A Traffic-Balanced Adaptive Wormhole Routing Scheme for Two-Dimensional Meshes [1]

Jatin Upadhyay, Vara Varavithya, and Prasant Mohapatra

Department of Electrical and Computer Engineering
201 Coover Hall
Iowa State University
Ames, IA 50011

Abstract

In this paper, we have analyzed several issues that are involved in developing low latency adaptive wormhole routing schemes for two-dimensional meshes. It is observed that along with adaptivity, balanced distribution of traffic has a significant impact on the system performance. Motivated by this observation, we have developed a new fully adaptive routing algorithm called *positive-first-negative-first* for two-dimensional meshes. The algorithm uses only two virtual channels per physical channel creating two virtual networks. The messages are routed positive-first in one virtual network and negative-first in the other. Because of this combination, the algorithm distributes the system load uniformly throughout the network and is also fully adaptive. It is shown that the proposed algorithm results in providing better performance in terms of the average network latency and throughput when compared with the previously proposed routing algorithms.

Index: Adaptive wormhole routing, Positive-First-Negative-First algorithm, Region of adaptivity, Traffic distribution, Two-dimensional mesh.

[1]This paper is a revised version of "Efficient and Balanced Adaptive Routing in Two-Dimensional Meshes, " by Jatin Upadhyay, Vara Varavithya, and Prasant Mohapatra which appeared in *International Symposium on High Performance Computer Architecture*, Raleigh, North Carolina, January 1995, pp. 112-121.

Index:

- Positive-First-Negative-First algorithm
- Region of adaptivity
- Traffic distribution
- Two-dimensional mesh

1 Introduction

In distributed parallel computers, tasks are executed by a set of intercommunicating nodes or processors. The communication is usually carried out by means of passing messages from one node to another over the interconnection network. Since direct networks utilize the locality of the message references more efficiently, most of the existing systems use direct networks such as k-ary n-cube or n-dimensional mesh.

The performance of the interprocessor communication scheme depends largely on the network dimension, the switching technique, and the routing algorithm. Most of the contemporary systems have used two or three dimensions. Store and forward, virtual cut-though and wormhole routing are the main switching techniques used for interprocessor communication. Due to lower latency and small buffer requirements, wormhole routing is preferred and is widely used in recent multicomputers [1].

Message passing in multicomputer systems is implemented based on a routing algorithm that determines the path a message follows to reach its destination. If the path between every pair of source and destination is fixed, the algorithm is called *deterministic.* For better system performance, it is preferable that the algorithm adapts itself to the traffic congestion by providing alternate paths. Adaptive routing algorithms are classified as *partially adaptive* or *fully adaptive.* Partially adaptive routing algorithms use only a subset of the available physical paths between the source and the destination. Turn model [2, 3], direction restriction model [4], and planar-adaptive routing [5] are examples of partially adaptive algorithms. Examples of fully adaptive algorithms include the routing schemes proposed by Linder and Harden [6], Duato [7], Su and Shin [8], Boura and Das [9], and Schwiebert and Jayasimha [10].

The adaptive algorithms presented in [7, 8, 9, 10] try to achieve more adaptivity by allowing more number of alternate paths for message routing. In order to achieve high adaptivity, these algorithms often favor some messages or some paths over the others, which in turn, cause an uneven traffic distribution in the network. As a result, a part of the network is heavily loaded whereas other regions may be sparsely utilized. This uneven network utilization often results in an early saturation of the network and limits the system performance. The system performance, thus depends not only on adaptivity of the algorithm, but also on how evenly the network traffic is distributed. In fact, our simulation results clearly indicate that the traffic distribution created by the algorithm has a significant impact on the system performance. In [11], Boppana and Chalasani have also shown that more adaptivity does not necessarily mean better performance. Thus, the main motivating factor behind our work is to develop a new routing algorithm that is not only more adaptive but also creates a balanced traffic distribution.

In this paper, we propose a fully adaptive minimal routing scheme for two-dimensional (2D) meshes. The algorithm uses only two virtual channels [12] per physical channel creating two virtual networks. Messages are routed positive-first in one virtual network and negative-first in the other. Because of the way the algorithm uses two distinct virtual networks, we call it *Positive-First-Negative-First* (PFNF) algorithm. The proposed routing scheme is described in two phases. First, we introduce a new concept called the *region of adaptivity*, the region where messages can be routed using all the available paths. Using this concept, we show how various algorithms cause an uneven traffic distribution in the network and their effect on the system performance. Second, we present the details of the PFNF routing algorithm. The results indicate that the PFNF routing algorithm outperforms the previously proposed adaptive routing algorithms in terms of the network latency and throughput. Using the concept of region of adaptivity and the simulation results we also show that the PFNF algorithm creates a balanced traffic load in the network. 3P [8], mesh_route [9], and opt-y [10] algorithms are considered for performance comparison because of their high adaptivity using the same amount of hardware resources.

The rest of the paper is organized as follows. Section 2 summarizes the required definitions. Section 3 discusses the motivation behind our work. The PFNF routing algorithm is described in Section 4. Simulation results are presented in Section 5 followed by the concluding remarks in Section 6.

2 Preliminaries

In this section, we define the terminologies associated with the adaptive routing scheme. Some of these definitions are reiterated from previous works [7, 13, 14] for the sake of completeness.

Definition 1: A *physical interconnection network*, PN, is a strongly connected graph, $PN(PV, PC)$, where PV represents the set of processing nodes and PC represents the set of physical channels connecting the nodes.

Definition 2: A *virtual interconnection network*, VN, is a strongly connected graph, $VN(PV, VC)$, where PV represents the set of processing nodes and VC represents the set of virtual channels, that are mapped to the set of physical channels PC.

Definition 3: A *routing function* $R : N \times N \rightarrow \rho(C)$, where $\rho(C)$ is the power set of VC, supplies a set of alternative output channels to send a message from current the node x to the destination node d. $R(x, d) = (c_1, c_2, \ldots, c_p)$.

Definition 4: A routing function for a given interconnection network is connected iff, for any pair of nodes x, $y \in N$, it is possible to establish a path $P(x, y) \subset \rho(C)$ between them using channels supplied by R.

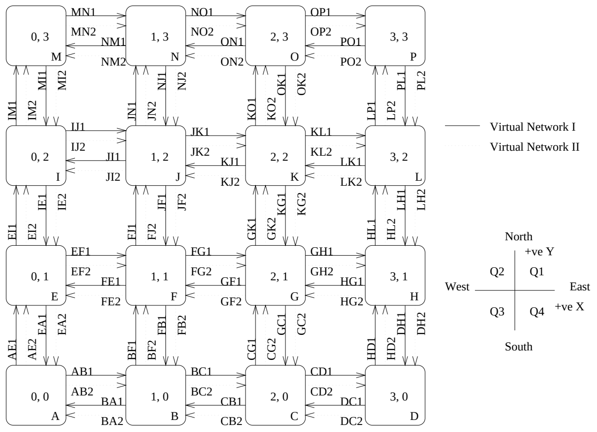

Figure 1: A 4×4 two-dimensional mesh.

Figure 1 shows a 4×4 2D mesh with two virtual channels per physical channel. The two virtual networks are shown as $VN1$ and $VN2$. Each node in the mesh is identified by a 2-coordinate vector (x_0, x_1). The figure also shows the directional notations, the quadrants, and labels used in this paper.

Definition 5: *Region of Adaptivity* is the area in which a message can route fully adaptively using all the available virtual channels. Quantitatively, the region of adaptivity of a source node is defined as a region constituting a set of contiguous nodes through which a message sent by the source can be routed fully adaptively using the underlying routing algorithm (i.e., using all of the physical paths or all of the virtual paths, if virtual channels are used). The region of adaptivity of a routing algorithm is the region of adaptivity of a source node at the center of the network.

Consider a 2D mesh network as shown in Figure 2(a). With node $(3,1)$ as the source node, under the East-First algorithm [3], all the messages directed towards any of the nodes in the shaded region can be routed fully adaptively, while messages to any node outside this region would be routed deterministically. We call the shaded region as the region of adaptivity of the node $(3,1)$ under the East-First algorithm. Figure 2(b) represents the region of adaptivity for the node $(1,2)$. As a whole, the region of adaptivity of the East-First algorithm can be represented as the shaded region shown in Figure 2(c). If virtual channels are used, the region of adaptivity can be obtained by considering adaptive regions of all the virtual networks.

The concept of region of adaptivity helps us understand the behavior of the algorithms, particularly on how evenly the algorithm distributes the network load. Since the East-First algorithm has one-half of the mesh as the fully adaptive region, all the messages have more number of alternate paths to route in this region than in the other half of the mesh. Since this region is not symmetric in the mesh, assuming uniform traffic generation at all nodes, it causes

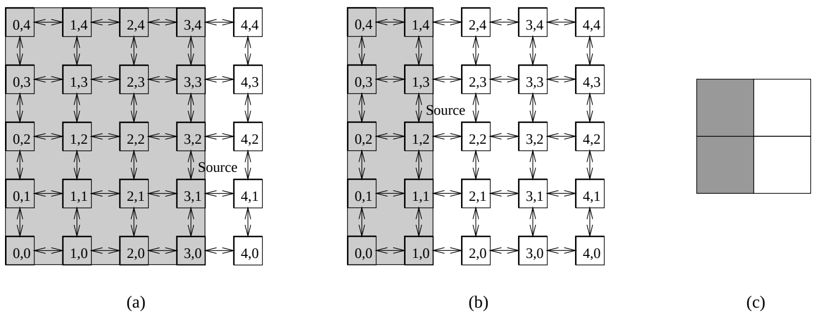

Figure 2: Region of adaptivity of (a) node $(3,1)$ (b) node $(1,2)$ (c) the East-First algorithm.

more traffic congestion in one part of the network and hence lead to early saturation. Saturation in only one part of the network saturates the rest of the network and hence degrades the system performance. Previous results have indeed shown that the symmetric Negative-First algorithm performs better than the partially East-First algorithm under uniform traffic distribution [3]. Thus, the region of adaptivity closely relates to the performance of the algorithm.

3 Motivation

The motivating factor for development of our algorithm is driven from the observation that most of the fully adaptive algorithms presented in the literature either have more routing restrictions or their adaptivity is improved at the expense of uneven traffic distribution in the network. We illustrate this point by comparing two recently proposed algorithms – 3P [8] and mesh_route [9]. Both 3P and mesh_route algorithms divide the virtual channels into two separate sets - waiting channels and non-waiting channels. Using 3P algorithm, a message can travel using any of the non-waiting virtual channels. If it gets blocked, it travels through the waiting channels using dimension order routing [14]. Similarly, in mesh_route algorithm, a message can travel using any available non-waiting channel. If the non-waiting channels at a node are not available, then the messages are restricted to the dimension order routing in waiting channels, except those who have to go negative in both x and y directions. These messages can use all the waiting channels without any restriction.

The average buffer utilizations of 3P and mesh_route algorithms for uniform traffic are shown in Figures 3 (a) and (b), respectively. While 3P algorithm distributes the traffic very evenly, the traffic is concentrated in one quadrant of the mesh for the mesh_route algorithm. This uneven load distribution becomes a bottleneck as the high traffic areas saturate early and in turn saturate the whole network. The 3P algorithm has more routing restrictions due to the dimension order routing in the waiting channels. On the other hand, mesh_route has relatively

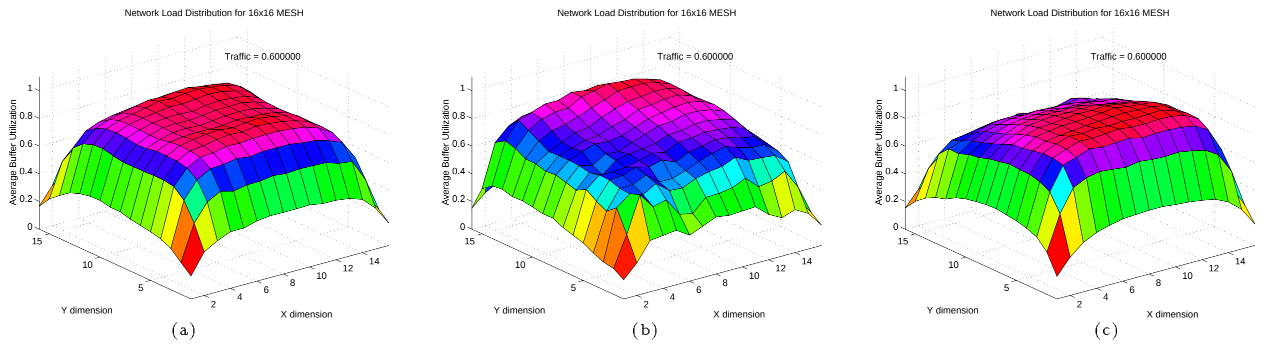

Figure 3: Traffic distribution (a) 3P algorithm (b) mesh_route algorithm (c) opt-y algorithm.

less routing restrictions and it favors messages going in the negative directions and hence creates an uneven traffic distribution in the network.

The recently proposed opt-y algorithm is proven to be optimal in terms of adaptivity and the number of required virtual channels [10]. But the traffic distribution created by the algorithm is not symmetric when the messages are uniformly generated, as shown in Figure 3 (c). Thus the goal of our work was to design an algorithm that is more adaptive as well as produces a balanced and symmetric network traffic load and thereby improves the system performance.

4 PFNF Routing Algorithm

The basic concept behind our algorithm is as follows. The physical interconnection network PN is logically divided into two virtual networks, $VN1$ and $VN2$, such that two virtual channels associated with the same physical channel are in different virtual networks. A different routing algorithm is used in each of the two virtual networks, $VN1$ and $VN2$. At each step, a set of virtual channels are chosen from the two virtual networks depending upon the routing tag and the routing function for that particular virtual network. The selection function then selects the channel through which the message is routed.

The PFNF algorithm for 2D mesh can be described as follows. A virtual channel directed from node (x_0, x_1) to (d_0, d_1) is denoted by $vc_{VN}((x_0, x_1), (d_0, d_1))$, where VN is the virtual interconnection network to which the virtual channel vc belongs. The routing algorithm and the associated procedures are defined as follows.

Routing algorithm(message_header)

```
/* Let the current node be $(x_0, x_1)$ and destination be $(d_0, d_1)$. */

1. Routing_tag(message_header).           /* form the routing tag. */
2. If all elements in routing_tag = 0, store the current message and return.
3. $VC$ = Routing_function(routing_tag).  /* determine the set of virtual channels, $VC$,
                                             for the next step routing. */
4. $vc$ = Selection_function(VC).         /* select an appropriate $vc$ from $VC$. */
5. If $vc \neq \emptyset$, forward the message along virtual channel, $vc$.
```

Procedure Routing_tag(message_header)

```
1. For $i = 0$ to 1 do
       If $d_i - x_i > 0$, routing_tag[i] = 1
       If $d_i - x_i < 0$, routing_tag[i] = -1
       If $d_i - x_i = 0$, routing_tag[i] = 0 /* Finish routing in dimension $i$ */
   end.
2. return routing_tag.
```

Procedure Selection_function(VC)

```
1. If the number of members in $VC = 1$, then select it.
2. Otherwise, use multiplex-turn bias (explanation follows) to select a $vc$.
```

When the message header arrives at an intermediate node, the routing_tag is calculated. This routing_tag is used to determine the routing dimension to be completed. The incoming message is consumed if all elements in the routing_tag is equal to zero. The function, Routing_function (described later), returns all virtual channels that are allowed by the routing algorithm. The specific virtual channel taken by a message is chosen by Selection_function. Selection_function will first check the availability of all virtual channels in the physical channels. If the number of available virtual channels is more than one, a selection policy is applied. The selection policy can be random, turn biased, or multiplex-turn biased. In random selection policy, the virtual channel is chosen randomly from the free set. The network contention is reduced if messages avoid making turns. The turn bias policy selects the virtual channel in the same direction if possible. The performance degradation due to the virtual channel is attributed to the delay in multiplexing the physical channel. The multiplex-turn bias first avoids sharing the physical channel with other message if possible, and then avoids making a turn as a second priority. The effect of these selection policies have been studied in [2, 15]. The simulation results show that the turn bias and multiplex-turn bias perform better than the random selection policy [2, 15, 17].

Procedure Routing_function(routing_tag)

```
$VC = \emptyset$
1. If all elements in routing_tag $\leq 0$
        If routing_tag[0] $< 0$
            add $vc_{VN1}((x_0, x_1), (x_0 - 1, x_1))$ and $vc_{VN2}((x_0, x_1), (x_0 - 1, x_1))$ to $VC$.
        If routing_tag[1] $< 0$
            add $vc_{VN1}((x_0, x_1), (x_0, x_1 - 1))$ and $vc_{VN2}((x_0, x_1), (x_0, x_1 - 1))$ to $VC$.
        return virtual channel set VC.
2. If all elements in routing_tag $\geq 0$
        If routing_tag[0] $> 0$
            add $vc_{VN1}((x_0, x_1), (x_0 + 1, x_1))$ and $vc_{VN2}((x_0 + 1, x_1), (x_0, x_1))$ to $VC$.
        If routing_tag[1] $> 0$
            add $vc_{VN1}((x_0, x_1), (x_0, x_1 + 1))$ and $vc_{VN2}((x_0, x_1), (x_0, x_1 + 1))$ to $VC$.
        return VC.
3. If routing_tag[0] $> 0$
        add $vc_{VN1}((x_0, x_1), (x_0 + 1, x_1))$ to $VC$.
   If routing_tag[0] $< 0$
        add $vc_{VN2}((x_0, x_1), (x_0 - 1, x_1))$ to $VC$.
   If routing_tag[1] $> 0$
        add $vc_{VN1}((x_0, x_1), (x_0, x_1 + 1))$ to $VC$.
   If routing_tag[1] $< 0$
        add $vc_{VN2}((x_0, x_1), (x_0, x_1 - 1))$ to $VC$.
   return VC.
```

In the procedure of routing function, PF routing algorithm is implemented in $VN1$ and NF routing algorithm is implemented in $VN2$. In both PF and NF routing algorithms, a message can route without restriction to destinations in $< +x, +y >$ and $< -x, -y >$ directions from the source. When the destination is located in the directions $< -x, +y >$ or $< +x, -y >$ of the source node, the routing restrictions of PF and NF algorithms are applied. To implement the PF algorithm in one virtual network and NF routing algorithm in another, the routing function is divided into three cases. The first two cases return virtual channels in both $VN1$ and $VN2$ for the message destined to $< +x, +y >$ and $< -x, -y >$ directions. The third case returns virtual channels in the $VN1$ for the positive direction and the $VN2$ for the negative direction. The routing restrictions for PF and NF are applied in this case.

The deadlock freedom of the algorithm can be proved by using Duato's theorem stated as follows [16]. The proof of the theorem uses several terminologies associated with the channel dependency graph. These terminologies are defined earlier in [7, 14, 16].

Theorem 1: For a given interconnection network IN, a routing function R is deadlock-free iff there exists a routing subfunction R_1 which is connected and has no cycles in its extended channel dependency graph.

To prove that the PFNF algorithm is deadlock free, we first analyze the routing restrictions. The turn models for the PF and NF routing algorithms [3] are depicted in Figure 4 (a) and (b), respectively. The two sets of channels belonging to the virtual networks are distinguished by | and ||, respectively. Dotted lines represent restricted turns. Glass and Ni [3] showed that without any extra virtual channel, the routing algorithm is deadlock free if there is no cycle formed in turn model. When we consider the virtual networks individually, the routing algorithms in both the virtual networks (PF and NF) are deadlock-free. Since the PFNF algorithm allows a message to change from one virtual network to another, the channel dependencies between virtual networks also need to be considered. The turns involved between the two virtual networks are shown in Figures 4 (c) and (d). Dashed lines represent the conditionally restricted turns. These turns are restricted if a message has more than one dimension to traverse; otherwise, these conditionally restricted turns are allowed.

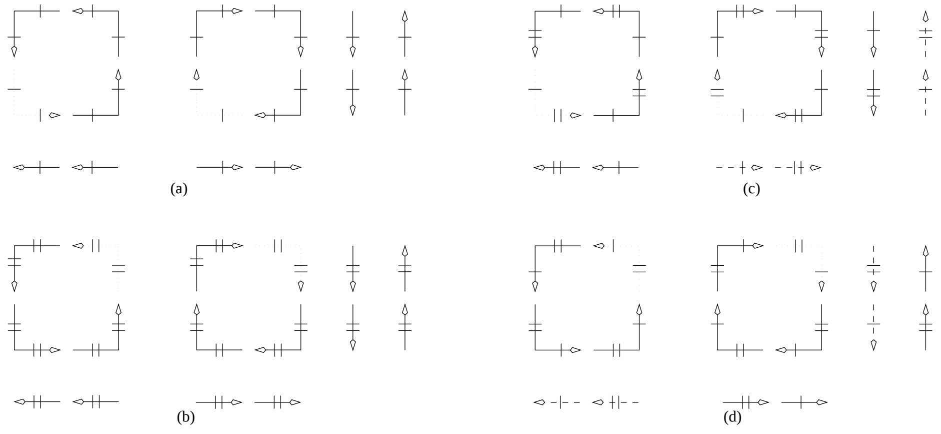

Figure 4: Routing restrictions in PF, NF, and PFNF algorithms.

The turn restrictions for the PFNF algorithm are summarized as follows:

- Within $VN1$: South-East and West-North turns are restricted. (PF restrictions)
- Within $VN2$: North-West and East-South turns are restricted. (NF restrictions)
- From $VN1$ to $VN2$: South-East and West-North turns are restricted. A message can use $VN1$ in South(West) direction iff it has finished routing in the East(North) direction using the PF restrictions imposed in $VN1$. Hence, the South-East(West-North) turns will not occur.
- From $VN2$ to $VN1$: North-West and East-South turns are restricted. A message can use $VN2$ in North(East) direction iff it has finished routing in the West(South) direction using the NF restrictions imposed in $VN2$. Hence, the North-West(East-South) turns will not occur.
- From $VN1$ to $VN2$: While routing in the North(East) direction, messages traveling from $VN1$ to $VN2$ are conditionally restricted. From NF restrictions in $VN2$, a message in $VN1$ can route further in the North(East) direction using $VN2$, only when the routing is not needed in any other negative direction. Hence, changing form $VN1$ to $VN2$ in North(East) direction is prohibited if a message has to route in the West(South) direction at a later time.

- From $VN2$ to $VN1$: While routing in the South(West) direction, messages traveling from $VN2$ to $VN1$ are conditionally restricted. From PF restrictions in $VN1$, a message in $VN1$ can route further in the South(West) direction using $VN2$, only when the routing is not needed in any other positive direction. Hence, changing form $VN2$ to $VN1$ in the South(West) direction is prohibited if a message has to route in the East(North) direction at a later time.

A routing subfunction R_1 is defined to show that an escape path without cyclic dependency always exists. Using R_1, a message is routed in dimension-order in $VN2$ in the North direction, and in $VN1$ in the South direction from the source node. R_1 is stated as follows: If the destination node (d_0, d_1) is equal to the current node (x_0, x_1), the message is consumed. If $d_1 < x_1$, the message is forwarded using dimension-order routing in $VN1$. If $d_1 > x_1$ then the message is forwarded using dimension-order routing in $VN2$. In other words, R_1 is conditionally assigned to both of the virtual networks.

Lemma 1. The routing subfunction R_1 is connected.

Proof: It is known that the dimension-order routing is connected. Therefore, the routing algorithm that assigns one virtual network to a message headed toward North direction and another virtual network for a message headed toward South direction is also connected. □

The dependency cycles can be classified as intra-dependency cycles and inter-dependency cycles. Virtual channels in the same virtual network are involved in the intra-dependency cycles. The inter-dependency cycles consist of virtual channels from different virtual networks. To prove that there is no cycle in the extended channel dependency graph, we need to show that there are no intra-dependency or inter-dependency cycles in the graph.

Lemma 2. Using the routing subfunction R_1, there is no cycle formed due to the channel dependencies in the same virtual network (intra-dependency cycle).

Proof: To form a counter-clockwise cycle within the same virtual network, virtual channels in the directions $(-x, -y, +x, +y)$ in sequence are required. Similarly, virtual channels in the directions $(+x, +y, -x, -y)$ in sequence are required for the formation of a clockwise cycle. The virtual channels directed to North(South) that belong to $VN1(VN2)$ cannot be supplied by R_1 (Since R_1 supplies virtual channels in $VN2$ for the North bound messages, the $+y$ virtual channels in $VN1$ is not supplied by R_1). Thus, the $+y$ virtual channels in $VN1$ are not involved in the extended channel dependency graph and no intra-dependency cycle can be formed in $VN1$. The same scenario is applicable in $VN2$, where $-y$ virtual channels are not supplied by R_1. □

Lemma 3. Using the routing subfunction R_1, there is no cycle created from the channel dependencies in one virtual network combined with the channel dependencies in the other virtual network (inter-dependency cycle).

Proof: The virtual channels that are involved in the direct and indirect dependencies are supplied by R_1. Since R_1 defines different virtual networks for messages headed in North and South directions, the direct and the indirect dependencies are within the same virtual network. However, the dependency between virtual channels of two different networks can be created from cross dependencies. Using R_1, the virtual channels that are assigned to restricted destinations are only in the x dimension. For example, $-x$ virtual channels in $VN2$ cannot be assigned using R_1 to route a South bound message, but it can be assigned to a North bound message. The inter-cross dependencies[2] from $VN2$ to $VN1$ are only from $-x$ virtual channels of $VN2$ to $-y$ or $-x$ virtual channels of $VN1$. Similarly, the inter-cross dependencies from $VN1$ to $VN2$ are only from $+x$ virtual channels of $VN1$ to $+y$ or $+x$ virtual channels of $VN2$. These dependencies are shown in Figure 5. The only possible way of forming a cycle is the channel dependencies in sequence: ([$-x$ in $VN2$], [$-y$ in $VN1$], [$+x$ in $VN1$], [$+y$ in $VN2$], [$-x$ in $VN2$]). But, there is no dependency from [$-y$ in $VN1$] to [$+x$ in $VN1$] using PF restrictions. Similarly, there is no dependency from [$+y$ in $VN2$] to [$-x$ in $VN2$] using NF restrictions. Hence, there is no inter-dependency cycle in the extended channel dependency graph. $\square$

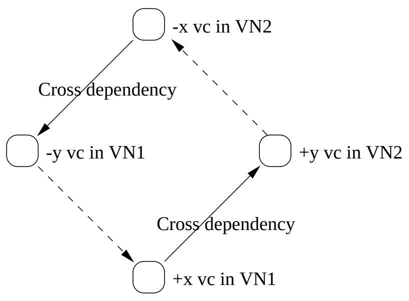

Figure 5: Possible cycle that can be formed between virtual networks.

Theorem 2. PFNF routing algorithm is deadlock free.

Proof: From Lemma 1 through 3, and using Theorem 1, PFNF routing algorithm is deadlock-free. $\square$

Next, we compare the region of adaptivity of all the four algorithms discussed earlier. As all these algorithms use at most one additional virtual channel per physical channel, we consider the whole network as consisting of two separate virtual networks. Region of adaptivity for each virtual network is(are) the quadrant(s) in which full adaptivity is offered by the algorithm in that particular network. The region of adaptivity of the actual network is then obtained by superimposing the adaptive regions of the two virtual network. The adaptive regions for each

[2]The inter-cross dependencies include both direct cross dependencies and indirect cross dependencies that connect channels in different virtual networks.

virtual network and the total region of adaptivity for the four algorithms are shown in Figure 6. The intensity of the shaded areas represent the adaptivity in the regions of the network as explained in Section 2. Note that, in Figure 6(d) the combination of deterministic algorithms in the second and fourth quadrants of the two virtual networks give full adaptivity in the actual physical network. For example, in the second quadrant the PF virtual network allows YX routing and the NF virtual network allows XY routing, thus providing full adaptivity. The figure demonstrate the comparison of the adaptive regions of the algorithms. The symmetry of the adaptive regions are also illustrated. It can be inferred from Figure 6 that since the region of adaptivity for the PFNF algorithm is symmetric and more adaptive, it would perform better than the other algorithms.

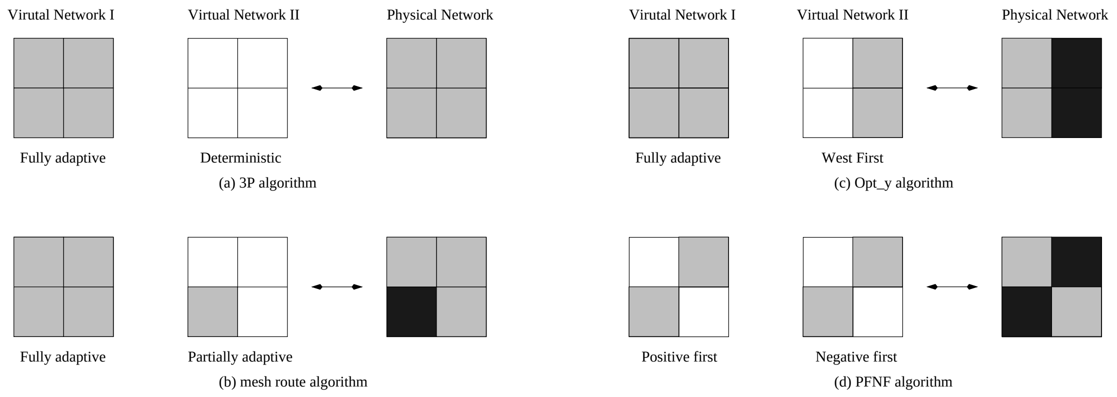

Figure 6: Region of adaptivity for (a) 3P (b)mesh_route (c) Opt-y (d) PFNF algorithms.

5 Performance Evaluation

In this section, we present results for various traffic patterns and network loads. We have compared the performance of the PFNF algorithm with that of the 3P, mesh_route, and opt-y algorithms. These algorithms are shown in the literature to have better performance or adaptivity than other existing adaptive routing algorithms.

We have developed an event driven simulator to evaluate the performance of the aforementioned routing algorithms. The results were reproduced several times and were observed to be consistent with a maximum deviation of only 1%. The simulations were conducted on a 16×16 mesh. We have assumed 20 flits per packet. Each virtual channel is assumed to have only one flit buffer associated with it. Packet generation rate is assumed to have an exponential distribution of inter-arrival time. We have used multiplex-turn biased selection policy for all the algorithms. The simulation was carried out for 150,000 packets. The effect of the first 40,000 to 60,000 delivered packets are not included in the results in order to reduce the transient effects in the simulations.

Uniform, hotspot, and transpose traffic patterns were considered in our study. Under the uniform traffic pattern, a node sends messages to every other node with equal probability. Under hotspot traffic, one particular node receives some additional traffic besides its normal traffic. Using the parameters from [11] we have considered only one hotspot node with the hotspot percentage of four, i.e., in a 16×16 mesh a message is directed to the hotspot node with a probability of 0.0438 and to each of the other nodes with a probability of 0.0038. We have chosen node $(5,5)$ as the hotspot node. Under transpose traffic, a message from node (i,j) is directed to node (j,i) if $i \neq j$. If $i = j$ node (i,i) sends messages to node $(K-i, K-i)$, where K is the network radix.

We have studied the average communication latency, the average throughput of the network and the network load distribution in the network. The communication latency is defined as the average time from the message generation to the time when the tail reaches the destination. The throughput is the average number of messages that finish routing per unit time. The network traffic distribution is measured by finding the average flit buffer utilization at each node. All the above parameters are studied against the network traffic. The network traffic is defined as the ratio of the average traffic generated by a node to the average bandwidth available per node.

Figures 7(a) and 7(b) plot the average latency and average throughput of the network against the network traffic under the uniform traffic pattern. In the low traffic region, all the four algorithms result in almost the same average latency. However, as the traffic is increased, opt-y and mesh_route saturate first and their latencies increase rapidly. The PFNF algorithm performs better than all the schemes. The same trend is also observed in the throughput results. All the algorithms give the same throughput for lower traffic rates (< 0.3), however at higher traffic, the throughputs of the opt-y and mesh_route algorithms drop abruptly. Throughput using 3P and PFNF schemes do not drop abruptly but instead saturate close to their maximum values.

The opt-y and mesh_route algorithms have less routing restrictions than the 3P routing. However, they create uneven traffic distribution in the network. The fact that 3P shows higher performance compared to opt-y and mesh_route, confirms our claim that the system performance depends significantly upon how evenly the traffic is distributed. The PFNF algorithm is highly adaptive and it also distributes the network load symmetrically, and hence demonstrates better performance.

To demonstrate that the PFNF algorithm indeed distributes the network load symmetrically we have plotted the traffic distribution for the PFNF algorithm in Figure 8. The results are in accordance with what was expected from the region of adaptivity analysis and clearly illustrate

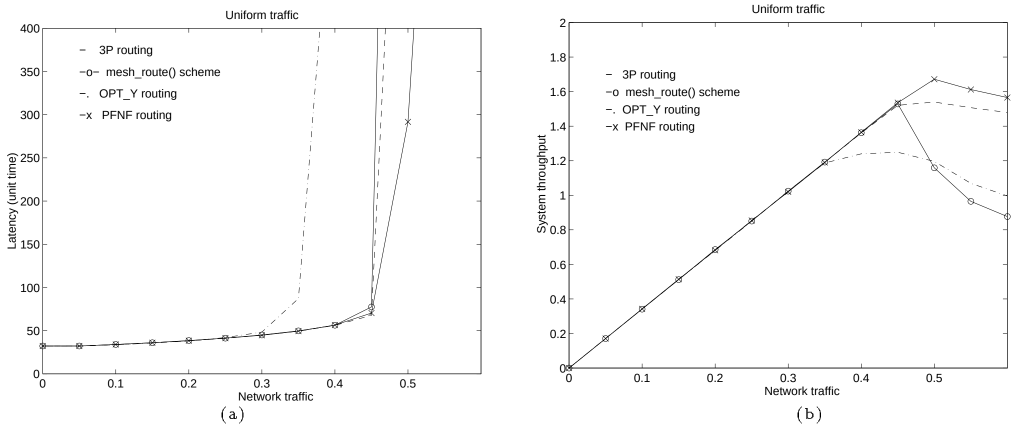

Figure 7: Performance results for the uniform traffic pattern.

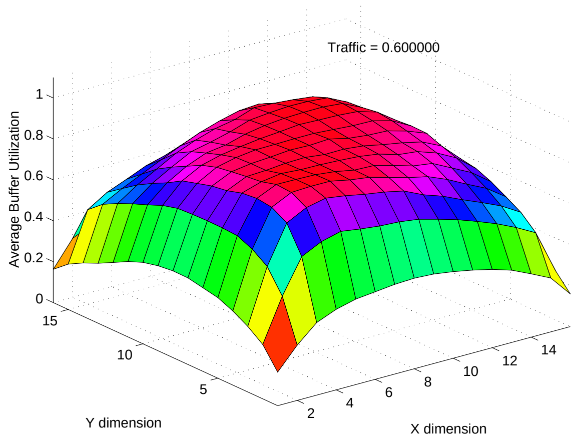

Figure 8: Traffic load distribution produced by the PFNF algorithm.

that the traffic distribution is more balanced using the PFNF algorithm than the mesh_route (refer to Figure 3 (b)).

Figure 9 shows the latency and throughput results under the hotspot traffic pattern. The opt-y and mesh_route algorithm perform poorly and saturate early. The PFNF algorithm outperforms all the three schemes in both the latency and throughput results. The same trend is observed under the transpose traffic pattern shown in Figure 10.

It should be noted that the opt-y algorithm uses a total of six virtual channels per router compared to eight used by the other three algorithms in a 2-dimensional mesh. Providing the same resources to opt-y will enhance the performance and it becomes almost equal to that of the 3P algorithm [17].

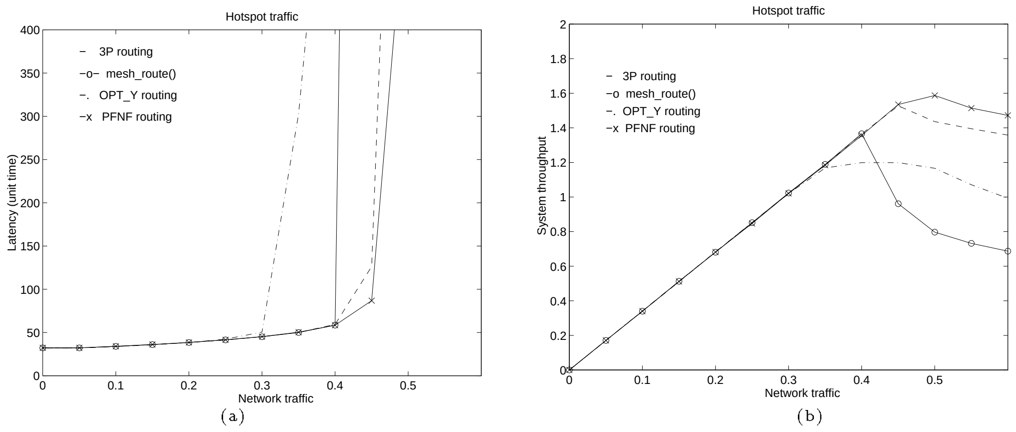

Figure 9: Performance results for the hotspot traffic pattern.

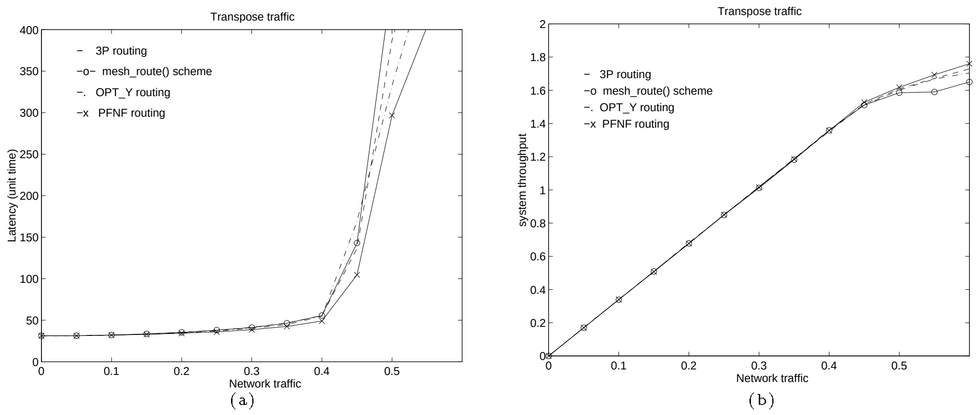

Figure 10: Performance results for the transpose traffic pattern.

6 Conclusions

In this paper we have analyzed adaptive routing in 2D meshes in light of two new concepts –region of adaptivity and balanced traffic distribution. Previously proposed fully adaptive algorithms have aimed at improving the adaptivity of the routing schemes. We have demonstrated through simulations that symmetric and balanced traffic distribution have a significant impact on the system performance along with higher adaptiveness. Motivated by this observation, we have proposed a new adaptive routing algorithm, called Positive-First–Negative-First(PFNF) for two-dimensional meshes. The algorithm uses a combination of Positive-First and Negative-First routing to balance the traffic distribution in the network. The simulation results show that the proposed scheme performs better than the previous schemes in terms of the average network latency and throughput. Our current work is focussed on designing a fault-tolerant PFNF routing algorithm and extending the algorithm for n-dimensional meshes [17].

References

[1] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct network," *IEEE Computers*, vol. 2, pp. 62-76, Feb. 1993.

[2] C. J. Glass and L. M. Ni, "Adaptive routing in mesh-connected networks," *Proceedings of the 1992 Int'l Conference on Distributed Computing Systems*, pp. 12-19, 1992.

[3] C. J. Glass and L. M. Ni, " The turn model for adaptive routing," *Jou. of the ACM*, pp. 874-902, Sept. 1994.

[4] Y. M. Boura and C. R. Das, "A class of partially adaptive routing algorithms for n-dimensional meshes," *Int'l. Conference on Parallel Processing*, vol. 3, pp. 175-182, Aug. 1993.

[5] A. A. Chien and J. H. Kim, "Planar adaptive routing: Low-cost adaptive networks for multiprocessors," *Jou. of the ACM*, pp. 91-123, Jan 1995.

[6] D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k-ary n cubes," *IEEE Trans. on Computers*, vol. 40, pp. 2-12, Jan. 1991.

[7] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole network," *IEEE Trans. on Parallel and Distributed systems*, vol. 4, no. 12, pp. 1320-1331, Dec. 1993.

[8] C. Su and K. G. Shin, "Adaptive deadlock-free routing in multicomputers using only one extra channel," *Proc. of the 22nd Int'l Conference on Parallel Processing*, vol. 3, pp. 175-182, Aug. 1993.

[9] Y. M. Boura and C. R. Das, "Efficient fully adaptive wormhole routing in n-dimensional meshes," *Int'l. Conference on Distributed Computing Systems*, 1994.

[10] L. Schwiebert and D. N. Jayasimha, "Optimally Fully Adaptive Minimal Wormhole Routing for Meshes," *Jou. of Parallel and Distributed Computing,* vol. 27, pp. 56-70, 1995.

[11] R. V. Boppana and S. Chalasani, "A comparison of adaptive wormhole routing algorithms," *Int'l. Symposium on Computer Architecture*, pp. 351-360, May 1993.

[12] W. J. Dally., "Virtual channel flow control," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, pp. 194-205, Mar. 1992.

[13] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Trans. on Computers*, vol. 39, no. 6, pp. 775-785, June 1990.

[14] W. J. Dally and C. L. Sietz, "Deadlock free message routing in multiprocessor interconnection networks," *IEEE Trans. on Computers*, vol. 36, no. 5 , pp. 547-553, May 1987.

[15] W. J. Dally and H. Akoi, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 4, pp. 466-475, April 1993.

[16] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," *IEEE Trans. on Parallel and Distributed Systems,* vol. 6, no. 10, pp. 1055-1067, Oct. 1995.

[17] V. Varavithya, "Wormhole routing algorithms for mesh interconnection networks," *Master's Thesis*, Dept. of Electrical and Computer Engineering, Iowa State University, 1994.