Efficient fully adaptive wormhole routing in n-dimensional Meshes*

Younes M. Boura and Chita R. Das
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {boura,das}@cse.psu.edu

Abstract

An efficient fully adaptive wormhole routing algorithm for n-dimensional meshes is developed in this paper. The routing algorithm provides full adaptivity at a cost of one additional virtual channel per physical channel irrespective of the number of dimensions of the network. The algorithm is based on dividing the network graph into two acyclic graphs that contain all of the physical channels in the system. Virtual channels are classified as either waiting or nonwaiting channels. Busy channels that a message waits for to become available are classified as waiting channels, otherwise they are classified as nonwaiting channels. Thus, a message considers nonwaiting channels first to reach its destination. If all nonwaiting channels are busy, the message considers waiting channels. Messages acquire waiting channels in two phases. In each phase, waiting channels belonging to one acyclic network graph are traversed. This 2-phase routing algorithm could be either minimal or nonminimal. However, in this paper we concentrate on minimal routing. It is demonstrated that this adaptive routing algorithm can utilize the virtual paths (channels) between any two nodes more efficiently than any of the present algorithms with the same hardware requirement.

1 Introduction

A fundamental requirement for designing high performance multiprocessors is the availability of high-throughput low-latency communication networks. The parameters that determine the efficiency of such networks include the topology, switching mechanism, and routing algorithm. Direct networks represent a family of such high performance interconnection topologies. Unlike the constant path-length topologies such as multistage networks, direct networks provide low-latency communication by exploiting the communication locality exhibited by many scientific applications. Therefore, many contemporary parallel machines have adopted different instances of direct networks like hypercubes, n-dimensional meshes, and tori. Examples include MPP [5], Tera computer system [1], Touchstone Delta system [12], K2 [2], DASH [13], ALEWIFE [4], iPSC [19], and nCUBE [16].

Virtual channels were introduced in [9] for designing deadlock free routing algorithms. Virtual channels are abstractions that share the same physical link. Every virtual channel has its own separate queue. The bandwidth of a physical link is shared among the virtual channels by using time-multiplexing.

It was shown in [15] that 2^{n-1} virtual channels per physical channel are required to support full adaptivity in n-dimensional meshes. This high virtual channel requirement for making routing algorithms fully adaptive motivated researchers to develop partially adaptive routing algorithms. *Planar adaptive* routing [7], the *turn model* [11], and the *direction restriction model* [6] are three such partially adaptive schemes applicable to n-dimensional meshes. *Planar adaptive* routing requires 3 virtual channels per physical channel whereas the last two routing models do not need any extra hardware (virtual channels). The main drawback of these algorithms is their inability to use all possible paths that exist between any two nodes.

Duato introduced a fully adaptive minimal routing algorithm for binary hypercubes that requires only two virtual channels [10]. The algorithm is based on the creation of restricted and unrestricted virtual channels (virtual networks). At each routing step, a message is allowed to traverse any available unrestricted virtual channel. If the message gets blocked, then it is routed along a restricted virtual channel by using the oblivious *dimension order* routing algorithm (*e-cube* routing

*This research was supported in part by the National Science Foundation under grant MIP-9104485.

[18]). Fully adaptive minimal routing algorithms for n-dimensional meshes that also have a hardware requirement of one extra virtual channel per physical channel were introduced in [14][17]. All three algorithms are based on the same concept and hence have similar performance.

Even though the adaptive routing algorithms described in [10][14][17] make use of all the physical shortest paths that exist between any two nodes, they do not utilize all of the virtual shortest paths that exist in the network due to the usage of the oblivious *dimension order* routing algorithm in the restricted network. The inherent chained blocking that results in wormhole switching calls for the efficient utilization of the virtual paths in order to improve the performance of the network [8].

In this paper, an efficient fully (minimal and nonminimal) adaptive *wormhole* routing algorithm for n-dimensional meshes is developed. Like the previous schemes [10][14][17], only one additional virtual channel per physical channel is required for the prevention of deadlock. However, the new algorithm is shown to be more efficient in utilizing the virtual (channels) paths that exist between any two nodes than the previous algorithms having the same hardware requirement. The algorithm is based on dividing the network graph into two acyclic graphs that contain all of the physical channels of the system. Virtual channels are classified as either *waiting* or *nonwaiting* channels. The classification is based on the behavior of a message when it encounters a busy channel. Busy channels that a message waits for to become available are classified as *waiting* channels, otherwise they are classified as *nonwaiting* channels. Thus, a message considers *nonwaiting* channels first to reach its destination. If all *nonwaiting* channels are busy, the message considers *waiting* channels. Messages acquire *waiting* channels in two phases. In each phase, *waiting* channels belonging to one acyclic graph are traversed. This 2-phase routing algorithm could be either minimal or nonminimal. However, in this paper we concentrate on minimal routing. We compare our scheme and the previous schemes in terms of the efficiency of virtual channel utilization. It is shown that the proposed routing algorithm can utilize the available virtual channels between any two nodes more effectively than the other schemes. This improvement becomes significant as the network size increases.

The rest of the paper is organized as follows. Necessary notations and definitions are introduced in Section 2. Section 3 describes the routing algorithm for n-dimensional meshes. Section 4 discusses the efficiency of the new routing algorithm. Finally, conclusions are drawn in Section 5.

2 Preliminaries

Definition 1 : An interconnection network is a strongly connected graph, $G(V, C)$, where V represents the set of processing nodes and C represents the set of communication channels.

Definition 2 : An n-dimensional mesh is defined formally as an interconnection structure that has $K_0 \times K_1 \times \ldots \times K_{n-1}$ nodes. n is the number of dimensions of the network, and K_i is the radix of dimension i. Each node is identified by an n-coordinate vector $(x_0, \ldots, x_{n-1})$, where $0 \leq x_i \leq K_i - 1$. Two nodes, $X(x_0, \ldots, x_{n-1})$ and $Y(y_0, \ldots, y_{n-1})$ are connected if and only if there exists an i such that $x_i = y_i \pm 1$, and $x_j = y_j$ for all $j \neq i$. Figure 1 shows the structure of a 4 x 4 2-dimensional mesh.

Definition 3 : A channel along dimension i is termed a positive channel if its source node $X(x_0, \ldots, x_{n-1})$ and sink node $Y(y_0, \ldots, y_{n-1})$ differ in the ith coordinate such that $x_i = y_i - 1$. A network that contains only positive channels is termed a positive network P.

Definition 4 : A channel along dimension i is termed a negative channel if its source node $X(x_0, \ldots, x_{n-1})$ and sink node $Y(y_0, \ldots, y_{n-1})$ differ in the ith coordinate such that $x_i = y_i + 1$. A network that contains only negative channels is termed a negative network N.

Assuming that there exists a unidirectional channel between any two connected nodes in each direction, a positive network and a negative network contain all of the physical channels of the system. For example, Figure 2 and Figure 3 represent P and N of a 4 x 4 2-dimensional mesh respectively.

Definition 5 : A routing function, $R : C \times V \rightarrow C$, maps the current channel c_c and destination node v_d to the next channel c_n on the route from c_c to v_d. A channel is not allowed to route to itself, $c_c \neq c_n$ [9].

Definition 6 : A channel dependency graph, CDG, of an interconnection network G and routing function R, is a directed graph, $CDG(C, E)$, where the set of vertices, C, represents the set of channels of G, and the set of edges, E, represents the set of pairs of channels connected by R [9]. Thus,

$$E = \{(c_i, c_j) | R(c_i, v) = c_j \quad for\ some\ v \in V\}.$$

Theorem 1 *A routing function R for an interconnection network is deadlock free, iff there are no cycles in CDG . In other words, R is deadlock free, iff CDG is acyclic [9].*

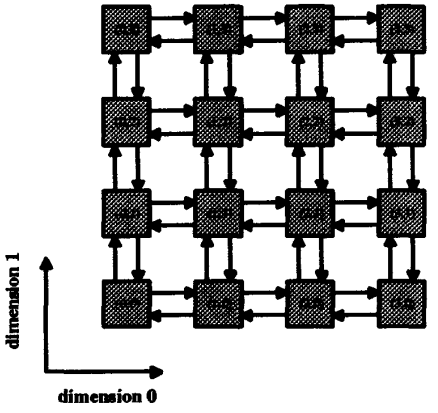

Figure 1: **A 4×4 2-dimensional mesh.**

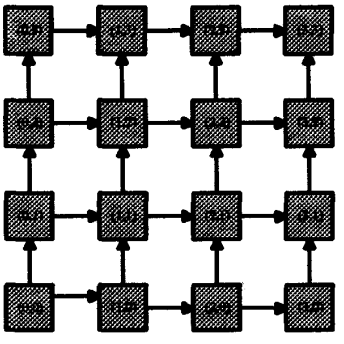

Figure 2: **A 4×4 2-dimensional positive mesh network P.**

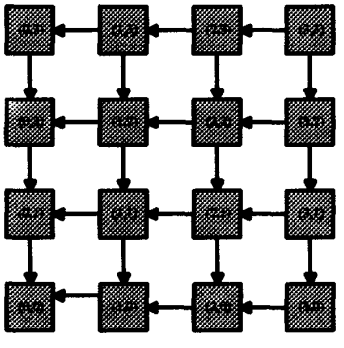

Figure 3: **A 4×4 2-dimensional negative mesh network N.**

3 Adaptive routing

A fully adaptive routing algorithm delivers a message to its destination by using any route that exists between the source and destination nodes of the message. As the number of virtual channels per physical channel increases, a distinction between virtual and physical routes arises. The number of virtual routes that exit between any two nodes is greater than the number of physical routes. The chained blocking that results in *wormhole* switching advocates the efficient utilization of virtual channels (virtual routes) in order to improve the performance of the network [8].

Virtual channels are divided into two classes. The first class consists of *waiting* channels, and the second class consists of *nonwaiting* channels. This classification is based on whether a message waits for a busy channel to become available so that it could traverse it to get closer to its destination. A message that needs to traverse a busy *waiting* channel has to wait until that channel becomes available. Hence, a *waiting* channel is a channel that can block messages. A message that encounters a busy *nonwaiting* channel does not wait until that channel becomes available. Thus, a message considers *nonwaiting* channels first to reach its destination. If all *nonwaiting* channels are busy, the message considers *waiting* channels. Previously proposed routing algorithms [10][14][17] allow messages to consider only one *waiting* channel at each routing step. Consequently, *waiting* channels are not used effectively. The proposed algorithm allows a message to consider a number of *waiting* channels at each routing step depending on the locations of the current and destination nodes of the message.

The algorithm in Figure 4 describes a fully adaptive deadlock free routing algorithm that requires one additional virtual channel per physical channel. Out of these two channels, one channel is classified as a *nonwaiting* channel and the other is classified as a *waiting* channel. A processor that needs to send a message across the network determines the routing tag ($routing\_tag[i] = destination\_address[i] - source\_address[i]$, $for \;\; 0 \leq i \leq n-1$). The integer vector *routing_tag* is appended to the *header flit* before the message is injected into the network.

The routing algorithm MESH_ROUTE() (Figure 4) routes messages by first traversing any available *nonwaiting* channel. If all *nonwaiting* channels are busy, it routes messages using *waiting* channels. *Waiting* channels are traversed in two phases. In the first phase, positive *waiting* channels are traversed one dimension at a time in an increasing order (for example, dimension i channels are traversed before dimension

j channels where $i < j$). In the second phase, negative *waiting* channels are traversed in any order depending on the availability of channels. Since blocked messages wait just for busy *waiting* channels to become available, only *waiting* channels contribute to deadlock. Therefore, in order to prove that the routing algorithm MESH_ROUTE() is deadlock free, it is needed to demonstrate that there are no cycles in the channel dependency graph consisting of only *waiting* channels. Hence, deadlock freedom is guaranteed if each phase of the routing algorithm does not introduce any deadlock.

```
MESH_ROUTE (routing_tag)
begin
  /* routing_tag[i] = destination[i] − source[i] */
  if (∄ i | routing_tag[i] ≠ 0) then
    return;
    /* message reached its destination */
  else
    if (∃ a free nonwaiting channel along dimension
      j | routing_tag[j] ≠ 0) then
      route along dimension j using a nonwaiting
      channel;
      return;
    else
      if (∃ i | routing_tag[i] > 0) then
        j := min_{0≤i≤n−1}(i| routing_tag[i] > 0);
        if (positive waiting channel along dimension
          j is free) then
          route along dimension j using a positive
          waiting channel;
          /* route along the lowest dimension in the
          positive direction */
          return;
        endif;
      else
        / * ∀ i, routing_tag[i] ≤ 0 * /
        route along any dimension i using any free
        negative waiting channel;
        return;
      end ifelse;
    end ifelse;
  end ifelse;
end;
```

Figure 4: **A fully adaptive routing algorithm for n-dimensional meshes.**

Lemma 1 *The first phase of the routing algorithm MESH_ROUTE() does not introduce any deadlock.*

Proof : In the first phase of the routing algorithm MESH_ROUTE(), messages traverse positive and negative *nonwaiting* channels and positive *waiting* channels. Since only *waiting* channels contribute to deadlock, we need to show that no cycles exist in the channel dependency graph consisting of only positive *waiting* channels. Each positive *waiting* channel is assigned a label $(dim), (A_{dim}), (d_0, \ldots, d_{n-1})$ where dim is the dimension where the channel resides, A_{dim} is the address of the sink node of the channel along dimension dim, and $(d_0, \ldots, d_{n-1})$ is the address of the sink node of the channel. For example, the channel whose sink node is node $(3, 2)$ along dimension 1 has the label $(1), (2), (3, 2)$ where $dim = 1$, $A_1 = 2$, and $(d_0, d_1) = (3, 2)$. Messages traverse positive *waiting* channels one dimension at a time in an increasing order. The following unique cases capture the scenarios that could arise when messages acquire positive *waiting* channels.

case 1 : A message acquires a positive *waiting* channel along dimension i, a number of positive *nonwaiting* channels along dimension j where $j > i$, and a positive *waiting* channel along dimension i.

$$(i), (x_i), (x_0, \ldots, x_j, \ldots, x_{n-1}) <$$
$$(i), (x_i + 1), (x_0, \ldots, x_i + 1, \ldots, x_j + k, \ldots, x_{n-1})$$
$$for \quad (0 \leq i \leq n-1) \wedge (j > i) \wedge$$
$$(0 \leq k \leq K_j - 1 - x_j). \qquad (1)$$

case 2 : A message acquires a positive *waiting* channel along dimension i, a number of negative *nonwaiting* channels along dimension j where $j \neq i$, and a positive *waiting* channel along dimension i.

$$(i), (x_i), (x_0, \ldots, x_j, \ldots, x_{n-1}) <$$
$$(i), (x_i + 1), (x_0, \ldots, x_j - k, \ldots, x_i + 1, \ldots, x_{n-1})$$
$$for \quad (0 \leq i \leq n-1) \wedge (j < i) \wedge$$
$$(0 \leq k \leq x_j). \qquad (2)$$
$$(i), (x_i), (x_0, \ldots, x_j, \ldots, x_{n-1}) <$$
$$(i), (x_i + 1), (x_0, \ldots, x_i + 1, \ldots, x_j - k, \ldots, x_{n-1})$$
$$for \quad (0 \leq i \leq n-1) \wedge (j > i) \wedge$$
$$(0 \leq k \leq x_j). \qquad (3)$$

case 3 : A message acquires a positive *waiting* channel along dimension i and a positive *waiting* channel along dimension j where $j \geq i$.

$$(i), (x_i), (x_0, \ldots, x_i, \ldots, x_{n-1}) <$$

$$(j), (x_j + 1), (x_0, \ldots, x_j + 1, \ldots, x_{n-1})$$
$$for \quad (0 \leq i \leq n-1) \wedge (j \geq i). \qquad (4)$$

Equations 1, 2, 3, and 4 demonstrate that when *nonwaiting* channels (either positive or negative) are traversed by messages, positive *waiting* channels are traversed in a monotonically increasing order. Therefore, the first phase of the routing algorithm MESH_ROUTE() does not introduce any deadlock. Q.E.D.

Lemma 2 *The second phase of the routing algorithm MESH_ROUTE() does not introduce any deadlock.*

Proof : In the second phase of the routing algorithm MESH_ROUTE(), messages traverse negative *nonwaiting* and *waiting* channels. Since only *waiting* channels contribute to deadlock, we need to show that no cycles exist in the channel dependency graph consisting of only negative *waiting* channels. Each negative *waiting* channel is assigned a label $(d_0, \ldots, d_{n-1}), (dim)$ where $(d_0, \ldots, d_{n-1})$ is the address of the sink node of the channel, and dim is the dimension where the channel exists. Since messages move only in the negative direction

$$(x_0, \ldots, x_i, \ldots, x_{n-1}), (i) >$$
$$(x_0, \ldots, x_i - k, \ldots, x_{n-1}), (i)$$
$$for \quad (0 \leq i \leq n-1) \wedge (0 < k \leq x_i). \qquad (5)$$
$$(x_0, \ldots, x_j, \ldots, x_i, \ldots, x_{n-1}), (i) >$$
$$(x_0, \ldots, x_j - k, \ldots, x_i, \ldots, x_{n-1}), (j)$$
$$for \quad (0 \leq i \leq n-1) \wedge (j < i) \wedge$$
$$(0 < k \leq x_j). \qquad (6)$$
$$(x_0, \ldots, x_i, \ldots, x_j, \ldots, x_{n-1}), (i) >$$
$$(x_0, \ldots, x_i, \ldots, x_j - k, \ldots, x_{n-1}), (j)$$
$$for \quad (0 \leq i \leq n-1) \wedge (j > i) \wedge$$
$$(0 < k \leq x_j). \qquad (7)$$

Equations 5, 6, and 7 reveal that when a message traverses a negative *nonwaiting* channel, negative *waiting* channels are traversed in a monotonically decreasing order. Therefore, the second phase of the routing algorithm **MESH_ROUTE()** does not introduce any deadlock. Q.E.D.

Lemma 3 *The routing algorithm MESH_ROUTE() is deadlock free.*

Proof : In each phase of the routing algorithm MESH_ROUTE(), a different set of *waiting* channels is used. In the first phase, positive *waiting* channels are traversed by messages, while in the second phase negative *waiting* channels are used to deliver messages to their destinations. Since no deadlock arises in either phase of the routing algorithm (Lemma 1 and Lemma 2) and no cycle exists between the two phases, the routing algorithm MESH_ROUTE() is deadlock free. Q.E.D.

Lemma 4 *The routing algorithm MESH_ROUTE() is fully adaptive.*

Proof : In the first phase of the routing algorithm, a message can traverse any physical dimension link (positive and negative) that gets it closer to its destination by using *nonwaiting* channels. When the message has reached its destination in the positive direction along all dimensions, the routing algorithm starts the second phase where the message is routed in the negative direction along any dimension using either *waiting* or *nonwaiting* channels, Hence, the routing algorithm allows a message to use any physical path that exists between its source and destination nodes. Q.E.D.

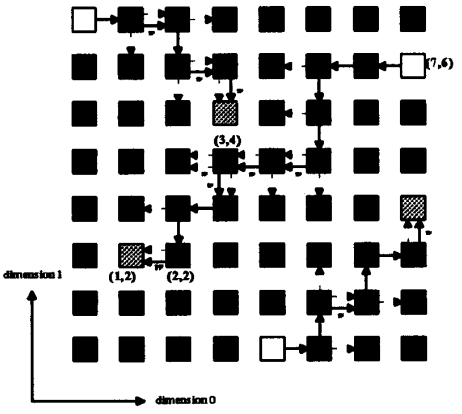

Figure 5: **Routing examples in a 8×8 mesh.**

Figure 5 shows some routing examples in an 8×8 2-dimensional mesh. In the figure, source and destination nodes are indicated by white and shaded squares respectively, and busy channels are indicated by dotted arrows. *Waiting* channels are marked with the letter w (only the required channels are shown in the figure to preserve clarity). Illustrating the routing procedure described above, we examine one of the rout-

ing examples shown in Figure 5. A message traverses any available *nonwaiting* channel that gets it closer to its destination. When all of the *nonwaiting* channels are busy, a message traverses *waiting* channels. For example, when the message whose source node is *node* $(7,6)$ and destination node is *node* $(1,2)$ reaches *node* $(3,4)$, all of the *nonwaiting* channels along both dimensions and the *waiting* channel along dimension 0 are busy. Therefore, the message traverses the free *waiting* channel along dimension 1 to get closer to its destination. In addition, when the message reaches *node* $(2,2)$, the *nonwaiting* channel that gets the message to its destination is busy. Hence, the message traverses the available *waiting* channel along dimension 0 to reach its destination.

4 Efficiency

An efficient fully adaptive routing algorithm should use any virtual path that exists between a source and a destination node. Let Q be the total number of shortest virtual paths that exist between node $X(x_0,\ldots,x_{n-1})$ and node $Y(y_0,\ldots,y_{n-1})$ in an n-dimensional mesh when 2 virtual channels per physical channel exist in the network. Hence,

$$Q = 2^{\sum_{i=0}^{n-1}\|x_i-y_i\|} \times \frac{(\sum_{i=0}^{n-1}\|x_i-y_i\|)!}{\Pi_{i=0}^{n-1}(\|x_i-y_i\|)!}. \quad (8)$$

Definition 7 : The efficiency of a routing algorithm R, ϵ_R, is defined as the ratio of the number of shortest virtual paths Q_R that the routing algorithm R can use for the delivery of messages between any two nodes in the network to the total number of shortest virtual paths Q that exist between these two nodes. Hence,

$$\epsilon_R = \frac{Q_R}{Q}, \quad and \quad 0 < \epsilon_R \leq 1. \quad (9)$$

Assuming that 2 virtual channels per physical channel exist in the network, the following shows the efficiencies of the oblivious *dimension order* routing algorithm, and an ideal routing algorithm that uses any virtual channel at each routing step.

$$\epsilon_{oblivious} = \frac{2^{\sum_{i=0}^{n-1}\|x_i-y_i\|}}{Q}. \quad (10)$$

$$\epsilon_{ideal} = 1. \quad (11)$$

Since the number of shortest virtual paths between a source node and a destination node that the routing algorithm MESH_ROUTE() allows to use is dependent on the locations of these nodes, the average efficiency $\epsilon_{MESH\_ROUTE()_{average}}$ is a better indicator of the average routing flexibility that the algorithm offers. The average efficiency of a routing algorithm R is determined by dividing the total number of shortest virtual paths among all source-destination pairs that the algorithm allows to use by the total number of shortest virtual paths that exist among all source-destination pairs.

In the following, we refer to the routing algorithms in [10][14][17] as URO (the name URO was chosen because the algorithms make use of unrestricted (U) channels, restricted (R) channels, and an oblivious (O) routing algorithm). In order to determine the total number of virtual paths among all source-destination pairs that the URO algorithm allows to use in a 2_dimensional mesh, we need to consider three cases. In the following, $X(x_0,x_1)$ is the source node and $Y(y_0,y_1)$ is the destination node. The following terms are used for calculating the total number of virtual paths between a pair of nodes.

$$\Psi = \frac{(\|x_0-y_0\|+\|x_1-y_1\|)!}{(\|x_0-y_0\|)!\times(\|x_1-y_1\|)!}. \quad (12)$$

$$\Psi_{0_k} = \frac{((\|x_0-y_0\|-k)+(\|x_1-y_1\|-1))!}{(\|x_0-y_0\|-k)!\times(\|x_1-y_1\|-1)!}. \quad (13)$$

$$\Psi_{1_k} = \frac{((\|x_0-y_0\|-1)+(\|x_1-y_1\|-k))!}{(\|x_0-y_0\|-1)!\times(\|x_1-y_1\|-k)!}. \quad (14)$$

The total number of virtual shortest paths that are allowed to be used for all source-destination pairs that satisfy the condition of case i is denoted by Q_i.

case 1: $\|x_0-y_0\| = 0$ *and* $\|x_1-y_1\| \neq 0$

$$Q_1 = 2^{\|x_1-y_1\|}. \quad (15)$$

case 2: $\|x_0-y_0\| \neq 0$ *and* $\|x_1-y_1\| = 0$

$$Q_2 = 2^{\|x_0-y_0\|}. \quad (16)$$

case 3: $\|x_0-y_0\| \neq 0$ *and* $\|x_1-y_1\| \neq 0$

$$Q_3 = \begin{cases} (\sum_{k=1}^{\|x_0-y_0\|}\Psi_{0_k}\times 2^{\|x_0-y_0\|})+ \\ (\sum_{k=1}^{\|x_1-y_1\|}\Psi_{1_k}\times 2^{\|x_0-y_0\|+k}). \end{cases} \quad (17)$$

$$\epsilon_{URO_{average}} = \frac{Q_1+Q_2+Q_3}{Q}. \quad (18)$$

There are six cases to consider for the determination of the total number of virtual shortest paths that the algorithm MESH_ROUTE() allows to use in a 2-dimensional mesh.

case 1: $\|x_0-y_0\| = 0$ *and* $\|x_1-y_1\| \neq 0$

$$Q_1 = 2^{\|x_1 - y_1\|}. \tag{19}$$

case 2: $\|x_0 - y_0\| \neq 0$ *and* $\|x_1 - y_1\| = 0$

$$Q_2 = 2^{\|x_0 - y_0\|}. \tag{20}$$

case 3: $x_0 > y_0$ *and* $x_1 > y_1$

$$Q_3 = \Psi \times 2^{\|x_0 - y_0\| + \|x_1 - y_1\|}. \tag{21}$$

case 4: $x_0 < y_0$ *and* $x_1 < y_1$

$$Q_4 = \begin{cases} (\sum_{k=1}^{\|x_0 - y_0\|} \Psi_{0_k} \times 2^{\|x_0 - y_0\|}) + \\ (\sum_{k=1}^{\|x_1 - y_1\|} \Psi_{1_k} \times 2^{\|x_0 - y_0\| + k}). \end{cases} \tag{22}$$

case 5: $x_0 > y_0$ *and* $x_1 < y_1$

$$Q_5 = \begin{cases} (\sum_{k=1}^{\|x_0 - y_0\|} \Psi_{0_k} \times 2^{k + \|x_1 - y_1\|}) + \\ (\sum_{k=1}^{\|x_1 - y_1\|} \Psi_{1_k} \times 2^{\|x_1 - y_1\|}). \end{cases} \tag{23}$$

case 6: $x_0 < y_0$ *and* $x_1 > y_1$

$$Q_6 = \begin{cases} (\sum_{k=1}^{\|x_0 - y_0\|} \Psi_{0_k} \times 2^{\|x_0 - y_0\|}) + \\ (\sum_{k=1}^{\|x_1 - y_1\|} \Psi_{1_k} \times 2^{\|x_0 - y_0\| + k}). \end{cases} \tag{24}$$

$$\epsilon_{MESH\_ROUTE()_{average}} = \frac{\sum_{i=1}^{6} Q_i}{Q}. \tag{25}$$

Figure 6 demonstrates that the new algorithm uses the virtual routes (channels) more efficiently than the URO algorithm. The figure shows that as the radices of the 2 dimensions increase the average number of virtual routes that the URO algorithm is able to utilize becomes a very small fraction of the total number of virtual routes that exit in the network. The new routing algorithm, on the other hand, is capable of using at least a quarter of those virtual routes. The implication of such increased utilization is a decrease in the blocking probability at each routing step, higher utilization of the virtual channels, and increased fault-tolerance. As the number of dimensions n increases, determining the efficiency of the new routing algorithm becomes very tedious and cumbersome. However, it will still be higher than the efficiency of the URO algorithm. In addition, the efficiency of the new algorithm in utilizing the virtual routes is higher for low dimensional meshes than for high dimensional meshes.

5 Conclusions

In this paper a fully adaptive *wormhole* routing algorithm for n-dimensional meshes was developed. The addition of one virtual channel per physical channel makes the proposed algorithm deadlock free. One virtual channel is defined as a *waiting* channel and the other as a *nonwaiting* channel. A message uses any available *nonwaiting* channel to reach its destination. When all *nonwaiting* channels are busy, a message uses *waiting* channels. The network graph is divided into 2 acyclic graphs that contain all of the physical channels of the system. *Waiting* channels are traversed in two phases. The first phase corresponds to traversing *waiting* channels belonging to one acyclic graph, while the second phase corresponds to traversing *waiting* channels belonging to the second acyclic graph. It was demonstrated that the new algorithm is more efficient in utilizing the virtual channels of the network than any algorithm with the same hardware requirement. This could provide better utilization of network resources and improved robustness without any extra cost.

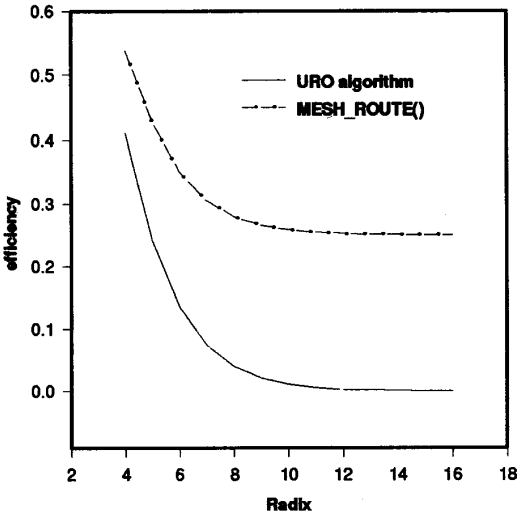

Figure 6: **Efficiency of MESH_ROUTE() and URO algorithms in a $K_0 \times K_1$ 2-dimensional mesh versus $K = K_0 = K_1$.**

There are few directions to pursue in the future. One direction is to determine the effect of different arbitration schemes (input and output scheduling policies of crossbar switches) on the performance of the al-

gorithm. A second direction is to determine the degree of fault-tolerance and effectiveness of the nonminimal and minimal versions of the proposed routing strategy in the presence of faults in systems. Finally, given an n-dimensional mesh with l virtual channels per physical channel, what is the best number of *waiting* and *nonwaiting* channels that results in high resource utilization and low blocking probability?

References

[1] R. Alverson *et al.*, "The Tera Computer System," *Proc. 1990 int'l Conf. on Supercomputing,* pp. 1-6, June 1990.

[2] M. Annaratone, M. Fillo, K. Nakabayashi, and M. Viredaz, "The K2 Parallel Processor: Architecture and Hardware Implementation," *Proc. The 17th Annual International Symposium on Computer Architecture,* Vol. 18, No. 2, pp. 92-101, June 1990.

[3] A. Agarwal, "Limits on Interconnection Network Performance," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 2, No. 4, pp. 398-412, October 1991.

[4] A. Agarwal, B. Lim, D. Kranz, and J. Kubiatowicz, "APRIL: A Processor Architecture for Multiprocessing," *Proc. The 17th Annual International Symposium on Computer Architecture,* Vol. 18, No. 2, pp. 104-114, June 1990.

[5] K. E. Batcher, "Design of a Massively Parallel Processor," *IEEE Trans. on Computers,* Vol. C-29, pp. 863-840, September 1980.

[6] Y. M. Boura and C. R. Das, "A Class of partially adaptive routing algorithms for n-dimensional meshes," *Proc. The 22sd International Conference on Parallel Processing,* Vol. 3, pp. 175-182, August 1993.

[7] A. A. Chien and J. H. Kim, " Planar-adaptive Routing: Low-cost Adaptive Networks for Multiprocessors," *Proc. The 19th Annual International Symposium on Computer Architecture,* pp. 268-277, May 1992.

[8] W. J. Dally, "Virtual channel flow control," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 3, No. 2, pp. 194-205, March 1992.

[9] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers,* Vol. C-36, pp. 547-553, May 1987.

[10] J. Duato,"A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems,* pp. 1320-1331, December 1993.

[11] C. J. Glass and L. M. Ni, " The turn model for adaptive routing," *Proc. The 19th Annual International Symposium on Computer Architecture,* pp. 278-287. May 1992.

[12] Intel Corporation, *A Touchstone DELTA System Description,* 1990.

[13] D. Lenoski, J. laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam, "The Stanford Dash Multiprocessor," *IEEE Computer,* pp. 63-79, March 1992.

[14] X. Lin, P. K. Mckinley, and L. M. Lin, "The message flow model for routing in wormhole-routed networks," *Proc. The 22sd International Conference on Paralle Processing,* Vol. 1, pp. 294-297, August 1993.

[15] D. H. Linder and J. C. Harden, "An adaptive and fault-tolerant wormhole routing strategy for K-ary n-cubes," *IEEE Transactions on Computers,* Vol. C-40, pp. 178-186, January 1991.

[16] NCUBE Company, *NCUBE 6400 Processor Manual,* 1990.

[17] C. Su and K. G. Shin, "Adaptive deadlock-free routing in multicomputers using only one extra channel," *Proc. The 22sd International Conference on Paralle Processing,* Vol. 3, pp. 175-182, August 1993.

[18] H. Sullivan and T. R. Bashkow, " A large scale, homogenous, fully distributed parallel machine," *Proc. The 4th Annual International Symposium on Computer Architecture,* pp. 105-117, May 1977.

[19] G. Zorpetta, "Technology 1991: Minis and Mainframes," *IEEE Spectrum,* pp. 40-43, January 1991.