Adaptive Deadlock- and Livelock-Free Routing With All Minimal Paths in Torus Networks

Pablo E. Berman *†

Luis Gravano †‡
e-mail: gravano@buevm2.vnet.ibm.com

Gustavo D. Pifarré †‡§
e-mail: pifarre@buevm2.vnet.ibm.com

Jorge L. C. Sanz ‡†
e-mail: sanz@ibm.com

Abstract

This paper consists of two parts. In the first part, a new algorithm for deadlock- and livelock-free routing for the n-dimensional torus network is presented. This algorithm, called **-Channels*, is *fully-adaptive minimal*, i.e. all paths with a minimal number of hops from source to destination are available for routing. **-Channels* works for messages of unknown size, thus yielding new routing techniques for both packet-switched and *worm-hole* models. **-Channels* differs radically from the packet-switched fully-adaptive minimal methods presented in SPAA '91 by Pifarré, Gravano, Felperin, and Sanz [PGFS91]. In particular, the packet-based techniques in [PGFS91] do not work for worm-hole routing as deadlock situations can be constructed.

**-Channels* requires only five virtual channels per bidirectional link of the n-dimensional torus. In fact, only three virtual channels are necessary for the links in one of the dimensions, thus yielding a total of $10(n-1)+6$ buffers per node.

This bound gives the smallest number of channels known in the literature for fully-adaptive minimal *worm-hole* routing. Previous algorithms for fully-adaptive worm-hole routing in n-dimensional tori require more than $O(2^{n-1})$ virtual channels per bidirectional link (see [LH91]). In addition, these results also yield the smallest number of buffers known in the literature for packet-switched fully-adaptive minimal routing.

In the second part of this paper, a comparison of four worm-hole techniques in the 2-dimensional torus is shown in terms of activity in the routing nodes and experimental performance evaluation. Simulation results on the performance of the four worm-hole algorithms are shown for dynamic injection models. Meaningful comparisons required the equalization of the number of virtual channels for all of the techniques, and this is accomplished by resorting to the concept of lane-channels introduced in [Dal90]. The performance of these schemes is measured for different traffic models: random and bit-reversal. Two worm lengths are tried.

1 Introduction.

Message routing in large interconnection networks has attracted a great deal of interest in recent years. Different underlying machine models and hardware architectures have been used and proposed [DS87], [RBJ88, Ran85], [Upf89, LM89], [Val88], [KS90], [NS87], [Hil85], [CEDK91], [BCC+88], [LLK+91].

A desirable feature of routing algorithms is adaptivity, i.e., the ability of messages to use alternative paths toward their destinations according to traffic congestion in the nodes of the network. The amount of hardware resources grow with the degree of adaptivity desired, and resources may become critical if deadlock and live-

* ESLAI, Escuela Superior Latino Americana de Informática, CC 3193,(1000) Buenos Aires, Argentina.

† Computer Research and Advanced Applications Group, IBM Argentina, Ing. E. Butty 275, (1300) Buenos Aires, Argentina.

‡ Computer Science Dept., IBM Almaden Research Center, San José, California.

§ Departamento de Computación, Fac. de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina.

SPAA '92 - 6/92/CA
©1992 ACM 0-89791-484-8/92/0006/0003 ..... $1.50

lock freedom in a deterministic sense are also needed.

The recent work reported in [KS90] shows an adaptive deadlock-free routing algorithm dubbed *Chaos*. The method has a non-zero probability that a message will not reach its destination after t routing steps, for an arbitrary t. However, this probability tends to zero as t approaches infinity. Furthermore, the technique in [KS90] and [BS91] applies only to packet and virtual cut-through routing and paths followed by the messages are not necessarily minimal.

Restricting the set of available paths in the network to a subset suitably chosen is a common way to reduce the hardware resources necessary for deadlock-free routing. When stringent restrictions are applied, oblivious algorithms or methods with partial adaptivity will be obtained. Oblivious algorithms have been studied thoroughly for meshes and tori [Lei90]. In [Lei90], the performance of these packet routing algorithms for meshes has been analyzed, for both static and dynamic injection models, and for both packet switching and virtual cut-through techniques. Recently, some mathematical analyses have been reported on the performance of worm-hole oblivious algorithms [RU91]. On the other hand, if few restrictions are imposed on the set of possible routes generated by a routing function, impractical algorithms may result.

A *fully-adaptive minimal* routing scheme is one in which all possible minimal paths between a source and a destination are of potential use at the time messages are injected into the network. Paths followed by the messages depend on the traffic congestion found in the nodes of the network. Full-adaptivity is a feature from which one can hope to obtain the best possible performance if no source of randomization is used. Full-adaptivity has been used by Upfal in [Upf89] to produce a deterministic optimal algorithm for routing in the multibutterfly.

In [PGFS91], fully-adaptive algorithms for the hypercube and mesh have been presented for packet-switching routing. These algorithms are deadlock-free, and can be implemented using only two queues per node. Recently, the principles shown in [PGFS91] have been used for the same routing model in n-dimensional torus networks by using three queues per node [CG92a], and this result is optimal for the given model and network [CG92b]. Unfortunately, the methodology of [PGFS91] and [CG92a] does not apply to worm-hole routing.

On the other hand, new algorithms for deadlock-free worm-hole routing have been reported in [LH91], [DA90], [Dal90], [GPFS91], and [Dua91]. The algorithms in [LH91] are for k-ary n-dimensional cubes and n-dimensional mesh-connected networks. These techniques need a number of virtual channels per link that increases exponentially with n: in a k-ary n-cube 2^{n-1} virtual networks are needed, each with $n+1$ copies of the network. In [DA90], a method for deadlock-free adaptive routing in k-ary n-dimensional cubes is presented, which can be used for worm-hole routing. In [Dal90], a technique based on the use of multiple independent *lanes* associated with each physical link in a routing node is shown. Given a fixed amount of storage space allocated to each physical channel, it is shown that breaking the storage into several buffers is a convenient methodology for improving network performance. In [GPFS91], fully-adaptive worm-hole algorithms were introduced for a variety of networks, including the hypercube. Later, the same fully-adaptive worm-hole algorithm for the hypercube was independently presented in [Dua91].

In Section 3, a new algorithm for routing on toroidal networks is presented [GPFS91]. This technique, called **-Channels*, is fully-adaptive, deadlock- and livelock-free, and requires a very moderate amount of resources in the routing nodes. The new method is presented for n-dimensional tori. **-Channels* works for messages of unknown size, thus rendering new routing techniques for both *packet-switched* and *worm-hole* models. It requires 5 virtual channels per bidirectional link of an n-dimensional torus. This algorithm is the first one with these characteristics, i.e., fully-adaptive minimal and deterministically livelock- and deadlock-free, that requires a number of virtual channels per link that does not depend on n or the size of the network. Actually, the number of virtual channels can be made equal to 3 for one of the dimensions. Thus, the total number of virtual channels per node is $10(n-1)+6$ for an n-dimensional torus. This count compares very favorably with that of the techniques presented in [LH91], as explained above. An instance of **-Channels* yields a fully-adaptive minimal deadlock-free worm-hole routing algorithm for the bidimensional torus using 3 virtual channels per bidirectional link associated with the X dimension, and 5 virtual channels per bidirectional link associated with the Y dimension.

In addition, **-Channels* also yields the smallest number of buffers known in the literature for packet-switched fully-adaptive minimal routing. For example, a total of 16 buffers are needed in the node of a 2-dimensional torus, and 26 buffers in a 3-dimensional torus. Thus, **-Channels* offers an appealing approach

to practical implementations of packet-switched low-dimensional toroidal networks because of its reduced storage requirement, ensuring freedom from deadlock and livelock *deterministically* without any source of randomization, and yet allowing all minimal paths for routing. Another remarkable characteristic of **-Channels* for the packet model is that *no central queue is necessary* to guarantee any of its properties. These properties make **-Channels* a practical alternative to chaotic routing [BS91], [KS91] for 2-dimensional and 3-dimensional adaptive torus networks.

Section 4 presents a comparison of worm-hole algorithms for the 2-dimensional torus. Four algorithms will be analyzed and compared: **-Channels*, *4-Classes*, *Linder-Harden's*, and *Oblivious*. Algorithm **-Channels* will be presented in Section 3, as mentioned above. *4-Classes* is a fully-adaptive minimal routing algorithm for the 2-dimensional torus that was described in [FGPS91]. Although *4-Classes* uses more virtual channels than **-Channels*, it allows a node model design with potentially more parallel logic than that in **-Channels*. *Linder-Harden's* algorithm [LH91], is fully-adaptive minimal, but uses more virtual channels than **-Channels* and *4-Classes*. *Oblivious* [DS87], is *oblivious* and *non-minimal*, i.e. the path followed by a message is completely determined by its source and destination, and the number of hops of this path may be greater than that of a minimal path. A node model for each of the algorithms and node activity are presented. For a fair comparison of the four algorithms, the number of virtual channels used by the different techniques is equalized.

2 Definitions.

Definition 2.1 *An* n-dimensional k-torus *(sometimes referred to as k-ary n-cube) is a network with* k^n *nodes. The* n *dimensions of the n-dimensional torus will be referred to as* $X_{n-1}, \ldots, X_0$. *Each node of the torus will be denoted by a tuple* $(x_{n-1}, \ldots, x_0)$, *with* $0 \leq x_i < k$ *for all* $0 \leq i < n$, *and will be connected to nodes* $(x_{n-1}, \ldots, x_{i+1}, (x_i + 1) \bmod k, x_{i-1}, \ldots, x_0)$, *and* $(x_{n-1}, \ldots, x_{i+1}, (x_i - 1) \bmod k, x_{i-1}, \ldots, x_0)$, *for all* $0 \leq i < n$.

The link connecting nodes $(x_{n-1}, \ldots, k-1, \ldots, x_0)$ and $(x_{n-1}, \ldots, 0, \ldots, x_0)$ along dimension X_i will be called a wrap-around link along dimension X_i. The directed link $((x_{n-1}, \ldots, x_i, \ldots, x_0), (x_{n-1}, \ldots, (x_i + 1) \bmod k, \ldots, x_0))$ corresponds to dimension X_i, and will be referred to as having orientation X_i^+. Analogously, link $((x_{n-1}, \ldots, x_i, \ldots, x_0), (x_{n-1}, \ldots, (x_i - 1) \bmod k, \ldots, x_0))$, corresponding to dimension X_i, will be referred to as having orientation X_i^-. Dimensions X_1 and X_0 of a 2-dimensional torus will often be referred to as X and Y, respectively.

3 Algorithm **-Channels.*

In this Section, a fully-adaptive, minimal, deadlock-free, worm-hole routing algorithm for the n-dimensional k-torus will be presented. This routing algorithm requires 5 virtual channels per bidirectional link for its implementation [1], a number that does not depend on the size or dimension of the torus, and allows each message to choose *adaptively*, step by step, among *all* the minimal paths that take it to its destination. No minimal path is discarded in order to obtain freedom from deadlock. This algorithm will be referred to as **-Channels*. A simplified routing node model is depicted in Figure 1 for the 2-dimensional torus.

The central idea in **-Channels* is simple. Figure 1 shows that there are basically two types of virtual channels, as will be explained below: *star* channels, and *non-star* channels. Messages will move through the star channels as when doing *dimension-order* oblivious routing [DS87]. The non-star channels will be used when taking any of the transitions that would not be allowed by the dimension-order routing algorithm, thus obtaining full adaptivity while preserving freedom from deadlock. A more detailed description of the algorithm follows.

Consider the directed link:

$$((x_{n-1}, \ldots, x_i, \ldots, x_0), (x_{n-1}, \ldots, (x_i+1) \bmod k, \ldots, x_0)).$$

This link will have three virtual channels associated with it:

- $c^*_{i,+,0,(x_{n-1},\ldots,(x_i+1) \bmod k,\ldots,x_0)}$,
- $c^*_{i,+,1,(x_{n-1},\ldots,(x_i+1) \bmod k,\ldots,x_0)}$, and
- $c_{i,+,(x_{n-1},\ldots,(x_i+1) \bmod k,\ldots,x_0)}$.

Analogously, link:

$$((x_{n-1}, \ldots, x_i, \ldots, x_0), (x_{n-1}, \ldots, (x_i-1) \bmod k, \ldots, x_0))$$

[1] In fact, only 3 virtual channels are needed for each bidirectional link associated with the most significant of the dimensions of the torus network.

will have three virtual channels associated:

- $c^*_{i,-,0,(x_{n-1},\ldots,(x_i-1) \bmod k,\ldots,x_0)}$,
- $c^*_{i,-,1,(x_{n-1},\ldots,(x_i-1) \bmod k,\ldots,x_0)}$, and
- $c_{i,-,(x_{n-1},\ldots,(x_i-1) \bmod k,\ldots,x_0)}$.

Channels:

- $c^*_{i,+,0,(x_{n-1},\ldots,(x_i+1) \bmod k,\ldots,x_0)}$,
- $c^*_{i,+,1,(x_{n-1},\ldots,(x_i+1) \bmod k,\ldots,x_0)}$,
- $c^*_{i,-,0,(x_{n-1},\ldots,(x_i-1) \bmod k,\ldots,x_0)}$, and
- $c^*_{i,-,1,(x_{n-1},\ldots,(x_i-1) \bmod k,\ldots,x_0)}$

will be referred to as *star* channels.

In both cases, the star channels will be used as when doing dimension-order routing [DS87]: messages will move through star channels with prefix $i,+,0$ while correcting dimension X_i following orientation X_i^+ before taking a wrap-around link along dimension X_i. After taking a wrap-around along dimension X_i following orientation X_i^+, messages will move through star channels with prefix $i,+,1$ when correcting dimension X_i.

Now, the routing function $\mathcal{R}$: *VirtualChannels* $\times$ *Nodes* $\rightarrow \mathcal{P}$(*VirtualChannels*) will be described, where $\mathcal{P}$(*VirtualChannels*) is the powerset of the set of *VirtualChannels*. It is supposed that each node p has an *injection channel* i_p where p injects the new messages into the network. Whenever a message arrives at a channel incident with its destination node, it is delivered and taken out of the network.

The routing function. [2]

$$\mathcal{R}\ (c_x, y) =$$

$$\begin{cases} c^*_{i,+,0,(x_{n-1},\ldots,x_i+1,x_i+1,x_{i-1},\ldots,x_0)} \\ \quad \text{if } x_j = y_j\ \forall j = n-1,\ldots,i+1 \text{ and} \\ \quad x_i \neq y_i \text{ and } x_i \neq k-1 \text{ and} \\ \quad \text{the message has not taken} \\ \quad \text{a wrap-around along} \\ \quad \text{dimension } X_i \text{ yet and} \\ \quad \text{dimension } X_i \text{ should be corrected} \\ \quad \text{following } X_i^+ \\ c^*_{i,+,1,(x_{n-1},\ldots,x_i+1,0,x_{i-1},\ldots,x_0)} \\ \quad \text{if } x_j = y_j\ \forall j = n-1,\ldots,i+1 \text{ and} \\ \quad x_i \neq y_i \text{ and } x_i = k-1 \text{ and} \\ \quad \text{dimension } X_i \text{ should be corrected} \\ \quad \text{following } X_i^+ \\ c^*_{i,+,1,(x_{n-1},\ldots,x_i+1,x_i+1,x_{i-1},\ldots,x_0)} \\ \quad \text{if } x_j = y_j\ \forall j = n-1,\ldots,i+1 \text{ and} \\ \quad x_i \neq y_i \text{ and the message has already} \\ \quad \text{taken a wrap-around} \\ \quad \text{along dimension } X_i \text{ and} \\ \quad \text{dimension } X_i \text{ should be corrected} \\ \quad \text{following } X_i^+ \\ c_{j,+,(x_{n-1},\ldots,x_j+1,(x_j+1) \bmod k,x_{j-1},\ldots,x_0)} \\ \quad \text{if } x_j \neq y_j \\ \quad \text{and dimension } X_j \text{ should be corrected} \\ \quad \text{following } X_j^+;\ j \in \{n-1,\ldots,0\} \end{cases}$$

where c_x is a virtual channel incident to node $x = (x_{n-1},\ldots,x_0)$, and $y = (y_{n-1},\ldots,y_0)$. In the definition above, only the equations for correcting each dimension following the "+" orientation have been described.

Therefore, a message will be allowed to correct *any* of the dimensions that need correction through non-star channels. A message m will be allowed to enter a star channel corresponding to dimension X_i only if X_i is the most significant dimension the message needs to correct, and only if the star channel corresponds to the message's having taken a wrap-around along that dimension or not, as explained above.

Consequently, the resulting worm-hole routing algorithm is fully-adaptive minimal. Moreover, assuming that virtual channels are assigned fairly, and that messages are of finite but arbitrary length, this routing algorithm is free of deadlock. Star channels play the most important role in proving this.

The following lemma has been proven in [GPFS91].

[2] In fact, $\mathcal{R}\ (c_x, y)$ is the set of all the virtual channels above whose corresponding condition on the right hand size is true.

Lemma 3.1 *The routing algorithm that results from routing function $\mathcal{R}$ is free of deadlock.*

From Lemma 3.1 and $\mathcal{R}$, it follows that:

Theorem 3.2 *The* *-Channels *algorithm yields a worm-hole routing technique for any n-dimensional torus. The technique is fully-adaptive, minimal, free of deadlock and livelock in a deterministic sense, and can be implemented using 5 virtual channels per bidirectional physical link in all but one of the dimensions where only 3 channels suffice. Thus, the total number of virtual channels per node is* $10(n-1)+6$.

In fact, the above presentation yields an algorithm with 6 virtual channels per physical link. However, only 5 virtual channels per bidirectional link are necessary. This is so because the star channels $c^*_{i,+,1,(x_{n-1},\ldots,x_i,\ldots,x_0)}$ with $x_i \geq \lfloor k/2 \rfloor$ are never used, because paths are minimal, and so, each message travels at most $\lfloor k/2 \rfloor$ steps along each dimension. Channels $c^*_{i,+,1,(x_{n-1},\ldots,x_i,\ldots,x_0)}$ with $x_i \geq \lfloor k/2 \rfloor$ would be used by messages that had taken a wrap-around along dimension X_i and orientation X_i^+, and moved at least $\lfloor k/2 \rfloor - 1$ steps along X_i^+ after the wrap-around. So, entering such a channel would involve traveling more than $\lfloor k/2 \rfloor$ steps along dimension X_i.

Analogously, channels $c^*_{i,-,1,(x_{n-1},\ldots,x_i,\ldots,x_0)}$ with $x_i < \lfloor k/2 \rfloor$ are never used.

So, only 5 virtual channel per bidirectional link are needed, because channels $c^*_{i,+,0,(x_{n-1},\ldots,0,\ldots,x_0)}$ and $c^*_{i,-,0,(x_{n-1},\ldots,k-1,\ldots,x_0)}$ are never used. Furthermore, dimension X_{n-1}, the most significant one, does not need non-star channels. So, each bidirectional link corresponding to dimension X_{n-1} needs only 3 virtual channels.

As a result of this, the algorithm above instantiated to a 2-dimensional torus needs 3 virtual channels for each bidirectional link associated with dimension X, and 5 virtual channels for each bidirectional link associated with dimension Y.

The above theorem was presented in the context of a worm-hole model. In this model, messages serviced by the network are of unknown size. Therefore, the design of the routing node cannot be based on the assumption that messages will be completely stored in a node. This fact is because the model does not assume a known bound for the message size, and also because the required storage could become prohibitively large. In case a packet implementation is desired, **-Channels* also yields the following important and immediate result:

Theorem 3.3 *The* *-Channels *algorithm yields a packet routing technique for any n-dimensional torus. The technique is fully-adaptive, minimal, free of deadlock and livelock in a deterministic sense, and can be implemented using 5 buffers, in each node, per bidirectional physical link in all but one of the dimensions where only 3 buffers per link suffice. Thus, the total number of buffers per node is* $10(n-1)+6$. *Furthermore, the packet routing technique does not require any central queue mechanism for ensuring the above properties.*

As no central queue is mandatory, **-Channels* can be used for *combined routing*, i.e., both short fixed-size packets, and long messages of unknown size are handled by the same network and routing resources. While this type of combined routing can be naturally handled in multistage adaptive networks such as the multibutterflies [KU91], fully-adaptive combined routing remained as a difficult goal for toroidal networks before **-Channels* was discovered. Combined routing is accomplished in **-Channels* by a "partial" virtual cut-through implementation, i.e. the storage of long messages would span more than one routing node while short messages may be buffered completely in one node. Practical studies are being carried out to experimentally determine the performance of this combined routing, and the potential improvements offered by a central queuing mechanism for enhanced routing of the fixed-size packets.

4 Node models and simulations.

In this Section, a comparison of four different routing algorithms for the 2-dimensional torus network will be presented. The four algorithms studied in this Section are **-Channels* (see Section 3), *4-Classes* [FGPS91], *Linder-Harden's* [LH91], and *Oblivious* [DS87]. The first three algorithms are fully-adaptive minimal, whereas the last one is oblivious and non-minimal. These algorithms require 16, 32, 36, and 8 virtual channels per node, respectively, for their deadlock-free implementation. The comparison is carried out by equalizing the number of virtual channels used by the different routing techniques. This task is accomplished by adding *lanes*

to the algorithms originally requiring fewer virtual channels per physical link. In the scope of this paper, a *lane* in a physical link is simply a virtual channel that plays an identical role to one of the original virtual channels associated with the same link [Dal90]. Lanes are known to play a central role in enhancing the throughput of oblivious routing worm-hole algorithms [Dal90]. Thus, more than one message can be simultaneously using the same virtual channel by occupying different lanes associated with the virtual channel. For example, in order to compare *Oblivious* with *4-Classes*, 6 lanes will be added per physical link, thus matching the 32 virtual channels of *4-Classes*. Furthermore, notice that lanes are added in a way that crossbars grow with the total number of inputs.

The analysis of algorithms is carried out in two directions. First, routing node models are proposed for each technique. Second, simulations on their performance are shown for a network of 961 nodes.

4.1 Node models.

For some of the algorithms being compared, a partition of all of the buffers can be found such that the connections that the routing function performs are only between buffers that belong to the same set. Then, each set requires a crossbar that is independent of the others (see, for example, the node model of *4-Classes* and that of *Linder-Harden's* algorithm). It is assumed that each crossbar can set one new connection from its inputs to its outputs per cycle [KS91], and this is executed independently for each of the other crossbars in the same node. Therefore, the more independent crossbars each node has, the more parallelism within each switch is achieved. *Oblivious* is allowed to make all possible connections (depending on availability of output buffer space) in its crossbar in each routing cycle, because this crossbar is oblivious, and therefore simpler than the others.

4.1.1 Algorithm **-Channels.*

In Figure 1, a node model for **-Channels* is presented. Only 8 input and 8 output buffers per node are needed. The node also has an injection buffer and a delivery buffer, thus making a total of 9 the number of inputs and outputs of the adaptive crossbar, because adaptive connectivity between all of the inputs and all of the outputs of a node is required. This adaptive crossbar is drawn as a box in the middle of Figure 1. This is a mechanism to connect input frames to output frames adaptively which is functionally similar to a crossbar for oblivious routing, except that more than one useful output buffer, if available, may be used for potential progressing of a message toward its destination. An analogous input-to-output frame connectivity is used in other routers such as the Chaos [BS91].

Lanes are added to the virtual channels, as explained above. Therefore, each node will have an adaptive 17×17 crossbar. The size of all of these buffers depends on whether **-Channels* is intended to work for worm-hole or for packets of fixed and small size. In the latter, the size of the buffers will be identical to the packet size, typically 150-300 bits. The inputs to the adaptive crossbar and physical channel arbitration are handled with fairness to avoid starvation.

4.1.2 Algorithm *4-Classes.*

This algorithm requires four virtual networks, each of which is split into two levels. Therefore, there are eight levels. Virtual channels corresponding to one level are independent of those of other levels. Thus, 8 adaptive 3×3-crossbars are required. Figure 2 shows the node model for this algorithm. 2 input buffers, 2 output buffers and a 3×3-crossbar for each level, add up to 16 input buffers, 16 output buffers and 8 crossbars per node. Multiplexers connect the injection buffer to the 8 crossbars and the 8 crossbars to the delivery buffer.

4.1.3 *Linder-Harden's* Algorithm.

Figure 3 shows the node model for this algorithm. Each node has 36 buffers and 6 adaptive 4×4 crossbars, assigned as follows: 1 crossbar, 3 input buffers and 3 output buffers per independent level. Two multiplexers connect the injection buffer to the crossbars and the crossbars to the delivery buffer.

4.1.4 Algorithm *Oblivious.*

For this routing algorithm, the set of buffers cannot be split into independent classes. Therefore, the node model has a single crossbar (see Figure 4). Then, eight buffers and one 5×5-crossbar [DS86] are needed. 12 input and 12 output buffers are added as lanes to the virtual channels, as mentioned above. Adding buffers

to implement lanes gives this algorithm some degree of adaptivity, since messages can take different lanes depending on traffic congestion.

4.2 Simulation results.

Each routing algorithm was simulated according to the high-level node models sketched above. Several continuous injection simulations were tried. These experiments involved two different message sizes, namely worms of 15 and 31 flits, and two different traffic patterns: random traffic and bit-reversal traffic, on a 2-dimensional torus with 961 nodes. Under bit-reversal traffic, node $((x_p \dots x_0),(y_p \dots y_0))$ sends all of its messages to node $((y_0 \dots y_p),(x_0 \dots x_p))$, where $p = \lceil \log_2(k) \rceil$ [DA90].

Figures 5, 6, and 7 show the mean latency as a function of the achieved throughput, for the four routing algorithms and the different traffic patterns and worm lengths. The results corresponding to bit-reversal traffic for 31-flit worms are similar to those for 15-flit worms.

From the experimental performance study, several conclusions can be drawn. First, **-Channels* showed better performance than *Oblivious* for all of the communication patterns and at all of the applied loads tried. This comparison is "fair" because *Oblivious* was allowed to make all possible connections in its oblivious crossbar in each routing cycle, while in the adaptive crossbar of **-Channels* only one connection per cycle was allowed. Furthermore, the number of inputs and outputs to their crossbars was equalized.

It is important to remark that **-Channels* outperformed *Oblivious* even for random traffic, and did so with a large performance gap. This gap is not commonly encountered when comparing adaptive and oblivious packet-switching routers [BS91],[FLBS91],[KS91]. This observation holds true in spite of increasing the number of extra lanes in the oblivious router to yield a total of 32 buffers per node.

While the experimental results show that **-Channels* also outperformed the other two routers, this comparison would be "unfair". The reason is that even though all routers have been equalized with respect to the number of virtual channels, the resulting adaptive crossbars have *different* sizes. Thus, *Linder-Harden's* algorithm is based on a 4×4 adaptive crossbar, and *4-Classes* on a 3×3 crossbar, while **-Channels* uses an adaptive crossbar connecting all inputs to all outputs.

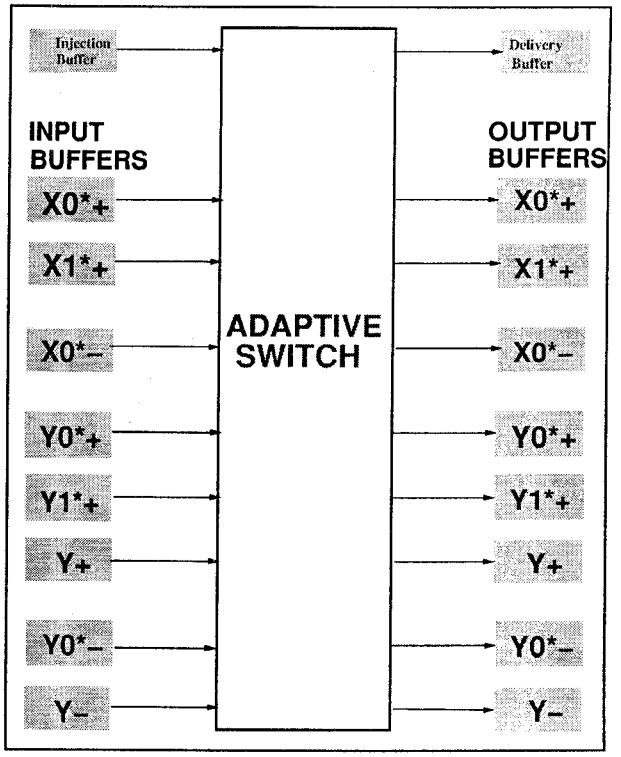

Figure 1: Node Model for Algorithm **-Channels.*

On the other hand, *4-Classes* outperforms *Linder-Harden's* and the comparison is "fair" because the size of the adaptive crossbar in the former is smaller than in the latter. The result of this comparison is consistent for all message sizes tried, all patterns of communication, and all applied loads.

References

[BCC+88] S. Borkar, R. Cohn, G. Cox, S. Gleason, T. Gross, H.T. Kung, et al. iWarp: an integrated solution to high-speed parallel computing. In *Proceedings of Supercomputing 88.* ACM, 1988.

[BS91] K. Bolding and L. Snyder. Mesh and torus chaotic routing. UW CS91-04-04, University of Washington, 1991. To appear in the MIT/Brown Advanced Research in VLSI and Parallel Systems Conference, March 1992.

[CEDK91] F. Chong, E. Egozy, A. DeHon, and T. Knight. Multipath fault tolerance in multistage interconnection networks. Transit note #48, MIT, June 1991.

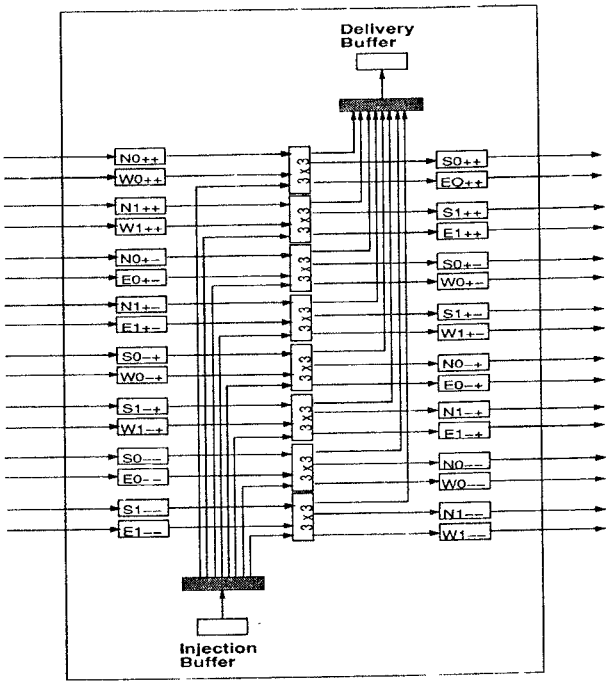

Figure 2: Node Model for Algorithm *4-Classes*.

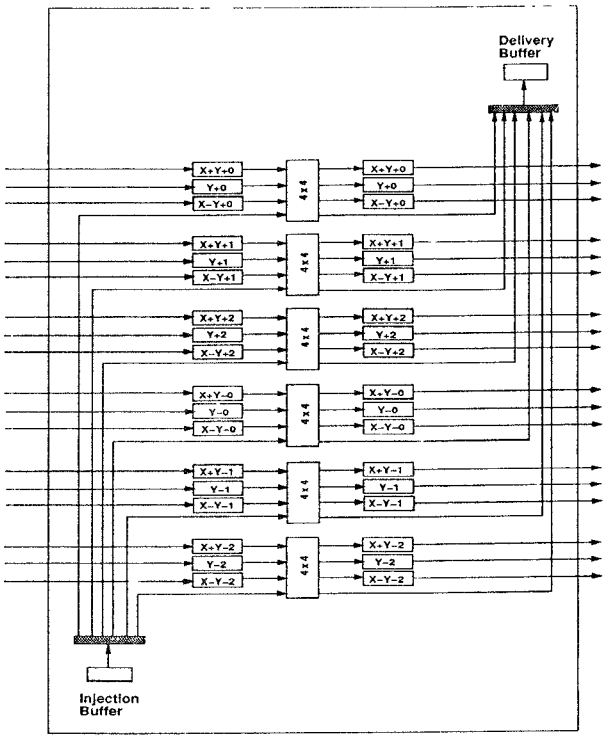

Figure 3: Node Model for *Linder-Harden's* Algorithm.

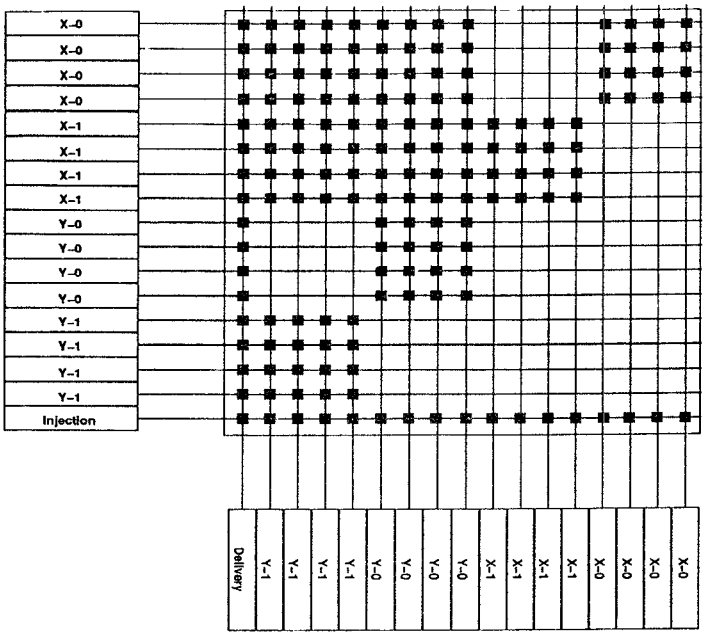

Figure 4: Node Model for Algorithm *Oblivious*.

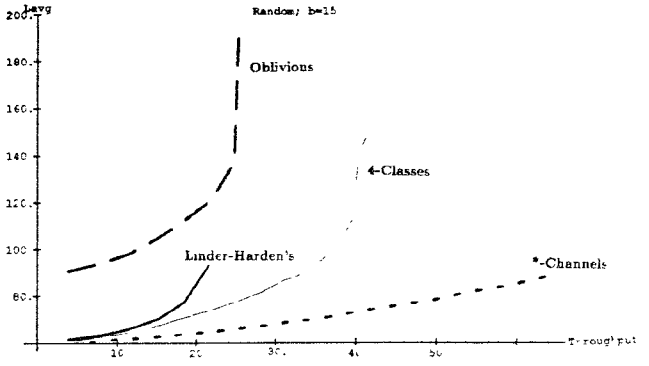

Figure 5: Results for *Random Routing* with worms of length 15.

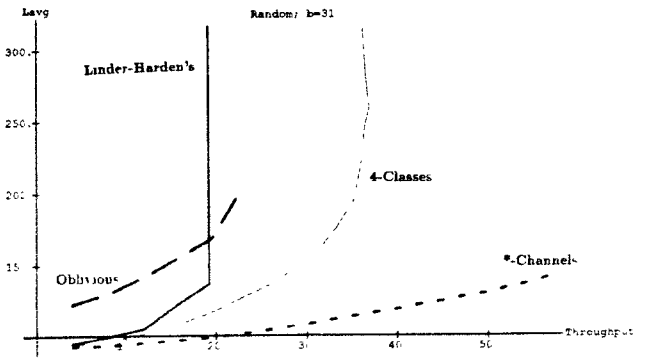

Figure 6: Results for *Random Routing* with worms of length 31.

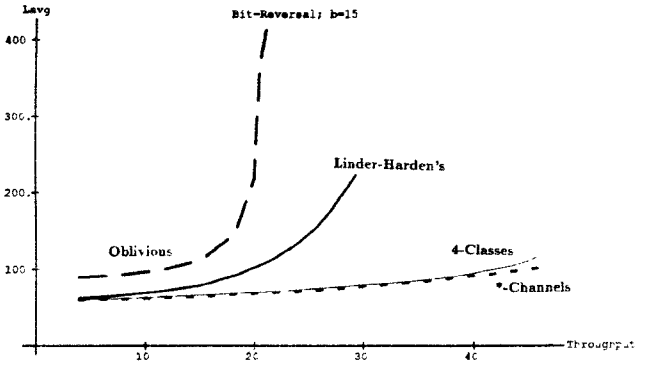

Figure 7: Results for *Bit-Reversal Routing* with worms of length 15.

[CG92a] R. Cypher and L. Gravano. Adaptive deadlock-free packet routing in torus networks with minimal storage. RJ:8571 (77350), IBM Almaden Research Center, January 1992. To be presented in ICPP '92.

[CG92b] R. Cypher and L. Gravano. Requirements for deadlock-free, adaptive packet routing. Technical report, IBM Almaden Research Center, February 1992. To be presented in PODC '92.

[DA90] W. J. Dally and H. Aoki. Adaptive Routing using Virtual Channels. Technical report, MIT, 1990.

[Dal90] W. J. Dally. Virtual-Channel Flow Control. In *The 17^{th} Annual International Symposium on Computer Architecture*, May 1990.

[DS86] W. J. Dally and C. L. Seitz. The Torus Routing Chip. *Distributed Computing*, pages 187–196, 1986.

[DS87] W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36:547–553, May 1987.

[Dua91] J. Duato. Deadlock-free adaptive routing algorithms for multicomputers: Evaluation of a new algorithm. In *Proceedings of the 3^{rd} IEEE Symposium on Parallel and Distributed Processing*. IEEE, December 1991.

[FGPS91] S. A. Felperin, L. Gravano, G. D. Pifarré, and J. L. C. Sanz. Fully-adaptive routing: Packet switching performance and worm-hole algorithms. In *Supercomputing*, 1991.

[FLBS91] S.A. Felperin, H. Laffitte, G. Buranits, and J.L.C. Sanz. Deadlock-free minimal packet routing in the torus network. TR:91-22, IBM Argentina, CRAAG, 1991.

[GPFS91] L. Gravano, G.D. Pifarré, S. A. Felperin, and J.L.C. Sanz. Adaptive deadlock-free worm-hole routing with all minimal paths. TR:91-21, IBM Argentina, CRAAG, August 1991.

[Hil85] D. Hillis. *The Connection Machine*. The MIT Press, 1985.

[KS90] S. Konstantinidou and L. Snyder. The Chaos router: A practical application of randomization in network routing. In *2nd. Annual ACM SPAA*, pages 21–30, 1990.

[KS91] S. Konstantinidou and L. Snyder. Chaos Router: Architecture and performance. In 18^{th} *International Symposium on Computer Architecture*, pages 212–221. IEEE, May 1991.

[KU91] S. Konstantinidou and E. Upfal. Experimental comparison of multistage interconnection networks. RJ:8451 (76459), IBM Almaden Research Center, November 1991.

[Lei90] T. Leighton. Average Case Analysis of Greedy Routing Algorithms on Arrays. In *SPAA*, 1990.

[LH91] D.H. Linder and J.C. Harden. An Adaptive and Fault Tolerant Wormhole Routing Strategy for k-ary n-cubes. *IEEE Transactions on Computers*, 40(1):2–12, January 1991.

[LLK+91] D. Lenoski, J. Laudon, Garachorloo K., A. Gupta, W. Weber, and J. Hennessy. Overview and status of the dash multiprocessor. In *International Symposium on Shared Memory Multiprocessing*. Tokyo, Japan, 1991.

[LM89] T. Leighton and B. Maggs. Expanders might be practical: Fast algorithms for routing around faults on multibutterflies. In IEEE, editor, 30^{th} *Annual Symposium on Foundations of Computer Science*, pages 384–389, October 1989.

[NS87] J.Y. Ngai and C.L. Seitz. A framework for adaptive routing. 5246:TR:87, Computer Science Department, California Institute of Technology, 1987.

[PGFS91] G.D. Pifarré, L. Gravano, S.A. Felperin, and J.L.C. Sanz. Fully-Adaptive Minimal Deadlock-Free Packet Routing in Hypercubes, Meshes, and Other Networks. In *Proceedings of the* 3^{rd} *Annual ACM Symposium on Parallel Algorithms and Architectures*, 1991.

[Ran85] A.G. Ranade. How to emulate shared memory. In *Foundations of Computer Science*, pages 185 – 194, 1985.

[RBJ88] A.G. Ranade, S.N. Bhat, and S.L. Johnson. The Fluent Abstract Machine. In J. Allen and F.T. Leighton, editors, *Fifth MIT conference on advanced research in VLSI*, pages 71 – 93. The MIT press, March 1988.

[RU91] P. Raghavan and E. Upfal. A theory of wormhole routing in parallel computers. Technical report, IBM Research, December 1991.

[Upf89] E. Upfal. An $\mathbf{O}(\log N)$ deterministic packet routing scheme. In 21^{st} *Annual ACM-SIGACT Symposium on Theory of Computing*, May 1989.

[Val88] L.G. Valiant. General purpose parallel architectures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North-Holland, 1988.