Improved Route Selection Approaches using Q-learning framework for 2D NoCs

Niyati Gupta, Manoj Kumar, Ashish Sharma, Manoj Singh Gaur, Vijay Laxmi
MNIT Jaipur, India
(niyati.mnit, bohra.manoj1980, ashishsharma.fitt)@gmail.com, (gaurms,vlaxmi)@mnit.ac.in

Masoud Daneshtalab, Masoumeh Ebrahimi
University of Turku, Finland
(masdan, masebr)@utu.fi

ABSTRACT

With the emergence of large multi-core architectures, a volume of research has been focused on distributing traffic evenly over the whole network. However, increase in traffic density may lead to congestion and subsequently degrade the performance by increased latency in the network. In this paper, we propose two novel route selection strategies for on-chip networks which are based on the Q-learning framework. The proposed strategies use variable learning rate to dynamically capture the current congestion status of the network using an additional parameter and improves the learning process to select a less congested output channel. Both the proposed selection strategies are found to adapt significantly faster to the changes in traffic load and traffic patterns by avoiding congested areas. The results demonstrate that proposed strategies achieve significant performance improvement over conventional Q-routing and its variants with slight area-overhead.

Categories and Subject Descriptors

B.4 [**Input/Output and Data Communications**]: Miscellaneous

General Terms

Algorithms, Design, Performance

Keywords

On-Chip Networks, congestion, Q-learning, Q-routing

1. INTRODUCTION

System-on-chip (SoC) has evolved from single core to integrating tens or hundreds of cores on single chip. As chip area grows, the communication among cores becomes the performance hold-up in high computing multiprocessor systems on-chip (MPSoCs).

On-chip network (OCN) has resolved the performance and speed-up issues of traditional bus-based systems for real-time embedded applications. OCNs have been proven reliable, modular and reusable solutions [1] than shared buses. The cores communicate with each other by routing packets using a routing algorithm. The OCN performance depends highly on the routing algorithm which is the main focus area of our research.

Routing algorithms are classified as deterministic, oblivious and adaptive. Deterministic routing algorithms (say, XY) always route packets along the same path between the source and the destination nodes. For non-uniform traffic, they are unable to distribute the traffic load across the network [15]. However, oblivious and adaptive routing algorithms can forward packets along multiple paths. Oblivious routing selects routing path without considering the current state of the network.

In adaptive routing, path selection depends on the present network state to improve the network performance. The working of an adaptive routing algorithm is divided in two phases; routing function and selection function. The routing function first supplies the set of available output channels for a packet depending upon the source and the destination position. The selection function selects an output channel from the set of available output channels.

However, in traditional methods (Q-routing) updating the routing tables are limited to the traversing path only when the respective path is chosen by the packet towards the destination. As a result, the Q-values do not reflect the recent network condition and at times, use a Q-value which has not been updated over a long period of time, to make the selection decision. Moreover, Q-routing uses constant learning rate throughout to modify the Q-values irrespective of how recently a Q-value has been updated. In this paper we have tried to overcome it.

This paper makes the following primary contributions:

- A new selection mechanism, Credence based Q-routing (CrQ-routing) is proposed. As an improvement to Q-routing, CrQ-routing uses variable learning rate to improve the learning.
- We also present an extension of CrQ-routing called Probabilistic Credence based Q-routing (PCrQ-routing). CrQ-routing uses credence values (C-values) to measure the uncertainty in the corresponding Q-values and to compute the learning rate.

The rest of the paper is structured as follows: Section 2

briefs about existing research on congestion handling. Section 3 illustrates the background of the proposed algorithms. Sections 4 and 5 present the details of the proposed algorithms, viz. CrQ-routing and PCrQ-routing, respectively. Results are discussed in Section 6 and finally Section 7 presents the conclusions with future direction.

2. RELATED WORK

In recent years, several research proposals have been published to mitigate congestion in OCNs. DyXY [13] uses local information, i.e. the current queue length of the corresponding input port of the neighboring switches, to decide on the next hop. DyXY forwards packets through congested area as the utilized local information is insufficient. Most common implementations of routing algorithms, e.g. RCA [10], CATRA [5] and DBAR [14] have focused on collecting local or global congestion information to get an estimate of the congestion levels in the network. An agent-based congestion-aware technique [4] is presented to estimate congestion at cluster level and make the routing decision based on this estimation.

Q-Learning [18] based frameworks have been proposed firstly for off-chip networks [11, 12]. Later on, it has also been applied in OCN [6, 16, 9, 3, 8, 7]. In [6], a Q-learning based congestion-aware algorithm is proposed to find the congested links in the network using Q-values. C-routing [16] divides the mesh topology into clusters to reduce the routing table size as compared to conventional Q-routing algorithms. CQ-routing [9] also utilizes clustering approach to improve the network performance with reduced table area. HARAQ [3] presented a turn-model based non-minimal routing algorithm. The focus of the work was on reduction of the area of the Q-table by storing regional Q-values. Bi-LCQ [8] reduces the area overhead of the table by maintaining Q-values only for x and y directions. Along with this, it uses the clusters to improve its performance with both forward and backward exploration. Congestion-aware routing Algorithm using Dual Q-routing (CADuQ) [7] uses both data packets and learning packets to update the routing table. It also uses a new congestion detection metrics for updating local congestion information and is used to set the learning rate to keep the values as updated as possible. In contrast, when the switch is not congested the learning rate is set to a minimum value.

3. BACKGROUND

Since our proposed algorithms are based on Q-learning based routing approaches. So, we briefly describe Q-learning and Q-routing.

3.1 Q-LEARNING [18]

Q-learning [18] is effective and simple model-free reinforcement learning method to act optimally without the knowledge of the environment. It successively improvises the quality of the actions taken at particular states by getting rewards (states close to goal state) or punishments (states away from goal state). It uses training information which evaluates the actions taken rather than instructs by giving correct actions. An agent takes an action at a particular state and evaluates the results, in terms of immediate reward or penalty received in the next state to which it is taken. By trying all possible actions in all states (trial-and-error), it learns the best state by the evaluating the reward. Q-learning is a primitive form of learning and algorithms based on Q-Learning are highly adaptive and flexible.

The reward or punishment is known as Q-value and is saved in the Q-table at each state for all state-action pairs. Q-value is represented as $Q(s, a)$, which includes the expected long-term reward of taking action a in state s. When the system is in state s, it observes the environment and selects action a using the Q-values. By performing the action, it moves from the present state s to next state s'. After reaching s' it sends back reward or punishment (a real number) to state s.

There are two methods for selecting an action from the possible actions in every state [17] as below:

1. Exploration or non-greedy approach: Actions, which are not chosen yet but may produce greater total reward in the long run, are added to the table. Therefore, the agent chooses an action randomly (random selection).

2. Exploitation or greedy approach: Actions are selected according to the explored Q-table to maximize the expected reward (Q-table based).

3.2 Q-ROUTING [6]

Q-routing is a network routing algorithm based on the Q-learning. As the agent learns the model in Q-learning, similarly the decision maker (router) learns the network traffic levels in Q-routing. Q-routing learns to represent the network state in terms of Q-values ($Q_x(y, d)$) and uses these Q-values to make route selection decisions. These Q-values estimate and represent the quality of connected routes. The $Q_x(y, d)$ represent a Q-value of switch x for destination d for neighbor y. Router is able to select the least congested path among the available paths from a source to a destination using these values. These values can be used to select the best neighbor to route a packet; the best estimate is in terms of a performance metric (latency, in our case). In other words, Q-value is the measure of congestion in the network. A higher Q-value than other implies that the link is more loaded than the other link.

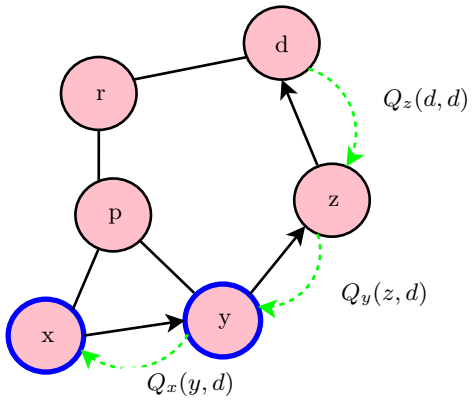


Figure 1: Example of Q-routing (solid arrows show data packets and dashed arrows show learning packets)

Figure 1 shows an example of Q-routing. Assume that a packet is to be routed from switch x towards the destination

d. There are two possible paths – (1) via switch p and (2) via switch y.

The Q-routing selects the neighbor having lower Q-value for the corresponding destination switch (d). The Q-value is updated each time the switch x sends a data packet to one of its neighbors (say y). The neighboring switch y, after receiving the packet, sends back a 'learning packet' to the switch x, as shown in Figure 1. This learning packet contains the estimated value $Q_x^{est}(y,d)$ (estimated latency or estimated congestion level) and is computed as:

$$Q_x^{est}(y,d) = Q_y(\widehat{z},d) + q_y \quad (1)$$

where, $Q_y(\widehat{z},d) = \min_{n \in N(y)} Q_y(n,d)$ (represents global congestion value) and $N(y)$ is the set of neighbors of switch y, and q_y (represents local congestion) is the waiting time in the input buffer of the switch y.

After receiving the estimated value, the switch x computes new Q-value $Q_x(y,d)$ and updates it as follows:

$$Q_x^{new}(y,d) = Q_x^{old}(y,d) + \gamma\ (Q_x^{est}(y,d) - Q_x^{old}(y,d)) \quad (2)$$

where, $Q_x^{new}(y,d)$ is the updated Q-value at switch x for destination d, $Q_x^{old}(y,d)$ is the old Q-value.

The $Q_x^{est}(y,d)$ is the best (smallest) estimated delay incurred in forwarding a packet to destination d from switch x via neighbor y.

Equation 2 also uses a learning rate (γ) to perform learning of the network state by updating the Q-values. Learning rate (γ) determines the rate at which the new information overrides the old one. Learning rate can take a value between zero and one. The learning rate of zero indicates that no learning is made by the algorithm while the learning rate of one means that Q-value is the most learned value.

Although, Q-routing is congestion aware routing but it suffers from the hysteresis problem [2] i.e. the network fails to adapt to the shortest (optimal) path after a period of increased network traffic. Once a packet takes the longer path due to increased traffic it may learn to adapt to this path in absence of packet forwarding through any minimal path. As a result, it continues to use this as a best path even when the traffic load reduces, unless and until the shortest path is selected and the Q-value is updated accordingly, the subsequent decrease in the link load remains unnoticed.

4. CrQ-ROUTING

In this section, we present Credence based Q-routing (CrQ-routing) selection mechanism. Q-routing does not have any parameter to specify how recently a Q-value has been revised. Moreover, a constant learning rate is applied throughout by Q-routing to modify the Q-values irrespective of how recent a Q-value has been updated. As an improvement to Q-routing, CrQ-routing uses variable learning rate (computed using confidence in the Q-value) to make the Q-value updates more efficient.

4.1 C-values

The performance CrQ-routing is enhanced by adding an extra parameter "Credence" to each of the Q-values in the CrQ-routing algorithm. A value $C_x(y,d)$ called Credence value (C-value), is associated with each and every Q-value ($Q_x(y,d)$) in the network. Credence value is a real number between 1 and 10. This new parameter represents how recently a Q-value has been updated or modified, thereby ensuring the freshness of Q-values. If the Q-value is not updated over a long period of time, this stale value does not truly represent the current state of the network. The C-value equal to 10 or close to 10 implies high confidence in the respective Q-value. It means that the Q-value has been recently updated, thus more reliable to be used in making the selection decision. However, the C-value equal to 1 or close to 1 implies that the respective Q-value has not been updated recently (unreliable). The credence value is the quantified freshness and accuracy measurement of a Q-value.

C-values are also used in determining the learning rate (γ) for the Q-value and the C-value update. C-values are updated in a reflective manner of how closely the corresponding Q-value represents the current state of the network. The learning rate γ is high if either:

- Confidence in old Q-value is low, or
- Confidence in new Q-value is high.

If the learning rate is high, the confidence in the updated Q-value is closer to that of the estimated Q-value′s confidence. Also, when a Q-value with low credence value is to be updated, it is better to update this Q-value with a higher learning rate.

The Q-values which are not updated recently do not reflect the current network traffic levels. So, the confidence in these Q-values should go down. Hence, C-value is reduced by 1 if the corresponding Q-value is not updated. On the other hand, every update of Q-value is associated with a corresponding update of C-value.

4.2 Learning packet

The proposed algorithm uses two types of packets in the network namely, data packets and learning packets. It uses separate virtual channels to propagate the packets. The learning packet is generated when the data packet reaches the next hop. As shown in Figure 2, it has four main fields described as follows:

Direction (1-bit)	Estimated Latency (6-bits)	C-value (4-bits)	Destination Switch (6-bits)

Figure 2: Learning Packet Format

Direction: It stores the direction of sender switch of the learning packet. It is 0 for x-direction and 1 for y-direction.

Estimated Latency: It is the sum of local (waiting time in the input buffer) and global (Q-value) congestion values.

C-value: It is the C-value associated with the chosen Q-value in the above step.

Destination Switch: It contains the destination switch of the data packet. However, for a large mesh network the field size can be increased.

4.3 Initialization of Q-values and C-values

Q-learning based algorithms have an initial learning duration. Due to the fact that even if the network is not congested it may not choose minimal paths for all the data packets. The solution is that for all the minimal paths the Q-values should be initialized as "000000" and non-minimal entries are set to "100000". Therefore, when the network is not congested, minimal paths are selected (due to low Q-values) and when the network traffic increases packets may

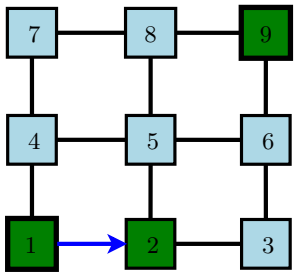


Table of Switch 1		
d	**x-dir**	**y-dir**
1	...	...
2	...	...
3	...	...
4	...	...
5	...	...
6	...	...
7	...	...
8	...	...
9	**3/2**	3/2

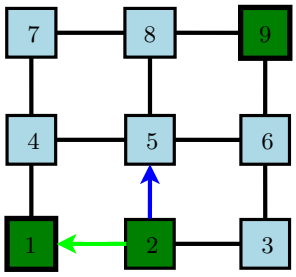


Table of Switch 2		
d	**x-dir**	**y-dir**
1	...	...
2	...	...
3	...	...
4	...	...
5	...	...
6	...	...
7	...	...
8	...	...
9	6/3	**2/2**

Learning rate $(\gamma) = 0.1 * \max\{(10-2), 2\} = 0.8$

Updated values after receiving learning packet from switch 2

$$\begin{aligned} Q_1(2,9) &= 3 + 0.8(2+0-3) = 2 \\ C_1(2,9) &= 2 + 0.8(2-2) = 2 \\ C_1(4,9) &= 2 - 1 = 1 \end{aligned}$$

Figure 3: Example of CrQ-routing (Q-value/C-value of Switch 1 and Switch 2 (green arrow shows learning packet and blue arrow shows data packet))

be routed via non-minimal paths. Initially, the corresponding C-values are set to 1, as the network is not learned yet.

4.4 CrQ-routing

When the switch x sends a data packet to its neighbor y by selecting the direction with smaller Q-value,

1. Switch y extracts the destination d from the data packet. It looks into its Q-table for the destination d and directions (x and y). It selects the next hop neighbor $\widehat{z}$ using Equation 3.

$$Q_y(\widehat{z}, d) = \min_{n \in N(y)} Q_y(n, d) \tag{3}$$

   where $N(y)$ is a set of neighbors of switch y.

2. Switch y computes best estimated latency (congestion level) $Q_x^{est}(y, d)$ using Equation 4.

$$Q_x^{est}(y, d) = Q_y(\widehat{z}, d) + q_y \tag{4}$$

   where $Q_y(\widehat{z}, d)$ represents the next neighbor latency (global congestion information from distant switches) and q_y is the waiting time in the input buffer of switch y (local congestion information of the input port).

3. This estimated Q-value $Q_y(\widehat{z}, d)$ along with its C-value $C_y(\widehat{z}, d)$ is encapsulated in the learning packet and is send to the switch x.

4. After receiving the learning packet, switch x extracts the estimated C-value $C_y(\widehat{z}, d)$ and computes the learning rate γ using Equation 5,

$$\gamma = 0.1 * \max(C_y(\widehat{z}, d), 10 - C_x(y, d)) \tag{5}$$

   where, $C_x(y, d)$ is previous C-value at switch x corresponding to switch y and destination d.

5. The Q-value is updated using Equation 6

$$Q_x^{new}(y, d) = Q_x^{old}(y, d) + \gamma \left[(Q_y(\hat{z}, d) + q_y) - Q_x^{old}(y, d) \right] \tag{6}$$

   Using the same learning rate (γ), the C-value associated with the above Q-value is also updated. The learning rate (γ) lies between 0 to 1.

$$C_x^{new}(y, d) = C_x^{old}(y, d) + \gamma \left[(C_y(\hat{z}, d)) - C_x^{old}(y, d) \right] \tag{7}$$

6. The C-value (for d^{th} row) corresponding to neighbor other than y, is decremented by 1.

$$C_x(y', d) = C_x(y', d) - 1 \tag{8}$$

   where y' is all neighbors of x except y.

It should be noted that any update in C-value or Q-value is subject to the value lying within their respective range and their values are rounded off, if the updates give decimal values.

Hence, CrQ-routing is able to adapt itself to the changing network scenario by adding confidence measures in the Q-value update rule. The variable learning rate has improved the quality of our proposed work in two ways:

- when the Q-values are unreliable, they are updated more and,
- when the new estimates have high confidence, they have a larger effect.

Figures 3-6 show an instance of CrQ-Routing in which a data packet is to be routed from switch 1 to destination switch 9. At switch 1 (as shown in Figure 3), the data packet can choose any direction (x or y) whichever has smaller Q-value. Since, the Q-values for destination 9 are same for both x and y directions, CrQ-routing selects any one direction randomly (say x) and forwards the packet in that direction. When switch 2 receives this data packet, it selects switch 5 as the next hop using Equation 3 and computes the best estimated latency (2) for this next hop neighbor using Equation 4. Switch 2 sends the estimated latency (next neighbor congestion level is 2 and waiting time in the input buffer of switch 2 is 0) along with credence value (2) encapsulated in the learning packet to switch 1. After receiving the learning packet, switch 1 calculates the learning rate (γ) using Equation 5 and updates the Q-values and C-values using Equations 6 and 7 respectively, as shown in the Figure 3.

When the data packet reaches switch 5, it selects x-direction to forward the packet because it has lower Q-value

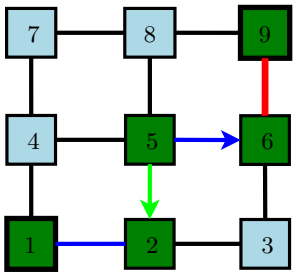


Table of Switch 5		
d	**x-dir**	**y-dir**
1	...	...
2	...	...
3	...	...
4	...	...
5	...	...
6	...	...
7	...	...
8	...	...
9	**5/5**	7/3

Learning rate $(\gamma) = 0.1 * \max\{(10-2), 5\} = 0.8$

Updated values after receiving learning packet from switch 5

$$\begin{aligned} Q_2(5,9) &= 2 + 0.8(5+2-2) = 6 \\ C_2(5,9) &= 2 + 0.8(5-2) = 4 \\ C_2(3,9) &= 3 - 1 = 2 \end{aligned}$$

Figure 4: Q-value/C-value of Switch 5 (red line shows exploratory path)

than the other and forms the learning packet. As can be seen in Figure 4, switch 5 sends the learning packet to switch 2 and the data packet to switch 6 simultaneously. Switch 2 extracts the estimated latency (Q-value) as 7 (next neighbor congestion level is 5 and waiting time in the input buffer of switch 5 is 2) and corresponding C-value as 5 from the learning packet. Using these values, it calculates γ and updates Q-value and C-value accordingly.

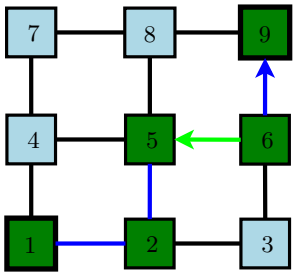


Table of Switch 6		
d	**x-dir**	**y-dir**
1	...	...
2	...	...
3	...	...
4	...	...
5	...	...
6	...	...
7	...	...
8	...	...
9	10/6	**1/5**

Learning rate $(\gamma) = 0.1 * \max\{(10-5), 5\} = 0.5$

Updated values after receiving learning packet from switch 6

$$\begin{aligned} Q_5(6,9) &= 5 + 0.5(1+4-5) = 5 \\ C_5(6,9) &= 5 + 0.5(5-5) = 5 \\ C_5(8,9) &= 3 - 1 = 2 \end{aligned}$$

Figure 5: Q-value/C-value of Switch 6

As shown in Figure 5, switch 5 also receives the learning packet (next neighbor congestion level is 1, waiting time is 4 and credence value is 5) from its neighbor (switch 6). It updates the Q-value to 5 and the C-value to 5 using the learning rate of 0.5. The C-value which is not associated with Q-value update, is reduced to 2.

Similarly (Figure 6), when switch 9 receives and processes the data packet sent by switch 6, it sends the learning packet to switch 6. Switch 6 updates its Q-value to 4 and C-value to 10 at the learning rate of 1.

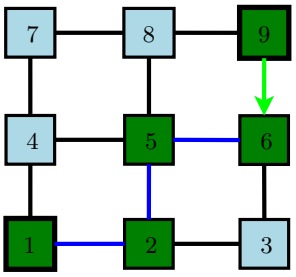


Table of Switch 9		
d	**x-dir**	**y-dir**
1	...	...
2	...	...
3	...	...
4	...	...
5	...	...
6	...	...
7	...	...
8	...	...
9	0/10	0/10

Learning rate $(\gamma) = 0.1 * \max\{(10-5), 10\} = 1$

Updated values after receiving learning packet from switch 9

$$\begin{aligned} Q_6(9,9) &= 1 + 1(4+0-1) = 4 \\ C_6(9,9) &= 5 + 1(10-5) = 10 \\ C_6(5,9) &= 6 - 1 = 5 \end{aligned}$$

Figure 6: Q-value/C-value of Switch 9

5. PCrQ-ROUTING

In this section, we present an extension of CrQ-routing called Probabilistic Credence based Q-routing (PCrQ-routing). CrQ-routing uses credence values (C-values) to measure the uncertainty in the corresponding Q-values and to compute the learning rate. Although, CrQ-routing improves the performance of the network in comparison with other Q-routing algorithms using variable learning rate but these C-values are not used in the exploring Q-values. As a result, it may, at times, use a Q-value, which has not been updated over a long period of time, to make the selection decision which in turn increases congestion. Therefore, using random selection decisions sometimes are useful in reverting back to the optimal selection policy when the network traffic condition is recovered. Also, the network is able to effectively adapt itself once the normal conditions are restored.

When the switch x sends a data packet to its neighbor y,

1. Switch y extracts the destination d from the data packet. For the destination d, it computes the variance for each Q-value using corresponding C-value. Let σ^2 be the variance function and is computed using Equation 9:

$$\sigma^2 = \frac{1}{C_y(n,d)} * k, \forall n \in N(y) \tag{9}$$

where, $N(y)$ is a set of neighbors of switch y, k is a user defined parameter that defines rate of change of Q-value. For optimal results, we have used k as 0.2.

2. Using the respective variance (σ^2) of each Q-value for destination d, new Q-value is calculated at switch y using Equation 10:

$$Q'_y(n,d) = (1-\sigma^2)Q_y(n,d), \forall n \in N(y) \tag{10}$$

3. Switch y selects the next hop neighbor $\hat{z}$ using Equation 11:

$$Q'_y(\hat{z},d) = \min_{n \in N(y)} Q'_y(n,d) \tag{11}$$

where $Q'_y(n,d)$ is the Q-value calculated in step 2.

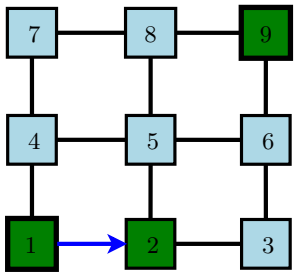


Table of Switch 1		
d	**x-dir**	**y-dir**
1	...	...
2	...	...
3	...	...
4	...	...
5	...	...
6	...	...
7	...	...
8	...	...
9	**12/6**	13/8

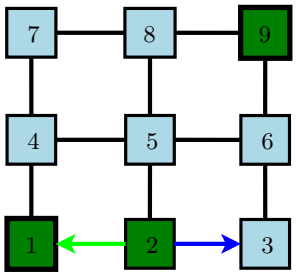


Table of Switch 2		
d	**x-dir**	**y-dir**
1	...	...
2	...	...
3	...	...
4	...	...
5	...	...
6	...	...
7	...	...
8	...	...
9	17/9	20/1

$$Q'_2(5,9) = (1-0.02)17 = 17 \qquad Q'_2(3,9) = (1-0.2)20 = \mathbf{16}$$

$$\textit{Learning rate } (\gamma) = 0.1 * \max\{1,(10-6)\} = 0.4$$

Updated values after receiving learning packet from switch 2

$$\begin{aligned} Q_1(2,9) &= 12 + 0.4(16+4-12) = 15 \\ C_1(2,9) &= 6 + 0.4(1-6) = 4 \\ C_1(4,9) &= 8 - 1 = 7 \end{aligned}$$

Figure 7: Example of PCrQ-routing (Q-value/C-value of Switch 1 and Switch 2 (green arrow shows learning packet and blue arrow shows data packet))

4. Switch y computes the best estimated latency $Q_x^{est}(y,d)$ using Equation 12:

$$Q_x^{est}(y,d) = Q'_y(\hat{z},d) + q_y \tag{12}$$

5. This estimated latency $Q_x^{est}(y,d)$ along with the associated C-value $C_y(\hat{z},d)$ is encapsulated in the learning packet and is sent to the switch x.

6. PCrQ-routing further follows the same steps as followed by CrQ-routing (steps 4 to 6).

When the confidence in a Q-value is high, the corresponding variance is low (Equation 9). Thus, it can be observed from Equation 10 that the new Q-value ($Q'(n,d)$) is very close to the previous Q-value ($Q(n,d)$). This illustrates the fact that the previous Q-value represents the present state of the network.

Similarly, when the confidence in a Q-value is low, variance is high and the new Q-value ($Q'(n,d)$) is far from the previous Q-value ($Q(n,d)$). It means that the previous Q-value ($Q(n,d)$) does not represent the present network status, thus a new Q-value ($Q'(n,d)$) should be preferred. CrQ-routing may not choose a path which was earlier more congested (high Q-value and C-value) than the alternate path (less Q-value), but now has become normal (still has same Q-value but low C-value) and the alternate route has become congested (but Q-value is still less than the other and high C-value). PCrQ-routing chooses the path according to the current state of the network as it calculates new Q-values ($Q'(n,d)$) using the confidence values. Hence, it adapts the present network in more efficient manner.

Figure 7 depicts an instance of the PCrQ-routing which routes the data packet from switch 1 to switch 9. Switch 1 sends the data packet to switch 2 via x-direction. When switch 2 processes this packet, it computes the variance and new Q-value using Equation 9 and 10 respectively. The newly computed Q-values for x and y directions are 16 and 17 respectively. PCrQ-routing chooses the next hop neighbor as switch 3 due to lower Q-value (new). However, CrQ-routing would have chosen y-direction for routing the data packet as it has minimum Q-value of 17 in the table (without any updates). Switch 2 sends the new Q-value (16) plus waiting time in the input buffer of switch 2 (4) and its associated C-value (1) in the learning packet to switch 1. Meanwhile, this data packet is forwarded to switch 3. Switch 1 extracts the fields from the learning packet and calculates the learning rate as 0.4 and thereafter, updates its Q-value $Q_1(2,9)$ to 15 and C-value $C_1(2,9)$ to 4. The other C-value $C_1(4,9)$ which is not updated in the last step is decremented by 1. The same procedure is followed by other the intermediate switches until the packet reaches the destination.

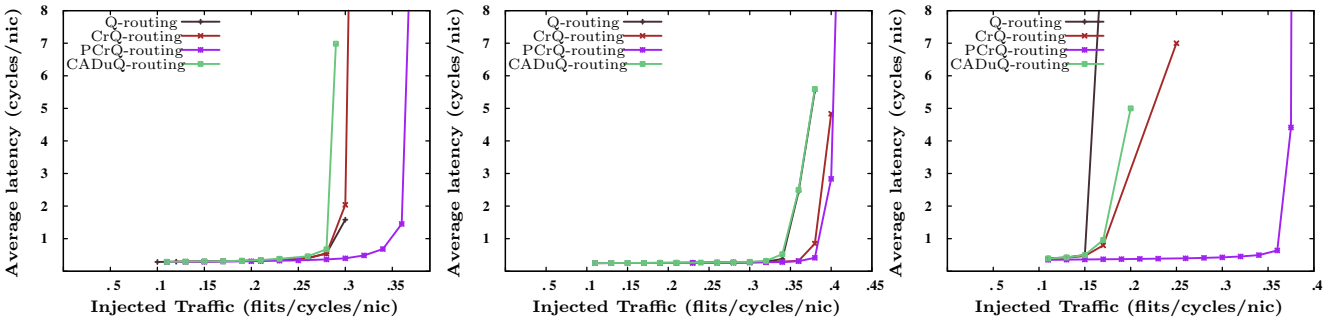
In short, PCrQ-routing is an extension of CrQ-routing in such a way that the confidence values can be used to explore and exploit route selection decisions. However, the usage of new Q-values leads to exploration of new paths as the earlier congested link might be back to normal now and as a result, might lead to more adaption in the network.

6. EXPERIMENTAL SETUP AND RESULT ANALYSIS

In this section, we analyze the performance achieved by the proposed algorithms when compared with the competing routing algorithms. We use an in-house C based OCN simulator to implement all the algorithms.

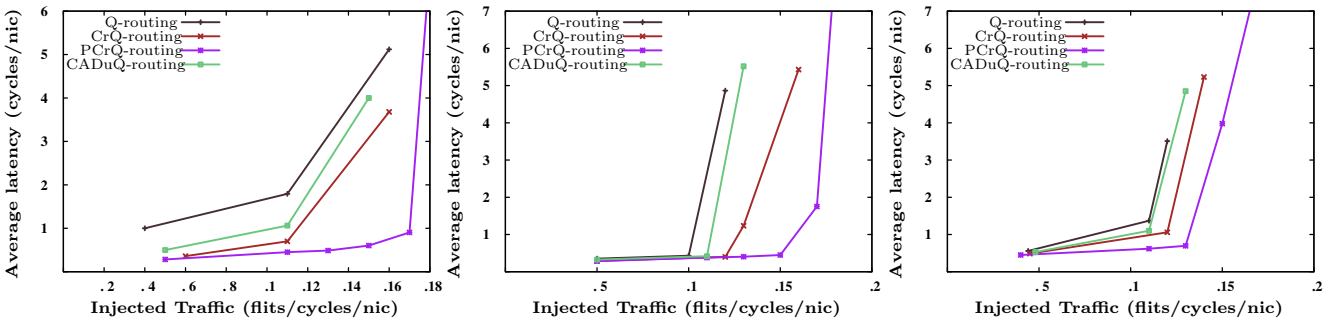
6.1 Experimental Setup

For our simulations, we use 4-stage pipelined router: IB (storing data into the buffer), RT (route computation), VA/SA (VC allocation/switch allocation) and X (crossbar). The router uses wormhole flow-control with flit-level crossbar switching. The size of the router queues is 6 flits. We use regular 4×4 and 8×8 mesh networks for the analysis. We use two virtual networks with one virtual channel each. To propagate data packets, one virtual network is used along each dimension, while for learning packets, one separate virtual network is utilized. Messages are 32-flits long with a



(a) Uniform Traffic **(b)** Shuffle Traffic **(c)** Bit-Complement Traffic

Figure 8: Average Latency for different traffic patterns for 4×4 mesh



(a) Uniform Traffic **(b)** Shuffle Traffic **(c)** Bit-Complement Traffic

Figure 9: Average Latency for different traffic patterns for 8×8 mesh

flit size of 4 bytes. We train all the algorithms for 12000 messages and then perform the testing using another 20000 messages. We use west-first routing algorithm as the underlined routing algorithm. The metric used for measuring the performance is the latency. Latency is defined as the time duration when the head flit is injected at the source core to the time when the tail flit is delivered to the destination core. We analyze CrQ-routing and PCrQ-routing for both uniform (random) and non-uniform traffic (shuffle and bit-complement) patterns.

6.2 Performance Evaluation

6.2.1 *Traffic Patterns*

In this section, we evaluate the latency results for the different traffic patterns. Figures 8 and 9 show the average end-to-end latency for 4×4 and 8×8 mesh sizes. In the uniform traffic pattern, a switch sends a packet to other switches with a uniform probability distribution.

We can observe from Figure 8 that for low flit-injection rate all the algorithms almost exhibit low latency. Later on, as the flit-injection rate increases both CrQ-routing and PCrQ-routing perform better than other algorithms for uniform, shuffle and bit-complement traffic patterns due to the fair load distribution in the mesh. It should be noted that PCrQ-routing performs better than CrQ-routing as former uses random Q-values which improves its adaption and performance in comparison to CrQ-routing and other algorithms.

Similarly in Figure 9 for low and medium traffic loads, we observe that the Q-routing schemes (Q-routing, CADuQ) behave same as CrQ-routing and PCrQ-routing. As the load increases, Q-routing variants are unable to tolerate the high traffic load condition, while PCrQ-routing outperforms by learning the network efficiently. CrQ-routing also performs better than other routing schemes (except PCrQ-routing). PCrQ-routing leads to the lowest latency as it increases adaptability manifold and distributes traffic more efficiently than the competing routing algorithms. In Q-routing, CADuQ and CrQ-routing, packets use a Q-value which has not been updated over a long time (low C-value). But PCrQ-routing performs considerably better than other variants as the selection decision uses the C-value to capture the congestion precisely. Hence, it quickly adapts the network and alleviates congestion in the network. The CrQ-routing and PCrQ-routing algorithms saturate at higher traffic load levels as compared to other algorithms.

6.2.2 *Area Analysis*

Table 1 shows the area requirement for different mesh sizes for all the algorithms. It should be noted that the size of Q-table is calculated using $n \times m \times k$, where n is number of switches, m is number of output channels and k is the size of each entry in the table. For Q-routing and CADuQ-routing, the value of k used is 6 bits, while CrQ-routing and PCrQ-routing uses k as 10 bits (6 bits for Q-value and 4 bits for C-value). We observe that the proposed algorithms have an acceptable area overhead.

Table 1: Area of Q-tables for Different Mesh Sizes

Algorithms/Size	Q-routing [9]	CADuQ [7]	CrQ-routing	PCrQ-routing
5×5	150 bytes	150 bytes	250 bytes	250 bytes
10×10	600 bytes	600 bytes	1000 bytes	1000 bytes
15×15	1350 bytes	1350 bytes	2250 bytes	2250 bytes

6.3 Implementation Analysis

To estimate the hardware cost of all algorithms, we have implemented routing logic of each algorithm with Verilog and synthesized with FPGA xilinx vivado tool for kintex-7 board with device xc7k325t-fbg900. The device area utilization and power consumption for each scheme is shown in Table 2. It can be observed that the proposed algorithms consume slightly more power with small area overhead

Table 2: Area and Power Analysis

Algorithms	Area (in LUTs)	Power (in Watt)
Q-routing [9]	525	0.152
CADuQ [7]	760	0.155
CrQ-routing	860	0.156
PCrQ-routing	988	0.158

7. CONCLUSIONS AND FUTURE WORK

In the existing Q-learning based approaches, some non-congested paths are avoided by the packets as their Q-values do not reflect the current network scenario. Thus, the selection decisions can be inaccurate on these stale Q-values as they do not represent the current congestion state of the network. To address this issue, we have proposed techniques to adjust the learning rate with the network condition (CrQ-routing) and to select a less congested output channel on the basis of its reliability (PCrQ-routing). The results show that the proposed algorithms are able to route packets more effectively by alleviating congestion and achieve significant performance improvement over traditional Q-learning based algorithms with slight area and power overheads. In future, we plan to reduce the area overhead at the switches.

8. REFERENCES

[1] L. Benini and G. De Micheli. Networks on chips: A new soc paradigm. *Computer*, 35(1):70–78, 2002.

[2] S. Choi and D.-Y. Yeung. Predictive q-routing: A memory-based reinforcement learning approach to adaptive traffic control. *Advances in Neural Information Processing Systems*, 8:945–951, 1996.

[3] M. Ebrahimi, M. Daneshtalab, F. Farahnakian, J. Plosila, P. Liljeberg, M. Palesi, and H. Tenhunen. HARAQ: congestion-aware learning model for highly adaptive routing algorithm in on-chip networks. In *Proceedings of the 6th IEEE International Symposium on Networks-on-Chip*, pages 19–26, 2012.

[4] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen. Agent-based on-chip network using efficient selection method. In *Proceedings of the 19th International Conference on VLSI and System-on-Chip*, pages 284–289, 2011.

[5] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen. CATRA- congestion aware trapezoid-based routing algorithm for on-chip networks. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 320–325, 2012.

[6] 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 the 2nd IEEE International Conference on Networked Embedded Systems for Enterprise Applications*, pages 1–7, 2011.

[7] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila. Adaptive load balancing in learning-based approaches for many-core embedded systems. *The Journal of Supercomputing*, 68(3):1214–1234, 2014.

[8] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila. Bi-lcq: A low-weight clustering-based q-learning approach for nocs. *Microprocessors and Microsystems - Embedded Hardware Design*, 38(1):64–75, 2014.

[9] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, and P. Liljeberg. Optimized q-learning model for distributing traffic in on-chip networks. In *Proceedings of the 3rd IEEE International Conference on Networked Embedded Systems for Every Application*, pages 1–8, 2012.

[10] P. Gratz, B. Grot, and S. W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *IEEE 14th International Symposium on High Performance Computer Architecture*, pages 203–214, 2008.

[11] S. Kumar and R. Miikkulainen. Dual reinforcement q-routing: An on-line adaptive routing algorithm. In *Artificial neural networks in engineering*, 1997.

[12] S. Kumar and R. Miikkulainen. Confidence based dual reinforcement q-routing: An adaptive online network routing algorithm. In *IJCAI*, volume 99, pages 758–763, 1999.

[13] M. Li, Q.-A. Zeng, and W.-B. Jone. DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *Proceedings of the 43rd Design Automation Conference*, pages 849–852. ACM, 2006.

[14] S. Ma, N. E. Jerger, and Z. Wang. Dbar: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip. In *Proceedings of the 38th IEEE International Symposium on Computer Architecture*, pages 413–424, 2011.

[15] M. Palesi and M. Daneshtalab. *Routing Algorithms in Networks-on-Chip*. Springer Publishing Company, Incorporated, 2013.

[16] M. K. Puthal, V. Singh, M. S. Gaur, and V. Laxmi. C-routing: An adaptive hierarchical noc routing methodology. In *Proceedings of the 19th International Conference on VLSI and System-on-Chip*, pages 392–397, 2011.

[17] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[18] C. Watkins. Q-learning. Technical report, University of Cambridge, England, 1992.