# Adaptive Deadlock-free Worm-hole Routing in Hypercubes.

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

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

Gustavo Denicolay ‡*

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

## Abstract

*In this paper, two new algorithms for worm-hole routing in the hypercube are presented* [1]. *The hypercube algorithm of Section 1 is adaptive, but non-minimal in the sense that some derouting is permitted. In Section 2, another deadlock-free adaptive worm-hole based routing algorithm for the hypercube interconnection is presented. This algorithm, unlike the one shown in Section 1, is minimal. Finally, in Section 3, some well-known worm-hole algorithms for the hypercube were evaluated together with the new ones on a hypercube of* $2^{10}$ *nodes. One oblivious algorithm, the* Dimension-Order, *or* E-Cube *routing algorithm[1] was tried. In addition, three partially adaptive algorithms were considered: the* Hanging *algorithm [2, 3], the* Zenith *algorithm [3], and the* Hanging-Order *algorithm [4]. Finally, a fully adaptive minimal algorithm presented independently in [5] and [6] was tried. This algorithm allows each message to choose adaptively among* all *the shortest paths from its source to its destination. Only four virtual channels per physical link are needed to achieve this. This technique will be referred to as* Fully. *The results obtained show that the two new algorithms are good candidates as a choice for worm-hole routing in the hypercube network.*

## 1 Adaptive and non-minimal routing algorithm.

In this Section, an adaptive, non-minimal, deadlock- and livelock- free routing algorithm for the hypercube will be presented. This algorithm will be referred to as *Non-Minimal*. Some limited derouting is allowed at each step of the routing algorithm for every message. This non-minimality, though limited, is aimed both at providing each message with a wealth of paths for arriving at its destination, and at breaking structured communication patterns. One important feature of this algorithm is that each virtual channel needs to reach only a constant number of other virtual channels, independent of the size of the network [7]. This is relevant in the hypercube because of the logarithmic growth of physical links per routing node, and should be contrasted with the algorithms in [1] for the hypercube.

Given a pair of nodes $s$ and $d$, any route built by this technique for a message going from $s$ to $d$ has length between $n$ and $2n$ in an $n$-dimensional hypercube. The route of a message will consist of $n$ phases. Each phase processes one dimension. At the start of phase $i$, dimensions $n-1$ through $i+1$ have already been corrected, and will never be modified again. When dimension $i$ is being processed, a message $m$ will either have to correct it or not. Regardless of this need, $m$ will be sent first through a dimension $j < i$ (if $i$ is not one of the least significant dimensions), no matter whether this means *correcting* dimension $j$ or not. Only after this will the message be sent through dimension $i$, if required. $j$ will be chosen within a set of $d$ dimensions, $dims(p, i)$, where $p$ is the node $m$ is in. In principle, the set of $d$ dimensions corresponding to a certain phase of the algorithm may be different for each node of the network, but it must be fixed in advance. The formal description of the algorithm together with correctness and freedom from deadlock proofs can be found in [7].

Now, the main problem to solve is how to select for each node and for each phase the dimensions associated so as to have something at least *similar* to the *expansion property* [8, 9], while keeping the number of virtual channels associated with each physical link independent of the size of the network.

Let $d = 3$. $dims(p, i)$ has to be defined for each node $p$ and for each dimension $i$. For every node $p$, $dims(p, i) = \{i-2, i-4, i-6\}$ *if* $i \geq 6$, $dims(p, 5) = \{3, 1\}$, $dims(p, 4) = \{2, 0\}$, $dims(p, i) = \emptyset$ *if* $0 \leq i \leq 3$. So, the algorithm that results from the definitions above needs at most 4 virtual channels per directed physical link. Each of these channels has to be connected to at most 4 other channels. There are $3^{n-6}2^2$ different paths between any pair of nodes. These paths are not necessarily virtual-channel-disjoint.

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

†Computer Science Dept., IBM Almaden Research Center, 650 Harry Road, San José, California 95120-6099.

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

[1] In fact, it should be noticed that, even though the presentation of the routing techniques in this paper is carried out in terms of worm-hole routing, all of them can be easily adapted for the packet-switching routing model.



0-8186-2672-0/92 $03.00 © 1992 IEEE

## 2 Adaptive and minimal routing algorithm.

In this Section, an adaptive, minimal routing algorithm for the hypercube will be presented. This technique will be referred to as *Subcubes*. Although this algorithm has a considerable amount of adaptivity, it requires only very moderate resources for its implementation, namely only 2 virtual channels per physical link.

This algorithm is based mainly upon considering the hypercube as a hierarchical network in which each node is a small hypercube. These small hypercubes will be referred to as *subcubes*[2]. Each subcube will consist of $2^k$ nodes, for some $0 \leq k < n$. Then, $k$ dimensions, $i_1, \ldots, i_k$, will be chosen such that they will identify the position of each node within the subcube it belongs to. Each subcube will be determined by the value in the remaining $n - k$ dimensions.

The routing functions over this network will be defined as if the hypercube were *hung* from one of these subcubes [2, 3]. The path of a message going from one subcube to another subcube will consist of two hierarchical phases. During the first hierarchical phase, the message will visit the nodes of the hierarchical network by moving downwards, considering the network as hung from a fixed subcube. During the second hierarchical phase, the message will visit the nodes by travelling upwards. When $k = 0$ the two phase algorithm used above is known to generate severe congestion at node $1 \ldots 1$ (if the network is hung from node $0 \ldots 0$) for moderate network sizes. If $k$ is different from 0, the congestion phenomenon is distributed to all the nodes of one subcube. Since $2^k$ nodes are present in this subcube, congestion is less likely to arise at a single node.

Several routing algorithms can be defined on top of this routing strategy for routing between any pair of nodes of the hypercube. These routing algorithms will differ in the way in which messages will move within each subcube. As an example of such an algorithm, each message will be allowed to move *within* the subcubes only during its first phase. By taking any of the hierarchical steps of the first phase, the message passes from one subcube to another, and different routing strategies can be followed within each subcube, as pointed out above. The message will have to have corrected all the dimensions $i_1, \ldots i_k$ before starting the second hierarchical phase. The routing strategy in each subcube can be chosen independently. Two possible choices are: routing by correcting dimensions in order [1], or routing as if each subcube were hung from one node [2, 3]. Either of these routing strategies requires two virtual channels per bidirectional link for its deadlock-free definition. The complete description of the algorithm has been presented in [7].

[2]The idea of defining a hierarchical topology using the hypercube has also been used in [10].

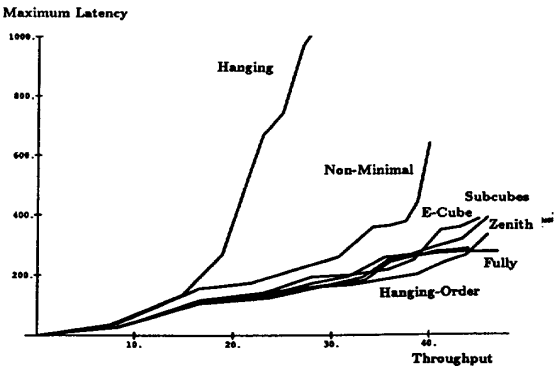


Figure 1: Results for *Random Routing* with worms of length 10.

## 3 Simulations.

In this Section, a variety of algorithms for wormhole routing on the hypercube are evaluated together with the new ones. A simple node model was designed and adapted for all the algorithms. This allows to compare the different strategies fairly from an algorithmic point of view. Several simulations for continuous routing were tried on a hypercube of $2^{10}$ nodes. The traffic patterns analyzed include *random* routing, in which each node selects a destination randomly whenever it injects a message, *complement*, and *transpose*, in which each node keeps injecting messages to its complement node, or to its transpose node, respectively. The routing node model, the network activity and the injection model of the simulation have been discussed in [11].

Figures 1 to 6 show the maximum latency of the messages as a function of throughput, for the different traffic patterns described above. Figures 1 to 3 correspond to messages 10 flits long, whereas Figures 4 to 6 are for messages 20 flits long.

Throughput is described as a percentage of $\lambda_{max} = 1/(2b)$, where b is the worm length (10 or 20 flits, in this case), as this is a theoretical upper bound for the maximum throughput achievable.

*Non-Minimal* performed well for every pattern of communication, as was expected for its ability to break structured communication patterns. However, its performance is worse than the best algorithm for each communication pattern as it is a non-minimal

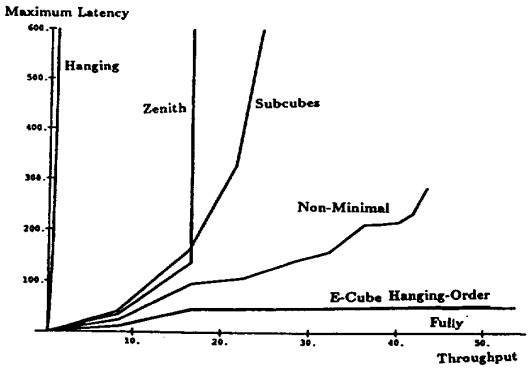


Figure 2: Results for *Complement* with worms of length 10.

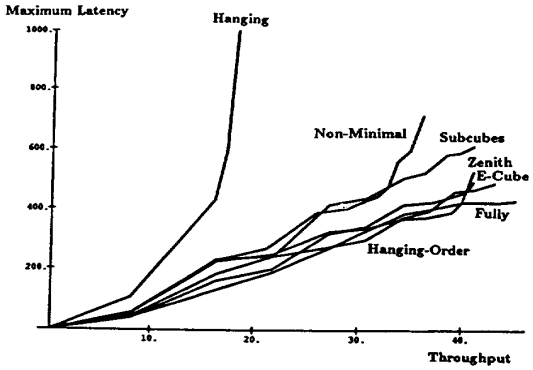


Figure 4: Results for *Random Routing* with worms of length 20.

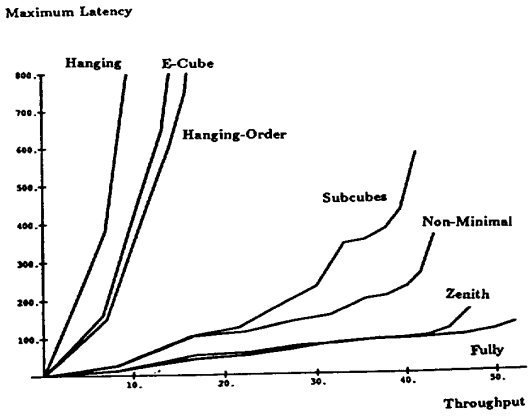


Figure 3: Results for *Transpose* with worms of length 10.

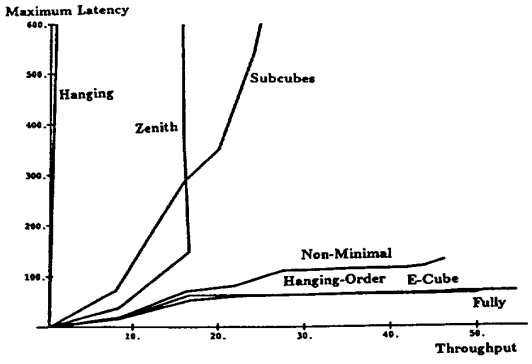


Figure 5: Results for *Complement* with worms of length 20.

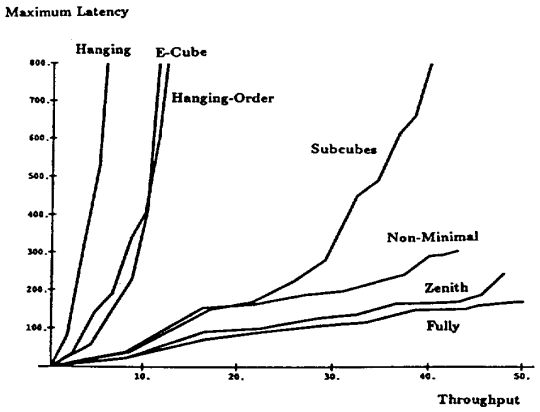


Figure 6: Results for *Transpose* with worms of length 20.

algorithm. Although *Non-Minimal* requires eight virtual channels per bidirectional link, the moderate connectivity between virtual channels required by this algorithm may result in a practical node design [7]. *Subcubes* seems to be a good candidate for a node design with a small number of virtual channels. Even for nodes with equal storage capacity, this algorithm leads to a simpler node model [11]. *Subcubes* was the only one among the algorithms requiring only two virtual channels per bidirectional link that could sustain a throughput of at least 20% of $\lambda_{max}$ in all the communication patterns that were tried.

## References


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

[2] Y. Birk, P. Gibbons, D. Soroker, and J. Sanz, "A simple mechanism for efficient barrier synchronization in MIMD machines," RJ 7078 (67141) Computer Science, IBM Almaden Research Center, October 1989.

[3] S. Konstantinidou, "Adaptive, minimal routing in hypercubes," in *6th. MIT Conference on Advanced Research in VLSI*, pp. 139–153, 1990.

[4] G.-M. Chiu, S. Chalasani, and C. S. Raghavendra, "Flexible, fault-tolerant routing criteria for circuit-switched hypercubes," in $11^{th}$ *International Conference on Distributed Computing Systems*, IEEE, 1991.

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

[6] 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.

[7] L. Gravano, G. Pifarré, and J. Sanz, "Adaptive Worm-hole Routing in Tori and Hypercubes," TR:91-10, IBM Argentina - CRAAG, March 1991.

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

[9] T. Leighton and B. Maggs, "Expanders might be practical: Fast algorithms for routing around faults on multibutterflies," in $30^{th}$ *Annual Symposium on Foundations of Computer Science* (IEEE, ed.), pp. 384–389, October 1989.

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

[11] G. Denicolay, L. Gravano, G. D. Pifarré, and J. Sanz, "Experimental comparison of worm-hole routing algorithms in hypercubes." In preparation, 1992.