Routing Strategies and Message Contention on Low-dimensional Interconnection Networks *

Didier Badouel, Charles A. Wüthrich, and Eugene L. Fiume
Computer System Research Institute
University of Toronto

December 30, 1991

Abstract

Non-uniform message distribution in interconnection networks is a common problem and causes network congestion. This paper proves that a careful choice of the routing algorithm can increase significantly the communication performance under pathological communication patterns, such as hot spots. A comparison between widely used e-cube routing and a new algorithm based on Computer Graphics line drawing techniques has been made for k-ary n-cube networks. The algorithm proposed uniformly distributes the direction in the path followed by the messages, thus loading in a more uniform way the links in the direction of the hot spot. The problem is analyzed both theoretically, and through simulations.

Index Terms Routing algorithms, interconnection networks, hot spots contention, highly parallel systems.

1 Introduction

Distributed memory parallel computers have gained wide acceptance as a cost-effective approach to large-scale computers. One of their major advantages lies in the fact that the architecture is scalable: as more processing elements are added to the system, its computing power, memory, and communication bandwidth increase proportionally. On the other hand, the cooperation among distributed processing elements generates communication among them and thus loads the interconnection network. When the saturation limit of the network is reached, the global computing power of the machine is severely affected. The problem of avoiding network saturation therefore plays an important role in the field of parallel computing.

*This work has been supported by the *Information Technology Research Centre* of Ontario, by the French *Institut National de Recherche en Informatique et Automatique*, and by the Swiss *Fondo Nazionale Svizzero per la Ricerca.*

In the use of parallel computers, basically two modes of computing can be identified, as pointed out by several authors [10, 9]. In *particle* mode, data transfers only occur between neighbours in the processor network. The system therefore does not require a routing mechanism. Typical examples of this computation mode are pipeline and systolic computations. In *continuum* mode, data transfers also occurs between non-neighbouring nodes, thus requiring a routing mechanism. In this case, the interconnection network becomes a crucial part of the architecture.

In particle mode, program design must take into account the topology of the architecture. The continuum mode instead is more general and more independent of the architecture; however, in order to obtain good performance, the user has still to comply with architectural constraints. If the application is not tailored to the system topology, message traffic conflicts can saturate the network and severely degrade the performance of the system. However, since distributed memory parallel computers are expected to be used for a wide range of applications, the user too must be freed as much as possible from architectural constraints. The network architecture therefore must be designed so that it can handle a wide range of communication patterns, and minimizes network saturation. The network routing strategy is a key component of the network architecture.

In this paper, we study routing strategies and present a new routing algorithm, derived from a basic algorithm in Computer Graphics, that uniformly distributes the directions in the path followed by a message in the network. This property allows a reduction of the network saturation when the message distribution is not uniform, allowing a higher throughput under critical conditions to be sustained. The new algorithm is compared with the widely used e-cube algorithm, and the improvements of the communication performance in the worst communication case, the *all-to-one* communication pattern, is presented theoretically. In order to confirm the results obtained, simulations of k-ary n-cube interconnection networks have been developed and are discussed.

2 Routing algorithms for k-ary n-cube topologies

Routing algorithms govern the choice of the path a message follows as it travels through a network. Multiprocessor interconnection network routing strategies have not changed much, despite recent changes in multiprocessor network topology, and the abandoning of binary n-cubes (also called *binary hypercubes*) in favour of k-ary n-cubes of lower dimensions (e.g. *meshes* and *tori*). To date, the routing strategy typically used in mesh topologies is directly derived from the strategy used in binary hypercube topologies. This is, we believe, primarily a historical artifact; we will show that alternative routing strategies can lead to increased performance under certain conditions.

Before considering alternative routing strategies, we introduce our notation. We will use the model based on the k-ary n-cube, since, as Dally states [3], many concurrent computers in use employ either such a network topology or one that is isomorphic to it. The constant n represents the *dimension* of the network, and k is called *radix* of the cube, i.e. the width of the network in each dimension. The total number N of the nodes of the network is given by $N = k^n$.

Any node in a k-ary n-cube can be uniquely identified by the n-uple $(a_0, a_1, \ldots, a_{n-1})$, where each a_i represents the position of the node in the i-th dimension ($0 \leq a_i \leq k-1$). There are two different ways of arranging connections among the nodes of the network. If node $(a_0, a_1, \ldots, a_i, \ldots, a_{n-1})$ is connected to nodes $(a_0, a_1, \ldots, a_i \oplus 1, \ldots, a_{n-1})$, where $\oplus$ indicates the sum modulo k and the coordinates are numbers modulo k, we say that the network is a *torus* (because of the "wrap around" over the border of the cube). The network is said to be a *mesh* if each node of the network $(a_0, a_1, \ldots, a_i, \ldots, a_{n-1})$ is linked with its upper and lower neighbours $(a_0, a_1, \ldots, a_i + 1, \ldots, a_{n-1})$ and $(a_0, a_1, \ldots, a_i - 1, \ldots, a_{n-1})$ in the i-th dimension. In this paper we only consider toroidal topologies, since the equivalent routing strategies can easily be generated for mesh topologies. From now on the operations on coordinates are assumed to be modulo k.

A message is a sequence of bytes which have to travel in the network from the source node to a destination node. The *path* followed by the message is the sequence of nodes followed by the message while travelling through the network. A *routing algorithm* generates the path that the messages have to follow. A path can be also seen as a sequence of one-step *hops* taken by the message in its journey through the network. The directions correspond to the n dimensions of the network, since each node $(a_0, a_1, \ldots, a_i, \ldots, a_{n-1})$ of the network is connected to only one other node in the same dimension, namely $(a_0, a_1, \ldots, a_i+1, \ldots, a_{n-1})$ (for all $a_i = 0, \ldots, k$). If the message chooses the i-th dimension at the step considered, it is said to take the i-th direction. While following its routing path, at each node a message takes therefore one direction according to the routing algorithm used.

In this paper we only consider routing algorithms that choose a shortest path between source and destination nodes. Moreover, we only consider deterministic algorithms; that is, algorithms that always generate the same path for each pair of source and destination nodes. Algorithms that adapt the path according to the state of the network, such as the one presented in [5], are excluded from our discussion, since they represent a variation of existing deterministic algorithms[1].

Traditionally, the routing algorithm used in k-ary n-cubes is the so-called e-cube algorithm [4], which avoids deadlocks among the messages in the network. The dimensions, and thus the directions, of the network are ordered in decreasing order of their related index. If a message has to travel from node $P = (p_0, p_1, \ldots, p_{n-1})$ to node $Q = (q_0, q_1, \ldots, q_{n-1})$, then $q_i - p_i$ is the number of times that the i-th direction has to be chosen ($i = 0, 1, \ldots, n$). The e-cube algorithm routes the message first $q_{n-1} - q_{n-1}$ times in the direction $n-1$, so that node $(p_0, p_1, \ldots, q_{n-1})$ is reached, then routes it $q_{n-2} - p_{n-2}$ times in the direction $n-2$ and so on until the node Q is reached. Since this algorithm routes in decreasing order of directions and the resulting path is made by subpaths parallel to the various directions, we shall call this approach *cartesian routing*. An example of the path followed by a message using cartesian routing in a two-dimensional mesh can be seen in Figure 1a.

Cartesian routing coalesces all one-step hops along one dimension. As we will show in Section 3, this routing strategy does not always perform as well as a routing strategy where the directions are uniformly distributed across the path. In the following, we describe a new routing algorithm, which we call *graphical routing*, derived from a line drawing algorithm in Computer Graphics [2]. Basically, graphical routing computes a discretization of a straight

[1]In fact, the new algorithm proposed in this paper could be used as a base for a non-deterministic routing strategy.

a) Cartesian routing

b) Graphical routing

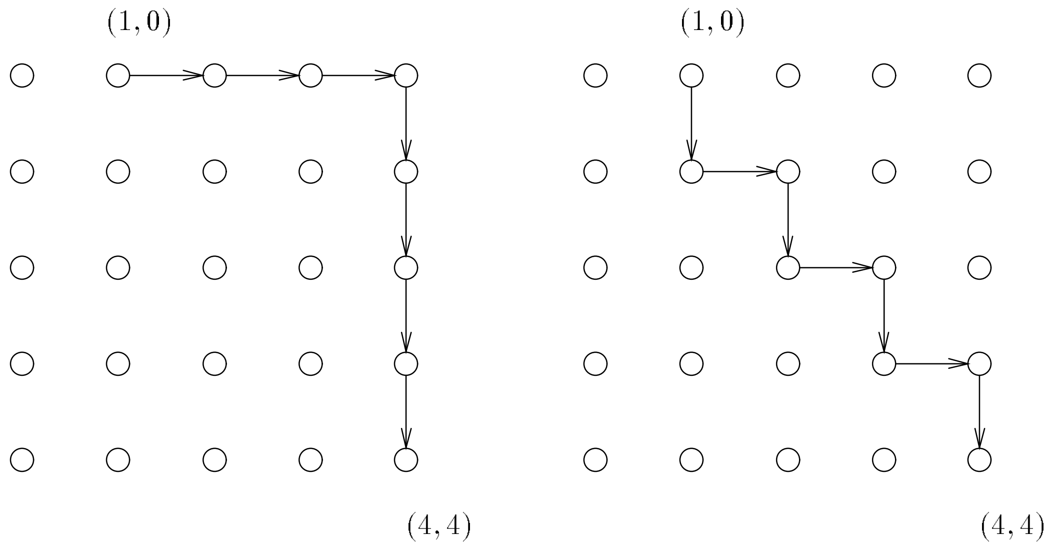

Figure 1: Two deterministic strategies for message routing.

line joining the source and the destination nodes. Before describing the algorithm in detail, we first introduce more formal notation.

One can consider the nodes of a k-ary n-cube as a finite subset of an n-dimensional lattice or *hyperlattice* containing all points $P = (p_0, p_1, \ldots, p_{n-1})$ of $\mathbb{R}^n$ such that their coordinates p_i are all positive integers smaller than k. One can also consider the links between nodes to define a neighbourhood relation between the Voronoi *hypervolume* associated with each point of the hyperlattice. In fact, in a k-ary n-cube, only voxels having a *hyperface* in common, i.e. voxels corresponding to hyperlattice points having $n-1$ coordinates in common, are considered neighbours. In a two-dimensional lattice, such a neighbourhood relation gives rise to edge-connection or 4-connection, while in the three-dimensional space it leads to face or 6-connection. The neighbourhood relations among the hyperlattice points introduce a rasterization procedure for curves into the hyperlattice: a rasterization of a curve is in fact a path of neighbouring lattice points. For the two-dimensional case it has been proven both for the 8-connected and for the 4-connected case [8, 11], that the directions of the rasterization of a straight line are uniformly distributed.

So far, no definition has been given for uniform distribution among directions in the n-dimensional space. We shall give an intuitive geometric definition here. Consider two hyperlattice points $P = (p_0, p_1, \ldots, p_{n-1})$ and $Q = (q_0, q_1, \ldots, q_{n-1})$. Let $n_i = q_i - q_i$. A shortest path between P and Q requires $m = \sum n_i$ steps assuming hyperface connectivity. The hyperline points are the points with coordinates[2] $x_i = (q_i - p_i)t + p_i$, with $t \in [0, 1]$. The parameter t introduces an ordering on the points of the straight line. Consider the straight line points P_{i,h_i} obtained for $t = \frac{h_i}{n_i}$, where $h_i = 1, \ldots, n_i$, and order them in increasing order of their corresponding parameter value. Whenever an ambiguity occurs, and two parameter

[2]We recall that operations on coordinates are defined modulo k.

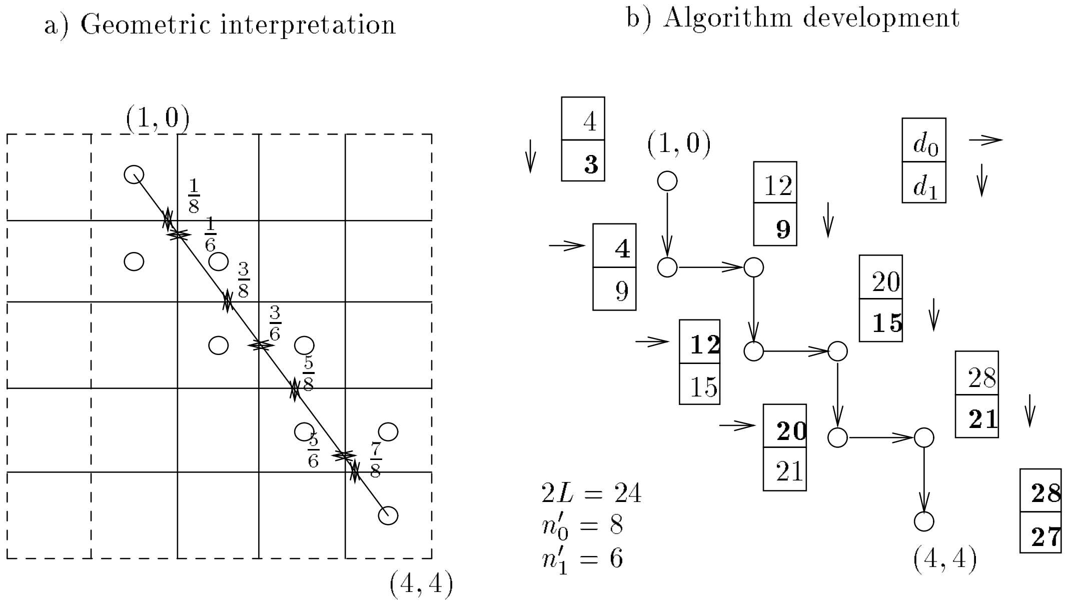

Figure 2: Graphical routing in a two-dimensional network.

values $\frac{i}{n_i}$ and $\frac{j}{n_j}$ coincide, an arbitrary order may be chosen[3]. In other words, the segment PQ is subdivided into n_i parts for each dimension i, and the points obtained on the straight line segment are ordered by increasing values of the parameter t. When two subdivision points coincide, the one corresponding to the smaller dimension is considered to precede the other one.

As shown by the two-dimensional example of Figure 2a, the segment is divided into $n_0 = 3$ parts in the 0-th (or horizontal) dimension and into $n_1 = 4$ parts in the 1-st (or vertical) dimension. The points are ordered from $(1,0)$ to $(4,4)$.

The resulting set is a finite ordered set of segment points P_{i,h_i}, which can be renamed as $A_0, A_1, \ldots, A_{m-1}$. Consider the finite path built by taking the sequence of directions $\{a_k\}_{k=0,\ldots,m-1}$, such that each direction a_k corresponds to point $A_k = P_{a_k,l}$ for some l. The directions in such a path are *uniformly distributed* in the sense that hops in the same direction are as evenly spaced as possible in the chain. Moreover, if we follow the path from point P, point Q will be reached[4].

A simplified version of the routing algorithm is obtained as follows. For each direction i, an integer counter d_i is reserved at the header of the message. In order to generate the path between the two nodes P and Q with uniformly distributed directions, it is necessary to compute the values of n_i. Their least common multiple $L = LCM(\{n_i\})$ is then computed[5] to obtain the values of $n''_i = \frac{L}{n_i}$. To obtain only integer computations, the values of $n'_i = 2n''_i$ are computed. The counters d_i are initialized to n''_i. At each step, n'_i is added to the d_i with the smallest value, and the i-th direction is taken by the message. The message is passed

[3] Although here we will always choose the one corresponding to the smaller dimension first.

[4] In fact, for mesh topologies, the path from Q to P coincides with the reserval of the path from P to Q.

[5] In fact, either a low complexity method in $O(n \log k)$ based on a table lookup or a simple common multiple can be used here.

to the next node (along direction i), where the procedure is repeated. The path of the message ends when all d_i have reached the value of $2L + n''_i$, which in this case is equivalent to $\forall i, d_i \geq 2L$.

Note that the initialization of the counters d_i is necessary, since if d_i were not initialized, then the path generated would correspond to another rasterization scheme, and would mostly end with the same direction.

Let us follow the simple two-dimensional example shown in Figure 2b, where the path from $(1,0)$ to $(4,4)$ has to be generated. Therefore, $n_0 = 3$, $n_1 = 4$, and $L = 12$. The counters d_0 and d_1 are initialized to $n''_0 = 4$ and $n''_1 = 3$, respectively, and the increments for the two components are $n'_0 = 8$ and $n'_1 = 6$. The smallest d_i value, i.e. $d_1 = 3$, indicates the first direction chosen, the vertical direction: d_1 is therefore incremented by $n'_1 = 6$, and the counters are now updated for the next step. This process is repeated as long as both d_0 and d_1 are smaller than $2L = 24$. Note that in the case of a torus only one half of the links need to be taken in consideration since the links are unidirectional.

If we restrict ourselves to the two-dimensional meshes, the algorithm introduced above coincides with any classical straight line drawing algorithm, such as the 4-connected version of Bresenham's algorithm [2, 11]. This algorithm uses the same technique used for computing the interections of a ray with the voxels of the 3-dimensional space except that we extend it to n dimensions.

Assuming the destination nodes are equiprobable, uniform distribution in the path directions implies that for each incoming message in a node of the network there is an equal probability of utilizing each outgoing link. For cartesian routing, on the other hand, the probability that the message has to be routed in the higher dimensions is very high at the beginning of its path sequence, whereas the probability for a message to be routed in the lower dimensions is much higher at the end of the path sequence. Although the overall distribution of the messages over all the channels in all dimensions is uniform with both routing techniques, locally the distribution is not uniform for the cartesian routing. As will show in Section 3, this causes problems when the network becomes overloaded. In particular, such a situation happens when many nodes want to communicate with the same node, i.e. in presence of *hot spots*. While one would expect that under normal network conditions the cartesian and the graphical algorithms would perform similarly, the use of the graphical algorithm should considerably improve the performance of the network when hot spots are present. A more detailed analysis of these concepts will to be presented in the following sections.

3 Analysis of network contention

When network traffic is non-uniform, regions of the network become overloaded, and the overall performance of the network degrades. Such *hot spot* situations have been previously analyzed in [7] for large parallel shared memory architectures based on a multistage switching network. They also occur in distributed memory parallel systems.

The performance of an interconnection network is usually expressed in terms of *throughput* per node, which for the entire network becomes its *bandwidth*, and *transmission delay* per message. The presence of a *hot spot* affects both measures. The saturation of the links around

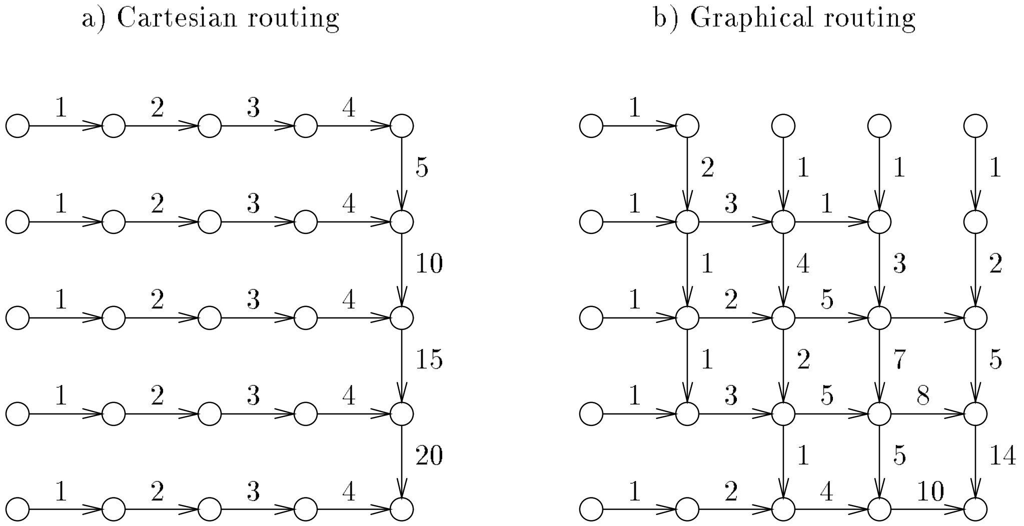

Figure 3: Links load comparison for an *all-to-one* communication situation. Each link is labeled by the number of messages using this path.

the hot spot location gradually propagates from the most used link, i.e. the *weakest link*, to the entire network. This *tree saturation* degrades communication in the entire network. The weakest link, which is the root of the tree saturation, is the main obstacle for sustaining high throughputs when message paths converge. Figure 3 illustrates the load of the links on a bi-dimensional network for a typical hot spot case, namely the *all-to-one* communication situation.

For all-to-one communication, the saturation of the weakest link is minimized by a routing algorithm that equally distributes the messages among the input links of the destination node. Such a load will be referred to as *ideal*. Figure 4a shows the ideal load of the weakest link on a k-ary n-cube topology. The ideal load represents a limit for the reduction of the saturation of the weakest link and is given by the following relation:

$$C_i = \left\lceil \frac{k^n - 1}{n} \right\rceil . \tag{1}$$

The load of the weakest link is plotted for several typical topologies, such as 2D-tori ($n = 2$), 3D-tori ($n = 3$), and binary hypercubes ($k = 2$). Figures 4b and 4c illustrate the load of the weakest link C_c and C_g for cartesian and graphical routing strategies respectively. The following propositions give the expressions for C_c and C_g.

Proposition 3.1 *The weakest link load for an all-to-one communication on a k-ary n-cube when cartesian routing is used is*

$$C_c = k^n - k^{n-1} \tag{2}$$

Proof. Let $Q = (q_0, \ldots, q_{n-1})$ be the destination node. The weakest link is the link joining $(q_0 - 1, \ldots, q_{n-1})$ and $(q_0, \ldots, q_{n-1})$, i.e. the link in the 0-th dimension, which is the last used by the cartesian algorithm. All messages routed to Q use the weakest link, except

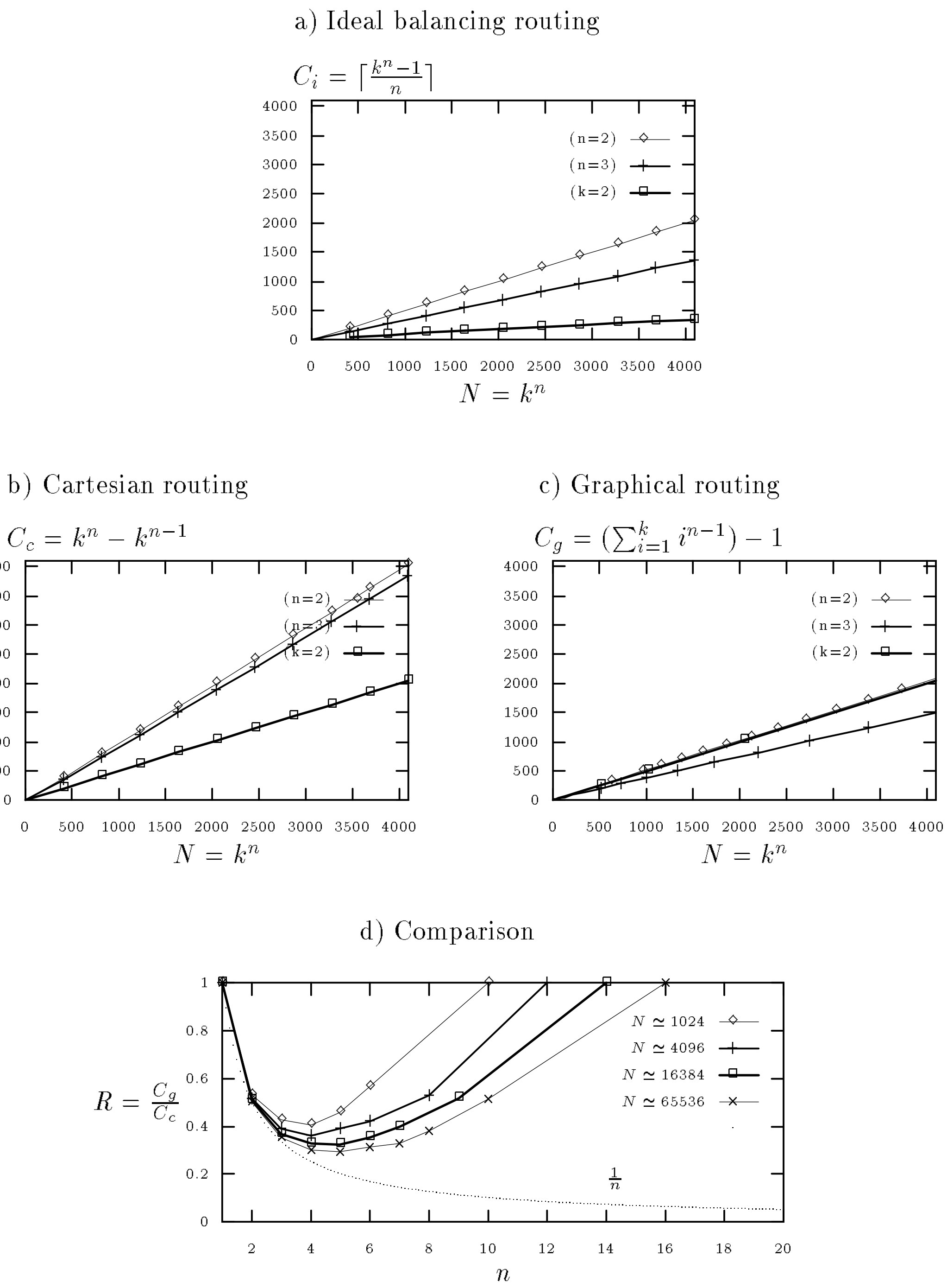

Figure 4: Weakest link saturation comparison in k-ary n-cube interconnection networks with respect to the routing strategy.

the ones coming from nodes belonging to the hyperplane $x_0 = q_0$. Those messages are issued from the nodes belonging to the set

$$E = \{(p_0, \cdots, p_{n-1}) \, : \, p_0 \neq q_0\} \, .$$

The load of the weakest link is thus $Card(E) = k^n - k^{n-1}$. □

Proposition 3.2 *The weakest link load for an all-to-one communication on a k-ary n-cube when graphical routing is used is*

$$C_g = \left(\sum_{i=1}^{k} i^{n-1} \right) - 1 \tag{3}$$

Proof. At the last step, the graphical algorithm routes a message to its destination node through the link corresponding to a dimension d if the number of steps required in this dimension is greater than the number of steps required in all other dimensions. When such a dimension is not unique, the default dimension is the one having the smallest index. The weakest link is thus the link joining $(q_0, \ldots, q_{n-1} - 1)$ and $(q_0, \ldots, q_{n-1})$, i.e. the link in the $(n-1)$-th dimension. Messages routed from a node P to the destination node Q use the weakest link if and only if P belongs to the set

$$F = \{(p_0, \cdots, p_{n-1}) \, : \, |p_{n-1} - q_{n-1}| \geq |p_j - q_j| \, , \, 0 \leq j \leq n-1\} - \{Q\} \, .$$

The load of the weakest link is thus $Card(F) = (\sum_{i=1}^{k} i^{n-1}) - 1$. □

$\sum_{i=1}^{k} i^{n-1}$ is a function of k which can be conveniently generated using the following recursive relation ([1], page 59):
If

$$\sum_{i=1}^{k} i^{n-1} = a_1 k^n + a_2 k^{n-1} + \cdots + a_{n+1} \tag{4}$$

then

$$\sum_{i=1}^{k} i^n = \frac{n}{n+1} a_1 k^{n+1} + \frac{n}{n} a_2 k^n + \cdots + \frac{n}{1} a_{(n+1)} k + [1 - n \sum_{i=1}^{n} \frac{a_i}{(n+2-i)}] \, .$$

Function $R(k,n) = \frac{C_g(k,n)}{C_c(k,n)}$ in Figure 4d shows the improvement that can be expected by using the graphical algorithm. Note that both the cartesian and the graphical routing algorithms are equivalent for the boundary cases on the k-ary n-cube model ($R(k,1) = R(2,n) = 1$). In the unidimensional case, i.e. for rings, this happens trivially because there is only one direction in the path. For binary hypercubes ($k = 2$), at most one step has to be made in each dimension. Therefore, in the graphical algorithm, all communications paths which use at least two directions fall into an ambiguous situation and choose the lower dimensions first, and, consequently paths generated by the cartesian and by the graphical algorithms coincide. Therefore $R(n,k)$ is only interesting in cases lying between these two boundary cases. Figure 4d shows that for small values of the dimension n, R behaves like $\frac{1}{n}$. When n increases, R diverges to reach the value of 1, the value obtained for the binary hypercube configuration. If the number of nodes is increased, the function R displaces its minimum along the curve $\frac{1}{n}$ to increasing values of n. This behaviour can be confirmed by computing the limit $R(n,k)$

N	256	$1K$	$512K$	$1M$	$512M$	$1G$
n	3	4	5	6	7	8

$K = 1024$; $M = K \times K$; $G = K \times M$.

Table 1: *Optimal* dimension for the function $R = \frac{C_g}{C_c}$.

for $N = k^n \to \infty$. Note that in the limit operation, the interesting cases are only when k varies, since $n \to \infty$ generates binary hypercube configurations, where $R(2, n) = 1$. From Equations 2, 3 and 4, and the definition of a_1, the following result can be deduced:

$$\lim_{k\to\infty} R(n,k) = \lim_{k\to\infty} \frac{a_1 + \frac{a_2}{k} + \cdots + \frac{a_{n+1}}{k^n}}{1 - \frac{1}{k}} = a_1 = \frac{1}{n}\,.$$

The slow shift of R along the curve $\frac{1}{n}$ indicates that the greatest improvements in the performance of the graphical algorithm over that of the cartesian algorithm are obtained in low-dimensional interconnection networks. The notion of low-dimension depends on, and should be therefore weighted by, the number of nodes in the network. Table 1 shows the dimensions optimizing the function R for different network sizes.

In networks having a low number of dimensions, therefore, the graphical algorithm can reduce the contention of the weakest link in the all-to-one situation by a factor of n. In the next section we study its influence on the global performance of the network, i.e. on its throughput and on the message transmission delays.

4 Simulation of network contention

To simulate network contention, Pfister and Norton [7] altered the uniform distribution of the messages, which represents the ideal model for the traffic in a network, to include a single *hot spot*, i.e. a single node to which a specified fraction of the total messages is directed. Although they developed this method for perturbating distribution of memory accesses to model global locks in large shared memory systems, it is also useful to study hot spot behaviour when simulating message passing. It is clear that contention in the network can be more complex in a distributed memory parallel system. For example, *hot regions* can occur, i.e. a subset of neighbouring nodes to which messages require access at a given time. But both hot regions and hot spots cause similar effects in the network and affect performance in the same way. This is due to the fact that *tree saturation* originating from the weakest link of the hot spot or hot region occur in both cases. For this intrinsic similarity the contention situation modelled by a *hot spot* can in a reasonable way be used to simulate converging message communication patterns.

In our simulations, we model message production and routing in two main phases, namely the *production* phase and the *reabsorption* phase. During the *production* phase, each node attempts to produce messages at a given rate, i.e. it attempts to sustain a given throughput. During the *reabsorption* phase, no further messages are produced and the messages still

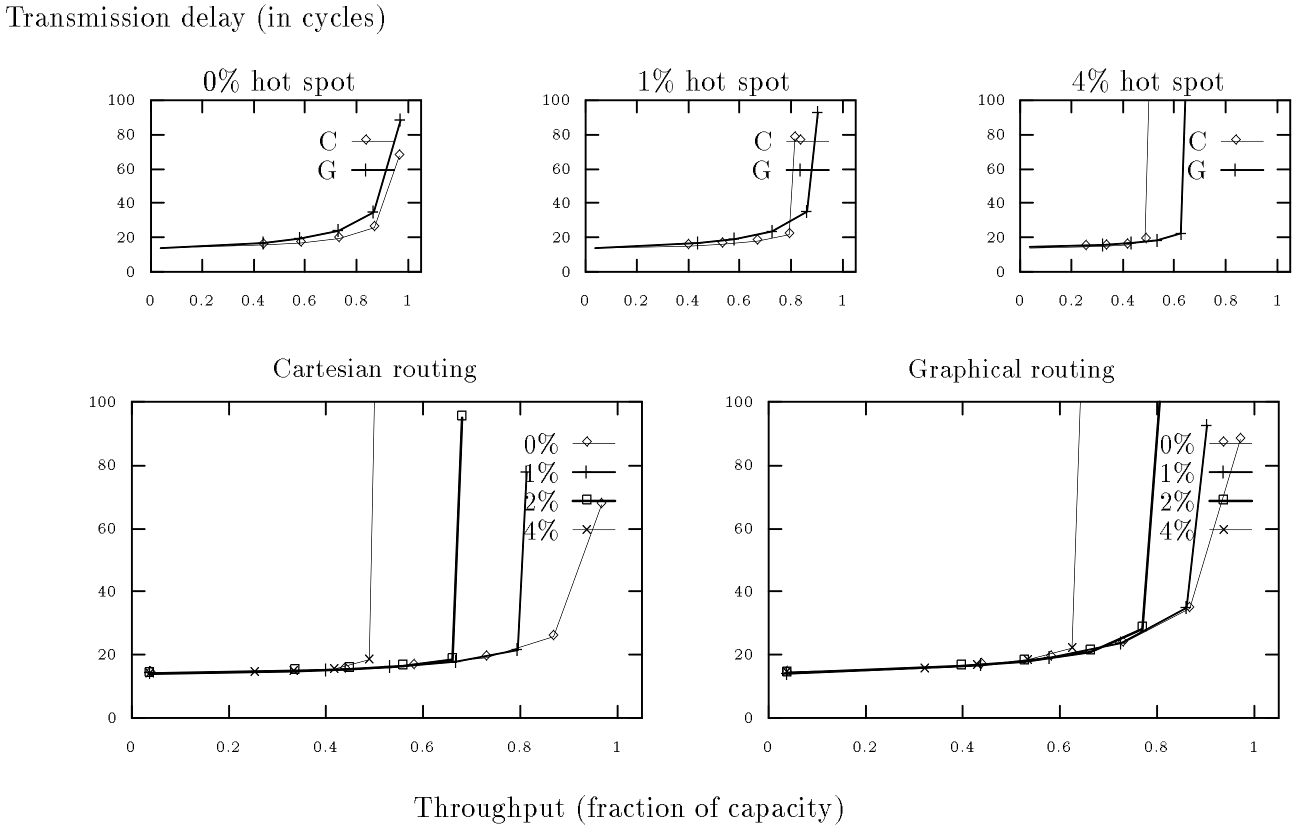

Figure 5: Communication delay versus troughput for a 15-ary 2-cube network. Each message length is $L = 1$ flit.

remaining in the network are routed until all of them have arrived at their destination. The throughput and average message delay is measured during the simulation. The model used has been kept as simple as possible since our purpose is to show the effects of network contention and to compare the two routing algorithms.

Each message is a set of flow control digits (flits). A *flit* is the smallest unit of information that a queue or channel can accept or refuse. It is also the unit of information exchanged in one communication cycle. Simulation results are given in terms of this communication cycle time. The throughput is given as a fraction of the network capacity. If one *flit* can be exchanged per node and per dimension in one cycle, then the maximum throughput per node is given by $T = \frac{n}{LD}$, where L is the average message length and D is the average path length for a message travelling the network. For k-ary n-cube networks and a uniform destination distribution for the messages, $D = (\frac{k-1}{2})n$, and thus $T = \frac{2}{L(k-1)}$.

We simulate the (*wormhole*) *virtual cut-through* switching technique [6]. When the first flit of a message arrives at an intermediate node and its outgoing channel is free, it is sent to the adjacent node in the direction of its destination before all of its flits have reached the node; if, on the other hand, the first flit of the message is blocked at an intermediate node due to a busy output channel, then the entire message is buffered. Compared to (*store*

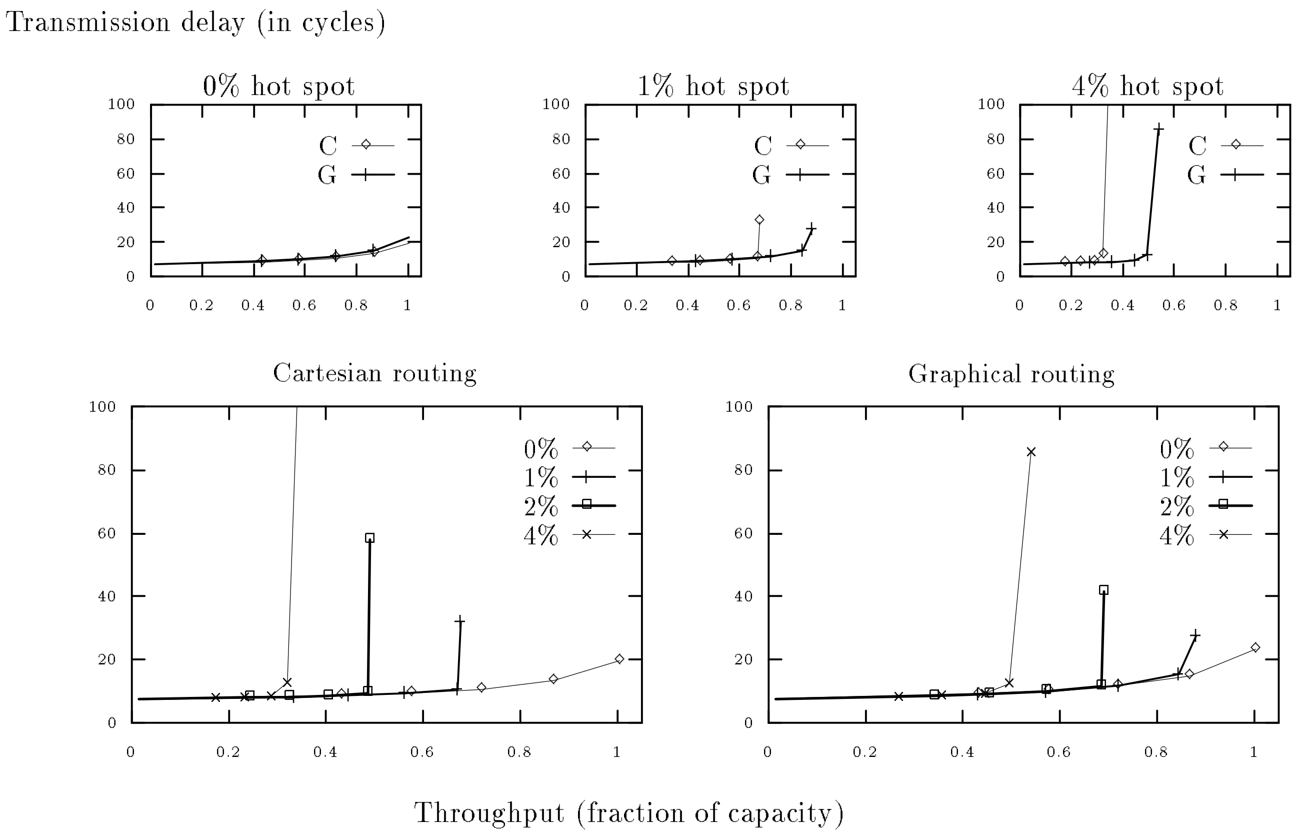

Figure 6: Communication delay versus troughput for a 6-ary 3-cube network. Each message length is $L = 1$ flit.

and forward) *message* switching, where the entire message is always received before it is forwarded, virtual cut-through switching offers much lower switching latency. Virtual cut-through differs from a pure wormhole technique [4] in that it buffers messages when they are blocked, removing them from the network. With wormhole routing, blocked messages remain in the network and require the use of virtual channels to avoid dead-locks [4].

In the simulation, each node is represented by a set of n input channels, a set of n ouput channels, and a storage device. Because only a limited amount of storage is provided at each node, blocking, and consequently retransmissions, or losses, of some messages can result. Such messages are discarded from the network. If the storage size at each node is large enough, messages are discarded only when the required throughput is too large, i.e. for unreasonable throughput values. Since the major purpose of the simulation is to measure communication performance, a more sophisticated protocol to resolve the message loss has not been considered.

5 Results

We have simulated k-ary n-cube networks of relatively small sizes so that they can be

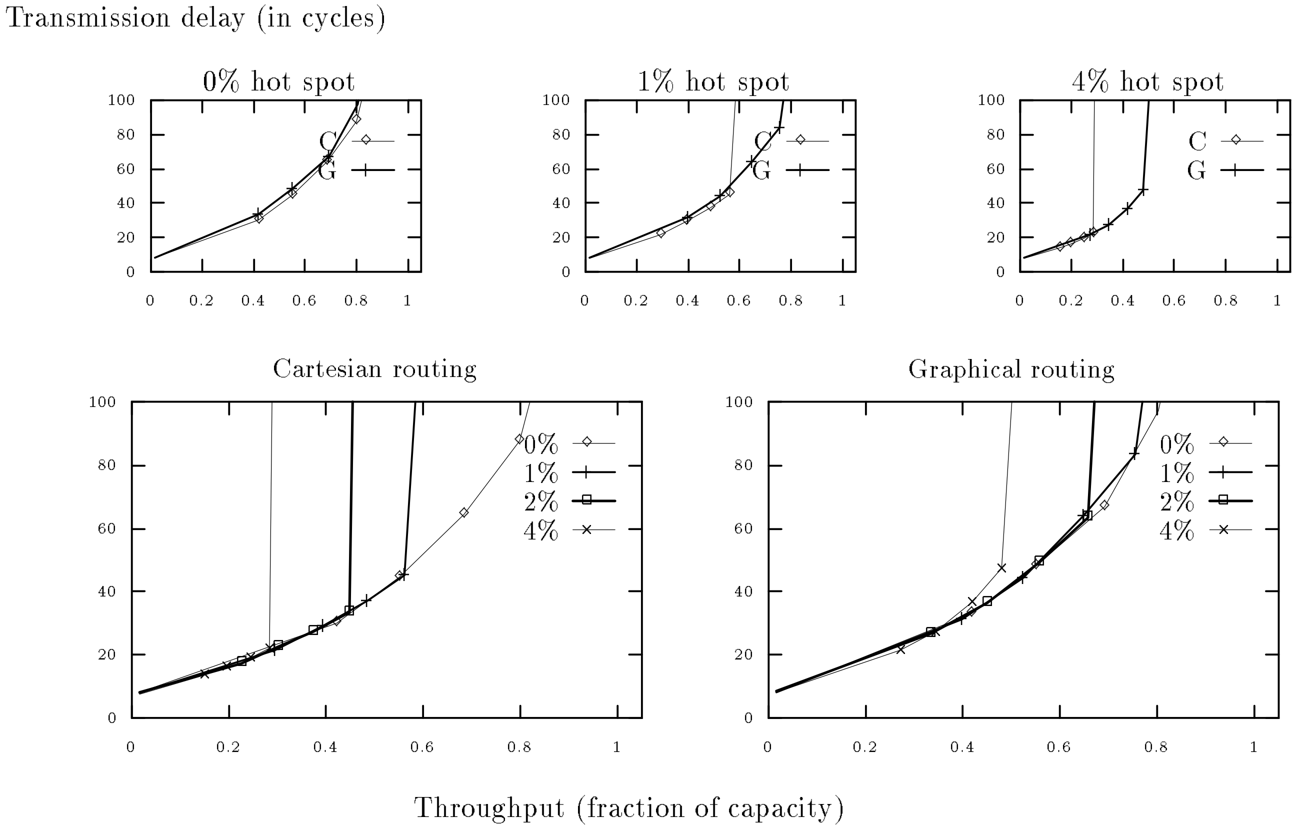

Figure 7: Communication delay versus troughput for a 6-ary 3-cube network. Each message length is $L = 10$ flits.

simulated within a reasonable time. Figures 5 to 8 show the communication performance measured for various low-dimensional network configurations. Figure 5 shows the results for a 2D-torus, Figures 6 and 7 compare the results for a 3D-torus for two different message lengths, and Figure 8 displays the results for a 4D-torus.

In each of these figures, the three top diagrams compare the cartesian and graphical routing strategies under increasing hot spot values, where the value of the hot spot indicates the proportion of messages sent to a single node. The lower diagrams show the behaviour of each strategy for increasing hot spot values. In each of these diagrams, the x axis represents the fraction of the network capacity used, and the y axis represents the average message delay.

Note that each curve must have a vertical asymptote before reaching the maximum throughput, because producing more messages than the network can handle adds only to the delay in delivering the messages; when the network is saturated, each newly produced message necessarily has to be queued. The throughput at which the asymptote occurs decreases for increasing hot spot values. Graphical routing performs similarly to cartesian routing when message distribution is uniform, but the throughput improves significantly even when only a small percentage of message convergence is introduced.

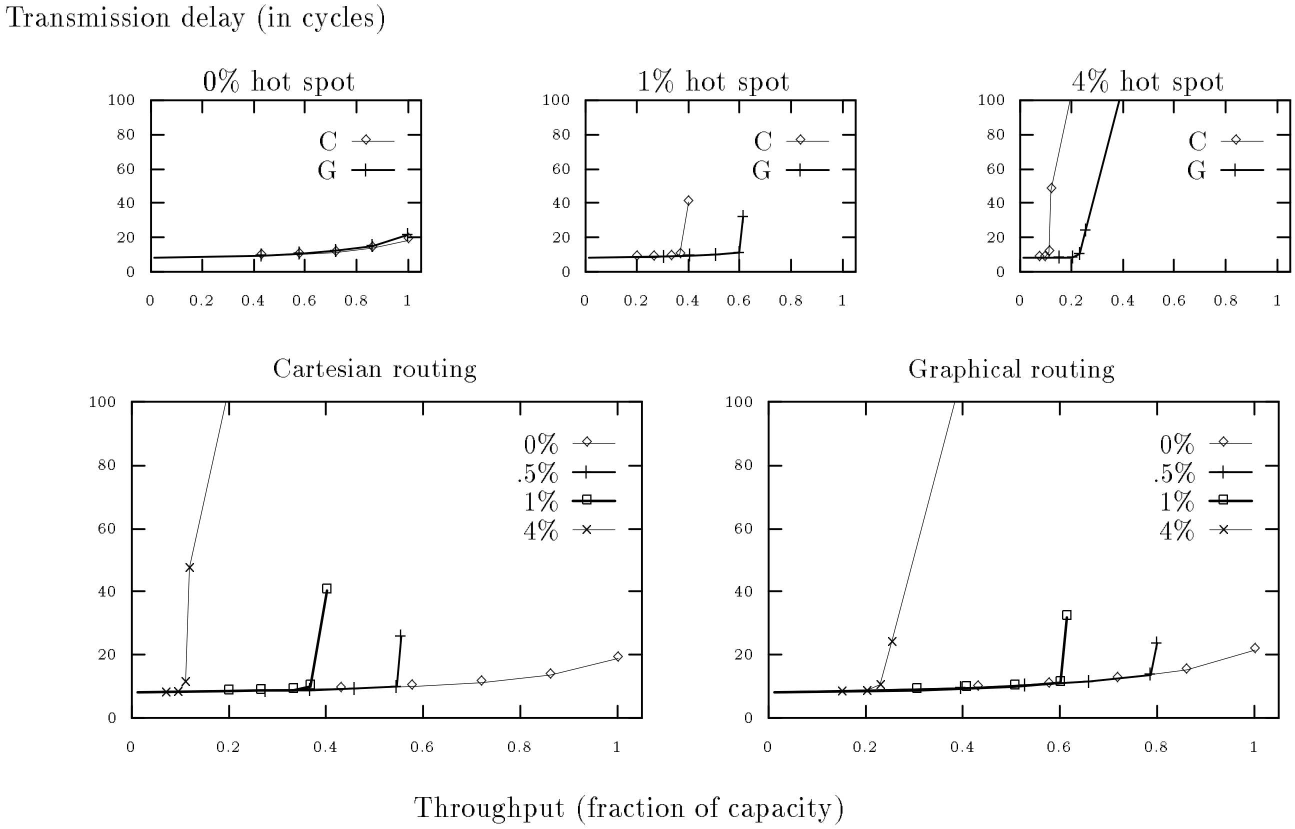

Figure 8: Communication delay versus troughput for a 5-ary 4-cube network. Each message length is $L = 1$ flit.

However, the implementation of graphical routing is more expensive, since it requires a longer header for the messages and a more complex routing chip. Following the algorithm description of Section 2, the header must include both the counters d_i $(i = 0, \ldots, n-1)$ and the increments n'_i, plus the value $2L$, which is used as a stop criterion[6]. The routing chip must be able to find the smallest d_j, add n'_j to it, and switch the message to the corresponding output channel. Cartesian routing, on the other hand, only requires the presence of the n_i in the header and the routing chip has to find the first non-zero n_j, decrement it by one and switch the message to the corresponding output channel.

Although a higher throughput can be obtained by using graphical routing, local conflicts are more likely to occur due to the equal distribution of the channel usage in the message paths. In fact, while the probability for a message to use an output channel at a certain node is strongly dependent on the incoming channel when cartesian routing is used, it is independent when graphical routing is used. In graphical routing, whenever a message arrives at an input channel of a node, the probability that there will be a conflict on an

[6]The size of these numbers are in $O(n \log k)$, which is affordable even for large systems.

output channel is

$$P_g = n\left(\frac{1}{n} - \frac{1}{k^n - k}\right)^n \sim \frac{1}{n^{n-1}} ,$$

where the first term derives from the uniform distribution in the message path, and the second term is the probability that the node is the destination node. In cartesian routing the probability that there will be a conflict is bounded above by

$$P_c = \frac{n}{k^{\frac{n(n-1)}{2}}} ,$$

which is significantly smaller. This means that, in order to sustain high throughput, buffers may have to be used at each node. Determining an optimal number of buffers will require further research. This is the cost that must be paid in order to obtain higher communication performances when the message distribution is non-uniform.

6 Conclusions

In this paper we have presented a new routing algorithm, derived from a Computer Graphics line drawing algorithm, which represents an alternative to the widely used e-cube algorithm. This algorithm improves significantly the communication performance in presence of hot spots, especially on low-dimensional interconnection networks, which are becoming increasingly popular because of their cost-effectiveness. The inherent presence of a non-uniform message distribution in a large range of parallel applications justifies the design of a routing chip integrating the proposed solution.

This study opens a new range of questions. For planar topologies, since the width of the network in each dimension is large, neighbourhood relationships could be rethought to optimize the traffic flow. A careful analysis is also needed on the properties of routing algorithms; in particular, the properties needed by an algorithm to efficiently use an interconnection network have to be outlined. The uniform distribution of the directions in the path is one such example.

Acknowledgment

The authors would like to thank M. Stumm of the University of Toronto, for his helpful comments in the revision process of the paper.

References

[1] Beyer, W., Ed. *Standard Mathematical Tables.* ISBN 0-87819-623-4. CRC Press, Cleveland, Ohio, 1976.

[2] Bresenham, J. Algorithm for computer control of a digital plotter. *IBM Systems journal 4*, 1 (Jan. 1965).

[3] Dally, W. Performance Analysis of k-ary n-cube Interconnection Networks. *IEEE Transactions on Computers 39*, 6 (June 1990).

[4] Dally, W., and Seitz, C. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers 36*, 5 (May 1987), 547–553.

[5] Germain, C., Béchennec, J., Etiemble, D., and Sansonnet, J. An Interconnection Network and a Routing Scheme for a Massively Parallel Message-Passing Multicomputer. In *Proceedings of Frontiers'90* (Maryland, Oct. 1990), pp. 368–371.

[6] Kermani, P., and Kleinrock, L. Virtual Cut-Through: A New Computer Communication Switching Technique. *Computer Networks 3* (1979), 267–286.

[7] Pfister, G., and Norton, V. Hot Spot Contention and Combining in Multistage Interconnection Networks. *IEEE Transactions on Computers 34*, 10 (Oct. 1985).

[8] Pham, S. On the Boundary of Digital Straight Line Segments. In *Advanced Computer Graphics, Proceedings of Computer Graphics Tokio 1986* (Tokio, 1986), T. Kunii, Ed., Springer Verlag, pp. 79–109.

[9] Scherson, I., and Corbett, P. Communications Overhead and the Expected Speedup of Multidimensional Mesh-Connected Parallel Processors. *Journal of Parallel and Distributed Computing 11* (1991), 86–96.

[10] Stone, H. *High Performance Computer Architecture*. Addison-Wesley, 1987.

[11] Wüthrich, C. A., and Stucki, P. An algorithmic comparison between square- and hexagonal-based grids. *Computer Vision, Graphics, and Image Processing (CVGIP) Graphical Models and Image Processing 53*, 4 (July 1991), 324–339.