

In this paper, we proposed a novel routing algorithm, named Highly Adaptive Routing Algorithm using Q-Learning (HARAQ) where the main contributions of the paper are summarized as follow:

1. A low-restrictive non-minimal algorithm to provide several alternative paths between each pair of source and destination switches.

The algorithm uses only an extra virtual channel in the Y dimension and provides a large number of paths for routing messages. Different turns are defined on each virtual channel,

978-0-7695-4677-3/12 $26.00 © 2012 IEEE
DOI 10.1109/NOCS.2012.10

such that the prohibited turns in one virtual channel are permitted in the other one. Another subtle point of the presented method is its ability to enable 180-degree turns on a single channel (i.e. a message can arrive through a channel that is previously used to deliver it) without creating cycles.

2. An efficient output selection strategy for finding a low-latency path from a source to a destination.

The presented routing scheme utilizes an optimized Q-Learning model for the output selection function. In this way, the output selection can efficiently estimate the latency of a message to reach its destination through each of the possible output channels. This information is extracted from a Q-Table available at each switch. Unlike typical Q-Routing methods, our proposed model is scalable and the size of Q-Tables is relatively small.

The rest of this paper is organized as follows: Section II reviews the related work. The minimal mad-y method and the proposed highly adaptive routing algorithm along with the presented optimal Q-Routing model are explained in Section III. The results are given in Section IV while we summarize and conclude in the last section.

II. Related Work

Most common implementations of minimal routing algorithms, e.g. FRA [7], NoP [8], RCA [9], and DBAR [10], have focused on collecting local or non-local congestion information to get an estimation of the congested areas in the network. However, due to a low degree of adaptiveness (i.e. at most two directions per switch in fully adaptive minimal methods), minimal routing algorithms cannot distribute the traffic over the network efficiently even if they have accurate knowledge of the network condition. Virtual channels can be used to avoid deadlock and increase adaptiveness. DyXY [11] and mad-y [12] are the methods using only two virtual channels along one of the two physical channels. Although they are fully adaptive in minimal paths, messages are limited in terms of routing options and thus the traffic load cannot be efficiently balanced over the network. Generally, non-minimal routing schemes have been proposed for tolerating faults rather than avoiding congestion [13][14][15]. Most of these models are proposed to support special cases of faults, such as one-faulty switches, convex or concave regions, which require a large number of virtual channels to avoid deadlock. In fact, adding virtual channels is expensive because of additional buffers and complex control logics. There are other fault-tolerant approaches [16][17][18] which do not require any virtual channels. However, these algorithms are partially adaptive and very limited in supporting faults. A highly resilient routing algorithm is proposed in [19] to tolerate a large number of faults without exploiting any virtual channel, but only one path can be selected between each pair of source and destination nodes. In general, each method defines a new tradeoff between the number of virtual channels, the ability to handle different fault models, and the degree of adaptiveness.

Fault-tolerant methods cannot be efficiently employed as non-minimal models to alleviate congestion in NoCs for several reasons. If congestion occurs in multiple disjoint concave/convex regions at the same time, the fault-tolerant models with a small number of virtual channels are either deterministic or unable to handle different hotspot (fault) models while using a large number of virtual channels is not cost efficient for on-chip networks. In addition fault-tolerant algorithms are relatively complex due to considering different fault models in order to find a path between a source and destination. The complexity added to the algorithms is unnecessary for congestion-aware methods because congestion does not disconnect a path and in some cases highly congested paths might be selected. Most of the proposed fault-tolerant algorithms support only static faults and a few of those take dynamic faults into consideration. If hotspots changes dynamically, the algorithms should be reconfigured for new hotspots and handle a transition phase whenever a location of hotspot changes. To overcome this, some fault-tolerant algorithms allow dropping messages in the transition phase and other approaches use relatively complex models which are not appropriate solutions for on-chip networks. In sum, the goal of our approach is to present a low-restrictive method using both minimal and non-minimal paths. The method utilizes only one virtual channel along the Y dimension while does not have the above mentioned restrictions of fault-tolerant models.

Q-Routing based models have been studied in several literatures such as [20] and [21], but they have rarely been investigated in the context of on-chip networks. The algorithm in [6] is proposed to handle communication among modules which are dynamically placed on a reconfigurable NoCs. This algorithm is inspired by the method in [21] for a general case of message switching networks. FTDR-H [22] utilizes Q-Routing methods to tolerate faults and find a path between each pair of switches as long as a path exists. Moreover, the size of Q-Tables is reduced by taking advantages of the clustering model. This clustering model is also extensively discussed in the C-Routing method [23].

III. Highly Adaptive Routing Algorithm Using Q-Learning Model

The proposed mesh-based routing scheme is based on the mad-y method which has been introduced by Glass and Ni in [12]. The mad-y and the presented routing methods utilize a double-Y network where the X and Y dimensions have one and two virtual channels, respectively (Fig. 1(a)). Each switch in the double-Y network has seven pairs of channels, i.e. East(E), West(W), North-vc1(N1), North-vc2(N2), South-vc1(S1), South-vc2(S2), and Local(L). In 2D mesh-based network, three types of turns can be taken: 0-degree, 90-degree, and 180-degree turns (U turns) [24]. By taking a 0-degree turn, a message transmits in a same direction with a possibility of switching between virtual channels. The turn is called 0-degree-ch if in a turn neither the direction nor the virtual channel changes (Fig. 1(b)) while it is a 0-degree-vc turn if the virtual channel changes (Fig. 1(c)). By taking a 90-degree turn, a message transmits between the switches in perpendicular directions (Fig. 1(d)). By taking a 180-degree turn, a message is transferred to a channel in the opposite direction [24]. If the virtual channel is changed, the turn is called 180-degree-vc (Fig. 1(e)); otherwise, it is represented as 180-degree-ch (Fig. 1(f)). Note that, in all figures, the vc1 and vc2 are differentiated by – and = respectively.

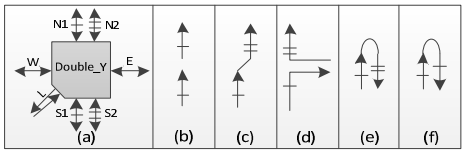

Fig. 1. (a) A switch in a double-Y network (b) 0-degree-vc (c) 0-degree-ch (d) 90-degree (e) 180-degree-vc (f) 180-degree-ch

A. The mad-y Method

In order to avoid deadlock, the mad-y method [12], prohibits several turns in the double-Y network. In mad-y, the 0-degree-vc turns from vc1 to vc2 are permitted, also all 0-degree-ch turns are allowable, however 0-degree-vc turns from vc2 to vc1 may cause deadlock in the network and are prohibited (Fig. 2(c) and Fig. 2(d)). As illustrated in Fig. 2(a) and Fig. 2(b), out of sixteen 90-degree turns that can be potentially taken in a network, four of them cannot be taken in mad-y. Finally, 180-degree turns are not allowed in mad-y. To prove the deadlock freeness in mad-y, a two-digit number (a, b) is assigned to each output channel of a switch in n×m mesh network. According to the numbering mechanism, a turn connecting the input channel (a_{ic}, b_{ic}) to the output channel (a_{oc}, b_{oc}) is called an ascending turn when (a_{oc}>a_{ic}) or ((a_{oc}=a_{ic}) and (b_{oc}>b_{ic})). Fig. 3 shows how the channels of a switch at the position (x,y) are numbered. Since this numbering mechanism causes the messages to take the permitted turns in strictly increasing order, so that mad-y is deadlock-free.

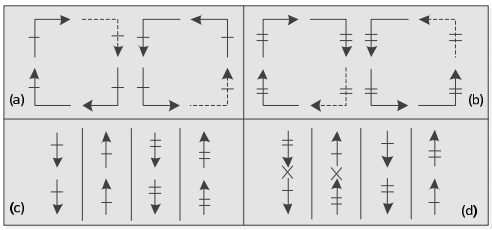

Fig. 2. 90-degree turns in (a) vc1 (b) vc2 (c) 0-degree-ch (d) 0-degree-vc

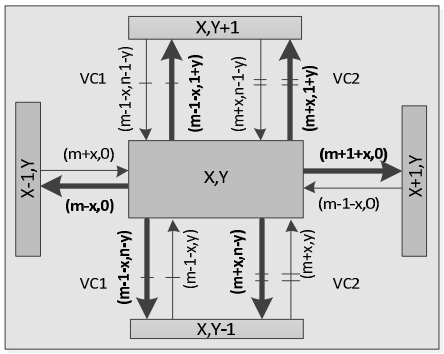

Fig. 3. Channel numbering in the mad-y method

B. Highly Adaptive Routing Algorithm

As mad-y is a minimal adaptive routing method, it cannot fully utilize the eligible turns to route messages through less-congested areas. The aim of the proposed routing algorithm, named Highly Adaptive Routing Algorithm (HARA), is to enhance the capability of the existing virtual channels in mad-y to reroute messages around congested areas and hotspots. Since the mad-y and HARA methods combine two virtual channels with different prohibited turns, they diminish the drawbacks of turn models prohibiting certain turns at all locations. In minimal routings, (e.g. mad-y), 180-degree turns are prohibited but can be incorporated in non-minimal routings. One way to incorporate 180-degree turns is to examine the turns one by one to see whether the turn causes any cycle. After determining all allowable turns, in order to prove deadlock freeness, the numbering mechanism is utilized. At first we use the numbering mechanism of the mad-y method to learn all 180-degree turns that can be taken in ascending order, and then modify the numbering mechanism to meet our requirements. According to the numbering mechanism in Fig. 3, among 180-degree-vc turns, those from vc1 to vc2 are taken in ascending order (Fig. 4(a), Fig. 4(b)), so that it is safe to employ them in the network. As all 180-degree-vc turns from vc2 to vc1 take place in descending order, so they cannot be used in the network (Fig. 4(c), Fig. 4(d)). Now, let us examine a 180-degree-ch turn when a message uses it in vc1 of the north direction (Fig. 4(e)). As shown in Fig. 3, the label on the output vc1 of the north direction is (m-1-x,1+y) and the label on the input vc1 of the north direction is (m-1-x,n-1-y). The turn takes place in ascending order if and only if n-1-y is greater than 1+y. Therefore, this turn can be safely added to a set of allowable turns if the y coordinate of a switch is less than (n-2)/2. Similarly, in Fig. 4(f), 180-degree-ch turn on the vc2 of the north direction is permitted if the y value of a switch is less than (n-2)/2. 180-degree-ch turns on the vc1 and vc2 of the south direction are permitted (Fig. 4(g) and Fig. 4(h)) if and only if the y coordinate of a switch is greater than n/2. Finally, 180-degree-ch turn on the west direction is always permitted (Fig. 4(i)) while 180-degree-ch turn on the east direction is prohibited in the network (Fig. 4(j)).

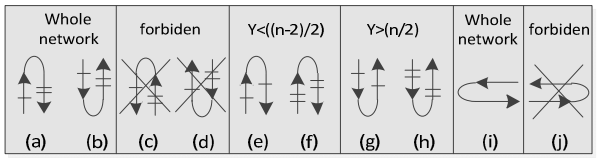

Fig. 4. Allowable 180-degree turns in the HARA method

As shown in Fig. 4, there are four conditional 180-degree turns. Two of those are allowable only in the northern part of the network and two others in the southern part of the network. This not only increases the complexity of the routing function but also imposes heterogeneous routing function for switches. To overcome this issue, we modify the numbering mechanism such that two turns are permitted in the whole network (Fig. 4(g) and Fig. 4(h)) and two other are prohibited (Fig. 4(e) and Fig. 4(f)). The numbering mechanism of HARA along with all permitted turns in the network is shown in Fig. 5. As can be observed from this figure, all allowable turns are taken in ascending order.

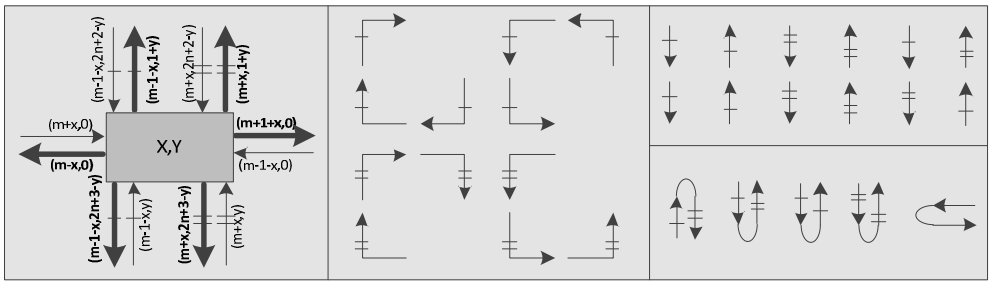

Fig. 5. The numbering mechanism of HARA along with all eligible turns in the network

Table 1. Potential output channels according to the input channel (InCh) and relative position of source and destination switches (Pos)

Pos. / InCh.	N	S	E	W	NE	NW	SE	SW
L	N1, N2, S1, W	N1, S1, S2, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W
N1	N2, S1, W	S1, S2, W	N2, S1, S2, E, W	S1, W	N2, S1, S2, E, W	S1, W	N2, S1, S2, E, W	S1, W
N2	-	S2	S2, E	-	S2, E	-	S2, E	-
S1	N1, N2, S1, W	N1, S1, S2, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W
S2	N2	-	N2, E	-	N2, E	-	N2, E	-
E	N1, N2, S1, W	N1, S1, S2, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1,S1, W
W	N2	S2	N2, S2, E	-	N2, S2, E	-	N2, S2, E	-

In the non-minimal routing, employing only eligible turns at each switch is necessary but not sufficient to avoid blocking in the network. The reason is that using the allowable turns (Fig. 5), a message may not be able to find a path to the destination from the next hop and is blocked. On the other hand, one of the aims of HARA is to fully utilize all eligible turns to present a low-restrictive adaptive method in the double-Y network. To achieve the maximal adaptiveness along with the blocking avoidance, for each combination of the input channel and destination switch position, we examined all eligible 0-degree, 90-degree, and 180-degree turns, separately. The output channels are selected in a way that not only the turn is allowable but also it is guaranteed that there is a path from the next switch to the destination. When a message arrives through one of the input channels, the routing unit determines one or several potential output channels to deliver the message. The routing decision is based on the relative position of the current and destination switches (i.e. within one of the following eight cases: North(N), South(S), East(E), West(W), Northeast(NE), Northwest(NW), Southeast(SE), and Southwest(SW)). All permissible output channels of HARA, for each pair of the input channel (Inch) and destination position (Pos) are shown in Table 1. As can be obtained from the table, HARA offers a large degree of adaptiveness to route messages. One of the drawbacks of non-minimal methods is the complexity of determining eligible output channels. However, as shown in Fig. 6 (i.e. that is extracted from Table 1), the implementation of HARA is relatively simple.

```
InCh: Input Channel; OutCh: Output Channel
Pos: Destination Position
----------------------------------------
IF Pos={L} THEN                      OutCh(L)<='1';
IF Pos={E or NE or SE} THEN          OutCh(E)<='1';
IF InCh={L or N1 or S1 or E} THEN    OutCh(W)<='1';
IF InCh={L or S1 or E} THEN          OutCh(N1)<='1';
IF InCh={L or N1 or E or S1} THEN    OutCh(S1)<='1';
IF (InCh/= {N2}) AND
   (Pos={N or E or NE or SE}) THEN OutCh(N2)<='1';
IF (InCh/={S2}) AND
   (Pos={S or E or NE or SE}) THEN OutCh(S2)<='1';
```

Fig. 6. Determining all eligible output channels by HARA

Theorem 1: HARA is deadlock-free.
Proof: If numbering mechanism ensures that all eligible turns are ordered in ascending order (descending order), no cyclic dependency can occur between channels. As can be observed from Fig. 5, all connections between input channels and output channels to form eligible turns in HARA take place in ascending order and thus HARA is deadlock free.

Theorem 2: HARA is livelock-free
Proof: In HARA, whenever a message transmits in the east direction, it can never be routed back in the west direction. Therefore, in the worst case, the message may reach to the leftmost column and then starts moving in the east direction toward the destination column. Therefore, after a limited number of hops, the message reaches the destination, and Theorem 2 is proved.

Fig. 7 shows an example of the HARA method in a 5×5 mesh network in which the source switch at (2,1) sends a message to the destination switch at (4,2). According to Table 1, the message arriving from the local channel and delivering toward the destination in the northeast position has six alternative choices (i.e. N1, N2, S1, S2, E, and W); among them, the output channels N1, N2, and E introduce the minimal paths and S1, S2, and W indicate the non-minimal paths. Since the neighboring switches in the minimal paths are in the congested area, the message is sent in a non-minimal direction that is not congested. Again, at the switch (2,0), all the minimal paths are congested, so the message is sent to switch (1,0) which is not congested. The same strategy is used until the message reaches the destination switch. This example shows the ability of the HARA method to reroute messages around congested areas.

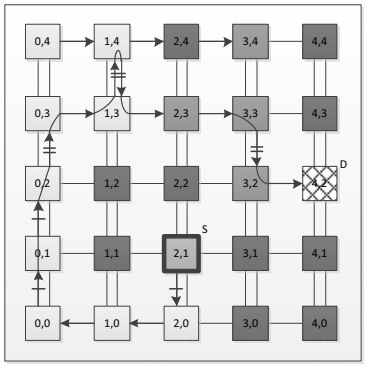

Fig. 7. An example of HARA

C. Q-Learning-based Output Selection Function

As mentioned earlier, we utilize an optimized Q-Routing model for the selection function of HARA to estimate the latency of sending a message from each output channel to the destination switch. Since the output selection function of HARA is inspired by the Q-Routing model, the proposed routing method is named HARAQ (HARA using Q-Routing). Let us explain the idea of HARAQ using the example of Fig. 8 where a message is generated at the source switch S for the destination D. According to HARA, when a message arrives from the local input channel and destined for a destination switch in the northeast position, six output channels can be selected to forward the message (i.e. N1, N2, S1, S2, E, and W). At Fig. 8(a), suppose that the colored entry of the Q-Table indicates the estimated latencies of sending a message from each possible output channel to the northeast region. Since output channel N1 has the lowest estimated latency, the message is delivered through this channel. At the switch X, the message is received by the input channel S1 (Fig. 8(b)). Using

the information in Table 1 or Fig. 6, multiple output channels can be used to forward the message (i.e. N1, N2, S1, S2, E, and W). Among eligible output channels, the output channel E has the lowest latency, and thus it is selected for sending the message to the switch Y. At this time, the local and global congestion values should be returned to the switch S. The time the message waited in the input buffer of the switch X before transmission to the switch Y is counted as the local information (i.e. B_X=1). The minimum estimated latency of routing messages from the switch X to the destination region via the neighboring switch Y is considered as the global latency and is extracted from the Q-Table of the switch X (i.e. $\min Q_X(D,Y)$=4). The summed value of the local and global information provides a new latency estimation of the path from the switch S to the destination D. Finally, the corresponding entry of the Q-Table at switch S (i.e. row: NE; column: N1) should be updated with the new value. This is done taking the average of the old and new latency estimation (Fig. 8(a)). At switch Y, the message is received via the west input channel (Fig. 8(c)). The output channel with the lowest latency is selected among the three possible output channels (i.e. N2, S2, and E). Upon connecting the input channel to the output channel of the switch Y, local and global information are returned to the switch X. The local congestion shows the waiting time of the message at the input buffer of switch Y (i.e. B_Y=3) while the global congestion indicates the estimated latency from the switch Y to the destination switch D via the neighboring switch Z (i.e. $\min Q_Y(D,Z)$=5). The sum of the local and global values is new latency estimation from the switch X to the destination switch. As shown in Fig. 8(b), the corresponding entry of the Q-Table at the switch X is updated taking an average of the new estimated value (i.e. $B_Y+\min Q_Y(D,Z)$) and an existing estimation ($Q_X(D,Y)$). Finally, the message arrives at the switch Z from the input channel S2 (Fig. 8(d)). This message can reach the destination by delivering it through N2 or E output channels. The output channel E has the lowest value and selected for routing the message. The local latency, the waiting period at the input buffer of the switch Z, is 3 while the global latency to the destination is equal to 0 as the message reaches the destination in the next hop. Similarly, the latency values are returned to switch Y and update the Q-table (Fig. 8(c)). Hence, as messages are propagated inside the network, Q-Tables gradually incorporate more global information [20].

Q-Table format: Q-Routing models learn the network condition at run time and based on the obtained information sends a message through the path that has the lowest estimated latency to the destination switch [6]. Generally, each switch maintains a Q-Table to store the estimated latency of routing messages to a destination switch via each output channel. Two typical types of Q-Table, named Q-Routing and C-Routing tables, are investigated in [23]. The size of a Q-Routing table is $n \times m \times k$ where n is the number of switches in the network, m is the number of output channels per switch, and k is the size of each entry in the Q-Table. The required area of Q-Routing tables not only is very large but also increases as the network size enlarges. C-Routing tables can decrease the size of tables by taking advantages of the clustering approach. The size of C-Routing tables is $(l+c) \times m \times k$ consisting of two parts: the cluster part having a size of $c \times m \times k$ where c is the number of clusters , and the local part having a size of $l \times m \times k$ where l is the number of switches within each cluster. The size of Q-Tables can be reduced by using C-Routing tables, but this model still suffers from the scalability issue since the size of C-Routing tables can become rather high as the network scales up. There are some other issues regarding the clustering model such as determining the size of each cluster for different network sizes or partitioning the network when the network size is not a multiple of the cluster size. The Q-Table in our model is named Region-based Routing (R-Routing); each row of this table corresponds to one of the eight different positions of the destination switch (i.e. N, S, E, W, NE, NW, SE, and SW) and each column indicates output channels (i.e. N1, N2,

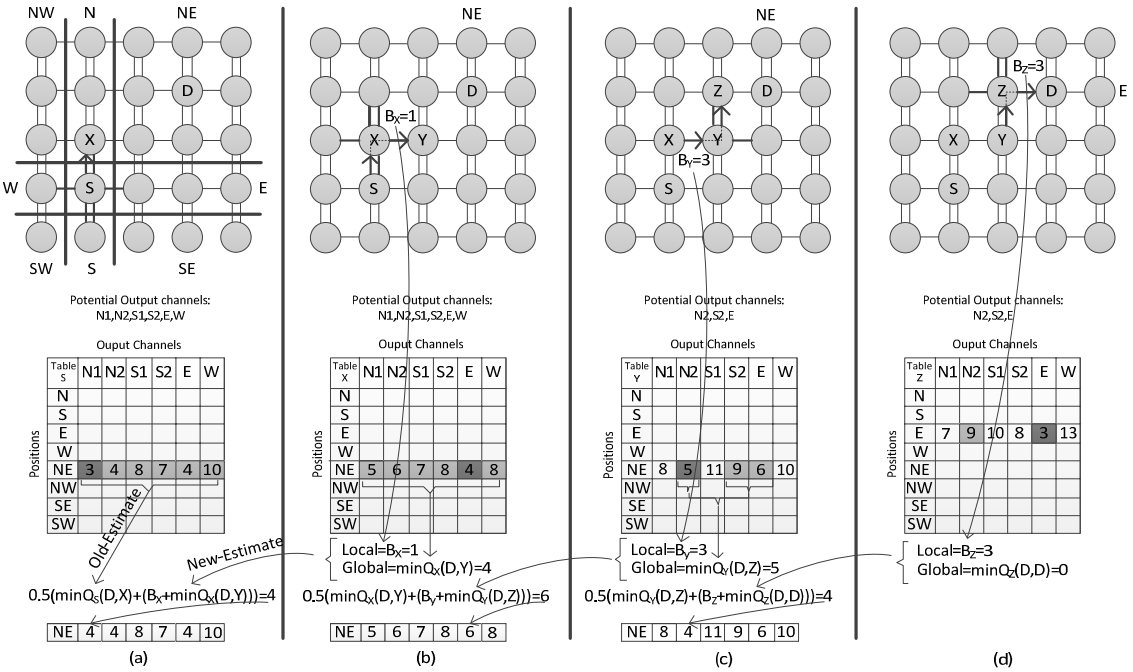

Fig. 8. The process of updating the Q-Tables

S1, S2, E, and W). Regardless of the network size, the size of R-Routing tables is 8×6×k that is considerably smaller than C-Routing and Q-Routing tables. The required size for each approach is given in Table 2 where l=4 and k=4. Note that the reported areas for Q-Routing and C-Routing tables are based on using no virtual channel in the network while R-Routing tables are based on utilizing an extra virtual channel. While our approach reduces the size of Q-Tables such that they can be applicable in NoCs, someone might think that the accuracy of the estimated latency toward each destination diminishes by our model. In fact, under real traffic conditions, each entry of the R-Routing table is inherently influenced by the switches which are in more communication with them at that period. Therefore, it is not necessary to allocate a row for each specific switch in the network. Moreover, R-Routing tables are updated more occasionally than C-Routing and Q-Routing tables since messages designated for the same regions can be used to update R-Routing tables while in two other models each entry is updated only by the messages for the same destination.

Table 2. the area overhead

Size\method	Q-Routing	C-Routing	R-Routing
8×8	128 bytes	40 bytes	24 bytes
16×16	512 bytes	64 bytes	24 bytes
32×32	2048 bytes	160 bytes	24 bytes

Transferring Local and Global Information: Q-Routing models are explored in on-chip networks but it is not obvious how the congestion is defined for local and global information and how many bits are considered for them. The congestion statuses are delivered over the channel whenever a message is transferred between two neighboring switches, which reduce the power consumption. In this work, a 4-bit congestion wire is used between each two neighboring switches to propagate local and global congestion information. The local congestion information is a 2-bit value indicating the waiting time of a message from when the header flit is accommodated in an input buffer until an output channel is dedicated to it. The pseudo code of Fig. 9 is used to encode the waiting time in 2-bit value. For example, if the waiting time of a message is less than the average size of three messages (i.e. waited for the service time of three messages), the local congestion can be encoded in "00". The global congestion information is a 4-bit value giving a global view of the latency from the output channel of the current switch to the destination switch region. This global information is extracted from the corresponding entry of R-Routing table. The Q-Values are updated whenever a message is propagated between two neighboring switches. Suppose that a message is sent from the switch X toward the destination switch D through the neighboring switch Y and then switch Z with the lowest estimated latencies. At the switch Y, upon connecting the input channel to the output channel, 2-bit local and 4-bit global values are aggregated into a 4-bit value (with the maximum value of "1111") and then transfer to the switch X. This value is a new estimation of the latency from the selected output channel of the switch X to the destination D. The corresponding entry of the Q-Table at the switch X is updated taking an average of the new estimated value (i.e. $B_Y+\min Q_Y(D,Z)$) and an existing estimation ($Q_X(D,Y)$). Commonly, in Q-Routing models, the following formula is used:

$$Q_X(D,Y) = (1-\propto)Q_X(D,Y)+\propto (B_Y + minQ_Y(D,Z))$$

In this formula, α represents the learning rate at which newer information overwrites the older one. A factor of 0 will make no learning while a factor of 1 would consider only the most recent information [22]. In our simulation, a 50-50 weight is assigned to old and new information so that α=0.5.

```
AMS: Average Message Size
LCV: Local Congestion Value
WT: Waiting Time
---------------------------
IF (WT ≤ 3×AMS) Then
   LCV<="00"; END IF;
ELSIF (WT ≤ 9×AMS) Then
   LCV<="01"; END IF;
ELSIF (WT ≤ 27×AMS) Then
   LCV<="10"; END IF;
ELSE LCV<="11"; END IF;
END IF;
```

Fig. 9. Encoding the waiting time into 2-bit

Table Initialization: Q-Routing models have an initial learning period during which it performs worse than minimal schemes. The reason is that there is a possibility of choosing non-minimal paths even if the network is not congested. To cope with this problem in the initialization phase, all entries of Q-Tables are initialized such that minimal output channels are set to "0000" and non-minimal output channels are set to "1000" and never can be less than it. Accordingly, in a low traffic condition, only minimal paths are selected while non-minimal paths are used to distribute traffic when the network gets congested.

IV. Results and Discussion

To evaluate the efficiency of the proposed routing scheme, DBAR and C-Routing schemes are also implemented. The former is an adaptive routing using local and non-local congestion information, while the latter is an adaptive and cluster-based routing using the Q-Learning technique. For fairness, DBAR and C-Routing utilize a fully adaptive routing function based on mad-y [12]. A wormhole-based NoC simulator is developed with VHDL to model all major components of the on-chip network and simulations are carried out to determine the latency characteristic of each network [25]. The message length is uniformly distributed between 1 and 5 flits. For all switches, the data width is set to 64 bits (the maximum bandwidth at each link is 1 flit per cycle) and each input channel has the buffer (FIFO) size of 8 flits. The request rate is defined as the ratio of the successful message injections into the network interface over the total number of injection attempts. The simulator is warmed up for 12,000 cycles and then the average performance is measured over another 200,000 cycles. Two synthetic traffic profiles including uniform random and hotspot, along with SPLASH-2 [26] application traces are used.

A. Performance Evaluation under Uniform Traffic Profile

In the uniform traffic profile, a switch sends a message to other switches with a uniform distribution. In Fig. 10, the average communication delay as a function of the average message injection rate is plotted for both mesh sizes. As observed from the results, in low loads, the Q-Routing schemes (HARAQ and C-Routing) behave as efficiently as DBAR. As load increases, DBAR is unable to tolerate the high load condition, while the Q-Routing schemes learn an efficient routing policy. HARAQ leads to the lowest latency due to the

fact that it can distribute traffic more efficiently than the other two schemes. In fact, in DBAR and C-Routing, messages use minimal paths so that under this traffic they are routed through the very center of the network which creates large permanent hotspots in the network. Correspondingly, messages traveling through the center of the network will be delayed much more than they would use any non-minimal paths. Due to the fact that the HARAQ method can reroute messages, it alleviates the congested areas and performs considerably better than other schemes. Using minimal and non-minimal routes along with the intelligent selection policy reduces the average network latency of HARAQ in the 8×8 network (near the saturation point, 0.3) about 18% and 37%, compared with C-Routing and DBAR, respectively.

B. Performance Evaluation under Hotspot Traffic Profile

Under the hotspot traffic profile, one or more switches are chosen as hotspots receiving an extra portion of the traffic in addition to the regular uniform traffic. In simulations, given a hotspot percentage of *H*, a newly generated message 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 performance of each network with *H = 10%* is illustrated in Fig. 11. As observed from the figure, the proposed routing scheme achieves better performance compared with the other schemes. In the 8×8 network, the performance gain near the saturation point (0.18) is about 31% and 42%, compared with C-Routing and DBAR, respectively. In addition, the impact of using non-minimal paths on the link utilization for the hotspot traffic profile is summarized in Table 3. We observe a reduction as large as 25% in the 8×8 network compared with DBAR. The results reveal that using the non-minimal scheme along with the Q-Learning policy can distribute the traffic efficiently.

Table 3. The impact of using non-minimal paths on the link utilization

		8×8	14×14
Maximum link utilization (flits/cycle)	DBAR	0.365	0.263
	C-Routing	0.352	0.256
	HARAQ	0.274	0.202
	Reduction over DBAR	25%	23%
	Reduction over CR	22%	21%

C. Application Traffic Profile

Application traces are obtained from the GEMS simulator [27] using some application benchmark suites selected from SPLASH-2. We use a 64-switch network configuration: 20 processors and 44 L2-cache memory modules. For the CPU, we assume a core similar to Sun Niagara and use SPARC ISA. Each L2 cache core is 512KB, and thus, the total shared L2 cache is 22MB. The memory hierarchy implemented is governed by a two-level directory cache coherence protocol. Each processor has a private write-back L1 cache (split L1 I and D cache, 64 KB, 2-way, 3-cycle access). The L2 cache is shared among all processors and split into banks (44 banks, 512 KB each for a total of 22 MB, 6-cycle bank access), connected via on-chip switches. The L1/L2 block size is 64B. Our coherence model includes a MESI-based protocol with distributed directories, with each L2 bank maintaining its own local directory. The simulated memory hierarchy mimics SNUCA [28] while the off-chip memory is a 4 GB DRAM with a 220-cycle access time. Fig. 12 shows the average message latency across four benchmark traces, normalized to DBAR. HARAQ provides lower latency than other schemes and it shows the greatest performance gain in Radix with 27% reduction in latency (vs. C-Routing). The average performance gain of HARAQ across all benchmarks is up to 22% vs. C-Routing and 33% vs. DBAR.

D. Hardware Analysis

To assess the area overhead and power consumption of the proposed scheme, the whole platform of each scheme is synthesized by Synopsys Design Compiler. Each scheme includes switches, communication channels, and congestion wires. For synthesis, we use the UMC 90nm technology at the operating frequency of 1GHz and supply voltage of 1V. We perform place-and-route, using Cadence Encounter, to have precise power and area estimations. The power dissipation of each scheme is calculated under the hotspot traffic profile near the saturation point (0.18) using Synopsys PrimePower in a 8×8 2D mesh. The layout area and power consumption of each platform are shown in Table 4. Comparing the area cost of the platform using HARAQ with the platforms using C-Routing and DBAR indicates that the C-Routing platform consumes more power and has a higher area overhead while the overhead of HARAQ platform compared with DBAR is less than 1% but with a significant performance gain. The HARAQ platform consumes more average power because of rerouting messages around the congestion areas which increases the hop counts. To illustrate how the presented approach reduces the network hotspots, the maximum power value of each platform is also reported in the table. The results indicate that the maximum power of the presented approach is 8% and 12% less than that of the DBAR and C-Routing platforms, respectively. This is achieved by smoothly distributing the power consumption over the network using the highly adaptive routing scheme which reduces the number of the hotspots. In fact, the maximum power values, reported in the table, belong to the switch designated as the hotspot one, (4, 4).

Table 4. Hardware implementation details

Network platforms	Area (mm^2)	Avg. Power (W) dynamic & static	Max. Power (W) dynamic & static
DBAR	6.791	2.41	3.33
C-Routing	6.954	2.52	3.46
HARAQ	6.822	2.81	3.06

V. Summary and Conclusion

In this paper, we proposed a highly adaptive routing algorithm based on minimal and non-minimal paths for on-chip networks. The presented algorithm provides a large number of paths for routing messages using only an extra virtual channel in the Y dimension. In the proposed method, messages can temporarily move away from the destination and be routed around congested regions. Moreover, the use of some 180-degree turns on a single channel is also permitted in the algorithm. To choose a less congested path, we have utilized an optimized and scalable learning model to estimate the latency from each output channel to the destination switch. In the learning model, switches maintain distributed tables to store the global congestion information from different regions of the network. This congestion information is collected via a fully distributed approach requiring a small number of bits per link. Finally, a less congested output channel is chosen by the selection function.

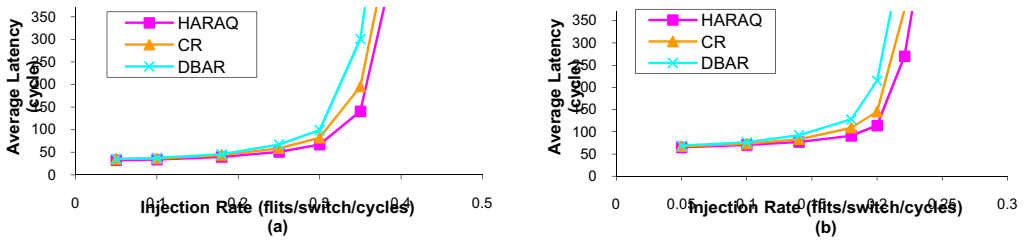

Fig. 10. Performance under different loads in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under the uniform traffic model

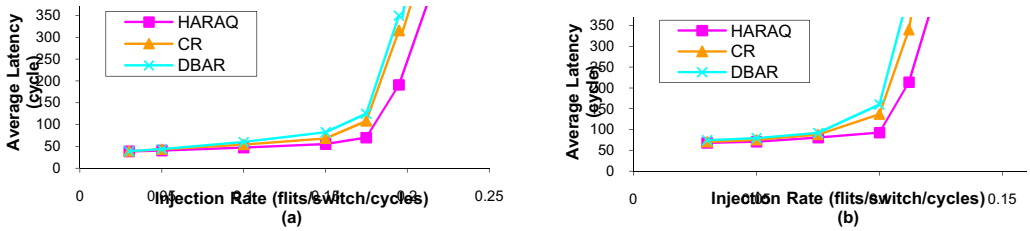

Fig. 11. Performance under different loads in (a) 8×8 2D-mesh and (b) 14×14 2D-mesh under hotspot traffic model with *H=10%*

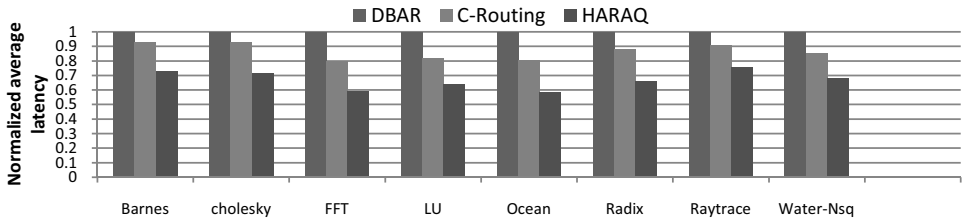

Fig. 12. Performance under different application benchmarks normalized to DBAR

References

[1] M. Daneshtalab, et al., "Memory-Efficient On-Chip Network with Adaptive Interfaces", IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, Vol. 31, No. 1, pp. 146-159, Jan 2012.

[2] X. Dai, et al., "An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control", In Proc. IEEE Transactions on Intelligent Transportation Systems, pp. 285-293, 2005.

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

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

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

[6] M. Majer, et al., "Packet Routing in Dynamically Changing Networks on Chip", in Proc. IPDPS, pp. 154b- 154b, 2005.

[7] M. Dehyadegari, et al., "An Adaptive Fuzzy Logic-based Routing Algorithm for Networks-on-Chip," in Proceedings of 13th IEEE/NASA-ESA International Conference on Adaptive Hardware and Systems (AHS), pp. 208-214, 2011.

[8] 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.

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

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

[11] M. Li, et al., "DyXY – a proximity congestion-aware deadlockfree dynamic routing method for network on chip", In Proc. of DAC, pp. 849-852, 2006.

[12] C. Glass and L. Ni, "Maximally Fully Adaptive Routing in 2D Meshes," In Proc. of Parallel Processing, pp.101-104, 1992.

[13] S.P. Kim, T. Han, "Fault-tolerant adaptive wormhole routing in 2D mesh," In Proc. of IEICE Trans. Inform, pp. 1064-1072, 1998.

[14] F. Chaix, et al., "A fault-tolerant deadlock-free adaptive routing for On Chip interconnects," in Proc. of DATE, pp. 1-4, 2011.

[15] A.A. Chien and J.H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors", in Proc. of Computer Architecture, pp. 268-277, 1992.

[16] Z. Zhang, et al., "A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip", in Proc. of DAC, 2008.

[17] S. Jovanovic, C. Tanougast, et al., "A new deadlock-free fault-tolerant routing algorithm for NoC interconnections", in Proc. of FPLA, pp.326-331, 2009.

[18] J. Wu and D. Wang, "Fault-tolerant and deadlock-free routing in 2-D meshes using rectilinear-monotone polygonal fault blocks", in Proc. of Parallel Algorithms Appl., pp.99-111, 2005.

[19] D. Fick, A. DeOrio, et al., "A Highly Resilient Routing Algorithm for FaultTolerant NoCs", in Proc. of DATE, pp. 21-26, 2009.

[20] 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.

[21] J.A.Boyan, M. L .Littman, "Packet routing in dynamically changing networks:A reinforcement learning approach", in Proc. of Advances in Neural Information, pp. 671-678, 1994.

[22] C. Feng, , et al. "A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip", in Proc. of NoCArc, pp.11-16 , 2010.

[23] M.K. Puthal, et al., "C-Routing: An adaptive hierarchical NoC routing methodology", in Proc. of VLSI-SoC, pp. 392-397, 2011.

[24] K.P. Fan, C.T. King, "Turn Grouping for Multicast in Wormhole-Routed Mesh Networks Supporting the Turn Model", Journal of Supercomputing, v.16, No.3, pp. 237-260, 2000.

[25] M. Daneshtalab, et al., "Adaptive Input-output Selection Based On-Chip Router Architecture", Journal of Low Power Electronics *(JOLPE)*, Vol. 8, No. 1, pp. 11-29, 2012

[26] S.C. Woo, et al., "The splash-2 programs: Characterization and methodological considerations", in Proc. of Computer Architecture (ISCA), pp. 24-36, 1995.

[27] M. K. Martin, et al. "Multifacet's general execution driven multiprocessor simulator (GEMS) toolset", SIGARCH Computer Architecture News, v. 33, No. 4, pp.92-99. November 2005.

[28] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches", in Proc. of the 37th annual IEEE/ACM International Symposium on MICRO, pp. 319–330, 2004.