Efficient Personalized Communication on Wormhole Networks

Fabrizio Petrini

International Computer Science Institute
1947 Center Street, Berkeley, CA 94704, USA
e-mail: `petrini@icsi.berkeley.edu`

Marco Vanneschi

Dipartimento di Informatica
Corso Italia 40, 56125 Pisa
e-mail: `vannesch@di.unipi.it`

Abstract

Bridging models, as the BSP, tend to abstract the characteristics of the interconnection networks using a small set of parameters, by dividing the computation in supersteps and organizing the communication in global patterns called h-relations.

In this paper we evaluate, through experimental results conducted on a wormhole-routed bi-dimensional torus and a quaternary fat-tree with 256 processing nodes, the execution time of three families of h-relations with variable degree of imbalance. We also prove a strong result that links the communication performance of the fat-tree with the BSP abstraction of the interconnection network. Given a generic h-relation, we can provide a value of g that, in the worst case, slightly overestimates the completion time and is very close to optimality.

Keywords: performance evaluation, parallel architectures and computation models, h-relations, BSP, interconnection networks, k-ary n-cubes, k-ary n-trees.

1 Introduction

An important problem in parallel computing is to the determine the extent to which one can write parallel programs using a high-level general-purpose and architecture-independent programming model and have them executed on a variety of parallel and distributed architectures without sacrificing efficiency [1].

Programming models should reflect the *cost* of executing operations at a level that is visible to the software. The information provided by the cost model on the target machine should be as little and as abstract as possible. One of the most difficult tasks is to incorporate the effect on performance of the target computer's interconnection network, both because it is difficult to describe a topology by a few parameters and because congestion is usually the most important, but hardest to model, factor. In the last few years, several interesting models have been proposed in the literature.

The *Postal* model was introduced by Bar-Noy and Kipnis [2]. In this model, the system consists of N nodes, processors with local memory. Each node can simultaneously use a single input port and a single output port and the communication latency is λ. Namely, each node during each communication step can send a fixed-size packet and at the same time receive a packet of the same size. Furthermore, a packet sent at step t will incur a latency of λ and will arrive at the receiving node at time $t+\lambda-1$. This model has been successfully used in [3] to analyze the performance of two collective communication patterns, the broadcast and the global combine.

The LogP model is a generalization of the Postal model. It reflects the convergence of parallel machines towards systems formed by a collection of complete computers, each consisting of an off-the-shelf processor, cache, and large main memory, connected by a communication network. Specifically, LogP is a model of a distributed memory multiprocessor in which processors communicate through point-to-point messages and whose performance is characterized by the following parameters: L, an upper bound on the latency, or delay, incurred in communicating a message, o, the message handling overhead, g, the gap, which is the reciprocal of the available per-processor communication bandwidth and P, the number of processor/memory modules. Examples of utilization of LogP are shown in [4], [5] and [6].

A model that has recently gained wide consideration in the scientific community is the BSP (*Bulk-Synchronous Parallel*) model [7] [8], because is neither too abstract nor too low-level and it seems to meet many of the requirements listed above [9]. It was proposed by Valiant as a bridging model that provides a standard interface between the domains of parallel architectures and algorithms. In the BSP model, a parallel architecture consists of a set of processors, each with its own private memory, that execute some virtual processes, and an interconnection network that can route packets of some fixed size between processors [10]. The computation is divided into *supersteps* and in each superstep a processor can perform operations on local data, and send and receive packets [11] [12]. A packet sent in one superstep is delivered to the destination processor at the beginning of the next superstep. Consecutive supersteps are separated by a global synchronization of all processors. It [13] it is shown that the BSP and LogP models can efficiently simulate one another, and that there is therefore no loss of performance in using the more structured BSP programming style.

BSP captures the properties of the interconnection network by two architectural parameters: the gap g, often called *permeability*, which reflects the network bandwidth on a per-processor basis and the latency L, which is a minimum duration of a superstep and which reflects the latency to send a packet through the network as well as the overhead to perform a global synchronization. The global communication pattern that takes place at the end of the superstep is called h-relation. Personalized communication and h-relations have been the subject of many research studies [14] [15] [16] [17].

In this paper we study the h-relation communication pattern in the context of wormhole interconnection networks. Wormhole networks represent the state-of-the-art in communication technology and most existing research prototypes and commercial machines adopt this switching technique.

In our study we analyze the execution times of three families of h-relations, each with an increasing degree of imbalance, on two topologies with 256 nodes, a bidimensional toroidal cube and a quaternary fat-tree. We use a detailed register-level simulation model to evaluate the impact of several routing algorithms and flow control techniques, e.g. the utilization of virtual channels [18]. The simulator allows a deep analysis on the behavior of the interconnection network and it provides insight on the relations between architectural aspects and the programming model that are often difficult to capture in a real parallel machine.

The main contribution of this paper is the experimental proof that an arbitrary h-relation can be routed on a fat-tree with a delay that is (1) very close to optimality and (2) easily predictable. This delay can be simply estimated as $h * g * \alpha$ cycles, where the permeability g is the inverse of the unidirectional link bandwidth and α a parameter that represents the routing and flow control overhead. α typically belongs to a narrow range, between 1.4 and 1.1. Also, counter intuitively, the more imbalanced is the h-relation, the smaller is α.

Unfortunately, this result cannot be generalized to the toroidal cubes. In this topology we can only determine a looser estimate of the execution time. The permeability of the network can be conservatively determined using the bisection bandwidth.

Though partially positive, these results show that the BSP way of reasoning in terms of collective communication patterns rather than in terms of the single interprocessor communication is a viable solution to model the behavior of the interconnection network with a simple and elegant model. This approach is also very efficient in practice.

The remainder of this paper is organized as follows. Section 2 describes the class of the k-ary n-cubes and the routing algorithms used in the experimental evaluation, a deterministic algorithm and a minimal adaptive algorithm based on Duato's methodology. Section 3 introduces the fat-trees and the k-ary n-trees and the adaptive routing algorithm implemented in these topologies. Section 4 presents the relevant details of the simulation model and Section 5 describes the types of h-relations and how the processing nodes schedule the transmission of the local information. The experimental results are reported in Section 6 and some conclusive remarks are given in Section 7.

2 k-ary n-cubes

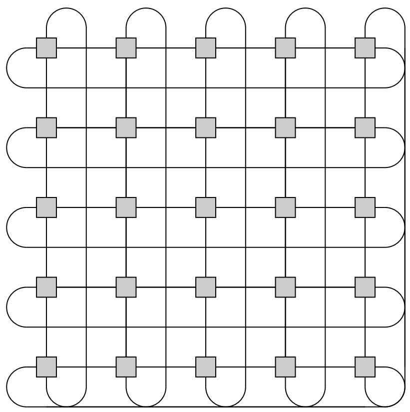


Figure 1: A toroidal 5-ary 2-cube.

Both meshes and hypercubes belong to the general class of the k-ary n-cube networks. A k-ary n-cube is characterized by its dimension n and radix k, and has a total of k^n nodes. The k^n nodes are organized in an n-dimensional mesh, with k nodes in each dimension. The binary hypercube is a special case of k-ary n-cube with $k = 2$. Also, the two dimensional mesh is another special case with $n = 2$. Figure 1 shows an example of toroidal k-ary n-cube.

Routing algorithms on the toroidal k-ary n-cubes are deadlock-prone and require sophisticated strategies for deadlock-avoidance. In this paper we utilize two algorithms, each offering a different degree of adaptivity:

deterministic and minimal adaptive based on *Duato*'s methodology [19].

The deterministic algorithm is a dimension order routing based on a static channel dependency graph [20]. Packets are sent to their destination along a unique minimal path. The potential deadlocks caused by the wrap-around connections are avoided doubling the number of virtual channels and creating two distinct virtual networks. Packets enter the first virtual network and switch to the second virtual network upon crossing a toroidal connection. Our version uses four virtual channels for each physical link.

Rather than using a static channel dependency graph, Duato's methodology only requires the absence of cyclic dependencies on a connected channel subset. The remaining channels can be used in almost any way. We associate four virtual channels to each link: on two of these channels, called *adaptive* channels, packets are routed along any minimal path between source and destination. The remaining two channels are *escape* channels where packets are routed deterministically when the adaptive choice is limited by network contention [21]. A similar algorithm has been recently adopted by the Cray T3E [22].

A central point of our adaptive algorithm is the interface between the processor and the router. We assume that packets can enter the network using only a subset of the adaptive channels [23]. This limitation, known as *source throttling*, makes the network throughput stable when the network operates above saturation [24].

3 k-ary n-trees

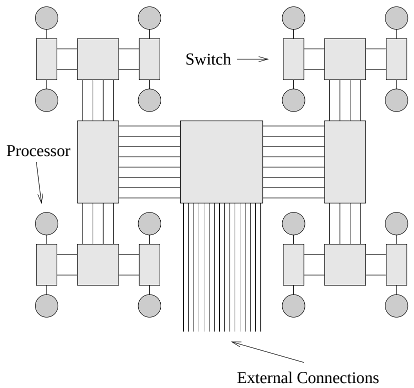


Figure 2: The structure of a fat-tree. Processors are located at the leaves, while internal nodes contain switches. At the root there are some external connections available to recursively build a bigger network or to interface it to the external world.

The fat-tree is an indirect interconnection network based on a complete binary tree. Unlike traditional trees in computer science, fat-trees resemble real trees, because they get thicker near the root. A set of processors is located at the leaves of the fat-tree and each edge of the underlying tree corresponds to a bi-directional channel between parent and child.

The arity of the internal switches of the fat-tree increases as we go closer to the root: this makes the physical implementation of these switches unfeasible. For this reason some alternative constructions have been proposed that use building blocks with fixed-arity [25]. These solutions trade connectivity with simplicity: incoming messages at a given switch in a "full" fat-tree may have more choices in the routing decision than in a corresponding network with fixed-arity switches.

Fat-trees have been adopted by many research prototypes and commercial machines. The data network of the Connection Machine CM-5 uses two distinct fat-trees [26]. The network is composed of routing chips that have either 2 or 4 parent connections. The hierarchical nature of the fat-tree is exploited to partition the CM-5 in dedicated subnetworks whose communication traffics do not interfere between them.

The Data Diffusion Machine (DDM) is a virtual shared memory architecture that implements a hierarchical COMA cache coherence protocol [27]. The distributed directory, which typically becomes a bottleneck in hierarchical bus-based implementations, is implemented in the internal switches of a fat-tree [28].

The communication chip Elite is the basic building block of the Meiko CS-2 network [29]. This network takes the form of a quaternary fat-tree. Its design is based on a multistage network and has the property that the overall communication bandwidth remains constant at each level. The CS-2 uses a randomized routing algorithm with header striping. The communication processor Elan, which interfaces each processing node to the network, attaches at the beginning of each outgoing message a string that is used by the communication processors to route the message in the ascending and descending phases. Other references to fat-trees include [30] [31].

Unfortunately, not much is known on the communication performance of the fat-trees. Most of the literature deals with the CM-5 and focuses on raw network performance [32] [33] [34]. Typical communication patterns include simple sends and ping-pong between pairs of nodes. Block permutations of data and grid shifts have been shown to have little or no contention on the CM-5. This makes the data network very efficient for regular communication patterns commonly used in numerical algorithms as the FFT. Heller [35] provided an analytical description of a class of such permutations, defined as *congestion-free*.

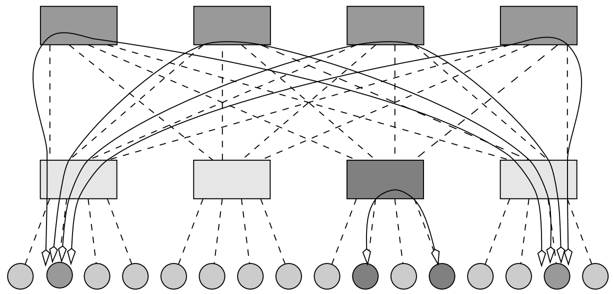


Figure 3: A 4-ary 2-tree is composed of 16 leaf nodes and 2 levels with 4 switches on each level. A packet can follow any minimal path passing through a nearest common ancestor of source and destination.

k-ary n-trees [36] are a particular subclass of the fat-trees and borrow from the k-ary n-butterflies [37] the topology of the internal switches. A k-ary n-tree has k^n leaf nodes and n levels of k^{n-1} switches. Each switch has $2k$ links. A 4-ary 2-tree is shown in Figure 3

Minimal adaptive routing between a pair nodes on a k-ary n-tree can be easily accomplished sending the packet to one of the common roots or *nearest common ancestors* (NCA) of source and destination and from there to the destination. That is, each packet experiences two phases, an *ascending* adaptive phase to get to one of the NCA, followed by a *descending* deterministic phase. The performance of a wormhole routed network can be enhanced mapping two or more virtual channels on each physical channel [18]. In our experiments we will examine three variants of the adaptive algorithm, with one, two and four virtual channels. They simply pick the less loaded link in the ascending phase, that is the link that has the maximum number of free virtual channels (a fair choice is made when more links are in a similar state).

4 Relevant details of the network model

This section presents the router model and the simulation environment, that are used in the following sections to analyze the performance of the k-ary n-trees and the k-ary n-cubes under three families of h-relations.

Figure 4 outlines the internal structure of a routing switch. We can distinguish the external channels or links, the input and the output buffers or lanes that implement the buffer space of the virtual channels and an internal crossbar. The switch has bidirectional channels and each channel on the single direction is logically composed of three interfaces: a *data path* that transmits messages on a flit level, the *ready* lines that flag the presence of a flit on the data path and specify the virtual channel where the flit is to be stored and the *ack* lines in the reverse direction that send an acknowledgment every time buffer space is released in the input lanes. The processing nodes have a compatible interface with the same number of virtual channels.

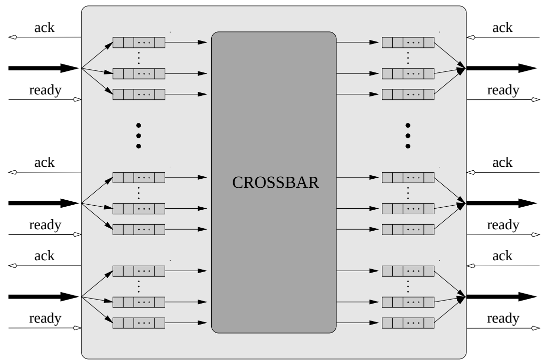


Figure 4: The internal structure of a routing switch.

A flit is moved from an output lane to the corresponding input lane in a neighboring node in T_{link} cycles, when there is at least a free buffer position. Each output lane has associated a counter that is initialized with the total number of buffers in the input lane, it is decremented after sending a flit and it is incremented upon receiving and acknowledgment. When multiple lanes are enabled, an arbiter picks one of them according to a fair policy.

When a header flit reaches the top of an input lane, the routing algorithm tries to establish a path in the crossbar with a suitable output lane, that is neither full nor bound to another input lane. This path will remain in action till the transmission of the tail flit of the packet. Our model allows the parallel routing of multiple headers every $T_{routing}$ cycles.

Although a physical link services in each direction at most one virtual channel every T_{link} cycles, multiple virtual channels can be active at the input and output ports of the crossbar. The internal flit propagation takes $T_{crossbar}$ cycles. Every time a flit is moved from an input lane to the corresponding output lane, a feedback is sent back to the neighboring switch or node to update the counter of free positions. In our experiments the three delays are equalized to a single clock cycle. Each input or output lane has capacity for 4 flit buffers.

This model is evaluated in the SMART (Simulator of Massive ARchitectures and Topologies) environment [38]. Implemented in C++, SMART is an object-oriented discrete-event simulation tool for evaluating massively parallel architectures. Configuring some shell scripts, it is possible to select the network topology and the internal router policies. The simulator allows the definition of the packet length, number of virtual channels and buffers for both input and output lanes. Also, it is possible to monitor several metrics and time-dependent events, that are gathered in trace files. At the moment, SMART supports three families of topologies: k-ary n-cubes, k-ary n-flies and k-ary n-trees and a node architecture with processing capabilities and a memory hierarchy .

A fair analysis of interconnection networks should take into account physical constraints as the pin count, wire delay, bisection width [39] and the router complexity [40]. In our experiments we normalize the communication performance by setting the flit and the data path size on the fat-tree at two bytes and at four bytes on the toroidal cube. If we consider two representative networks with 256 nodes, a 4-ary 4-tree and a 16-ary 2-cube, this normalization can be interpreted in the two following ways.

1. Technological constraints limit the number of pins on a given chip. In a quaternary fat-tree, the arity of the routing switches is eight, while the arity of the routing chip in a bi-dimensional cube is four, if we do not consider the connection with the local processing node. By doubling the data paths on the cube we have the same pin count on both routing chips.

2. Both k-ary n-tree and the toroidal k-ary n-cubes have $n * k^n$ links. Thus the quaternary fat-tree has got twice as many links as a bi-dimensional torus and our normalization equalizes the overall number of communicating links.

5 h-relations

The h-relation denotes the personalized communication where each node has at most h packets to send to various points in the network, and where each node is also due to receive at most h packets from other nodes.

In our definition of h-relation, we replace the packets with the amount of information, expressed in bytes, that must be exchanged by a pair of nodes, in order to decouple the problem definition with the actual implementation and the flow control policies.

We consider an input data set of m bytes distributed amongst N processing nodes in such a manner that no processing node holds more than $\frac{m}{N}$ bytes. Also, each node is the destination of no more than h bytes. We assume for simplicity and without loss of generality that both m and h are integer multiples of N.

Our input of size m is initially distributed cyclically across the N nodes such that each node i initially holds $\frac{m}{N}$ bytes, for $0 \leq i \leq N-1$. For $h = \frac{m}{N}$ the input consists of v_0 bytes for node 0, followed by v_1 bytes for node 1 and so forth (with $v_i = \frac{m}{N}$ bytes for node i), with the last $v_{N-1} = \frac{m}{N}$ for node $N-1$, as shown in [16].

For $h > \frac{m}{N}$ instead of an equal number of bytes destined for each node, the function v_i, for $0 \leq i \leq N-1$, is characterized by

$$\begin{cases} \lfloor h(1 - \frac{h}{2m-h} i)\rfloor, & if\ i < \frac{2m}{h},\ i \neq N-1, \\ 0, & if\ \frac{2m}{h} \leq i < N-1, \\ m - \sum_{j=0}^{\frac{2m}{h}} \lfloor h(1 - \frac{h}{2m-h} j)\rfloor, & if\ i = N-1. \end{cases} \tag{1}$$

The result of this data movement is that node 0 receives the largest imbalance of information, i. e. h, while other nodes receive varying block sizes ranging from h to 0 as we increase the node index. The last node, $N-1$, acts as a bucket that contains the difference between the initial data set size m and the values assigned to the other nodes. The h-relation factor h_rel defines the ratio between h and $\frac{m}{N}$

$$h = h\_rel\ \frac{m}{N}. \tag{2}$$

In our experiments we will consider three types of h-relations, by setting h_rel at two, four and eight. To an increasing value of h_rel corresponds and increasing degree of imbalance. For $h\_rel = 8$, approximately $\frac{3N}{4}$ nodes receive no elements, and hence this represents an extremely unbalanced case. The final data distributions of the three h-relations are sketched in Figure 5.

5.1 Scheduling the h-relations

Each h-relation is scheduled using a simple randomized algorithm. The goal is to reproduce, as long as possible, a uniform traffic inside the network.

- Each node has a communication vector which is initialized with the amount of information that must be sent to any other node.

- The information is exchanged in packets whose maximum size is chosen in order to optimize the network throughput. In our experiments we use a packet size of 64 bytes for both networks.

- The algorithm operates in sweeps. In each sweep the communication vector is scanned using an independent permutation of the destinations. An amount of information, whose size is determined

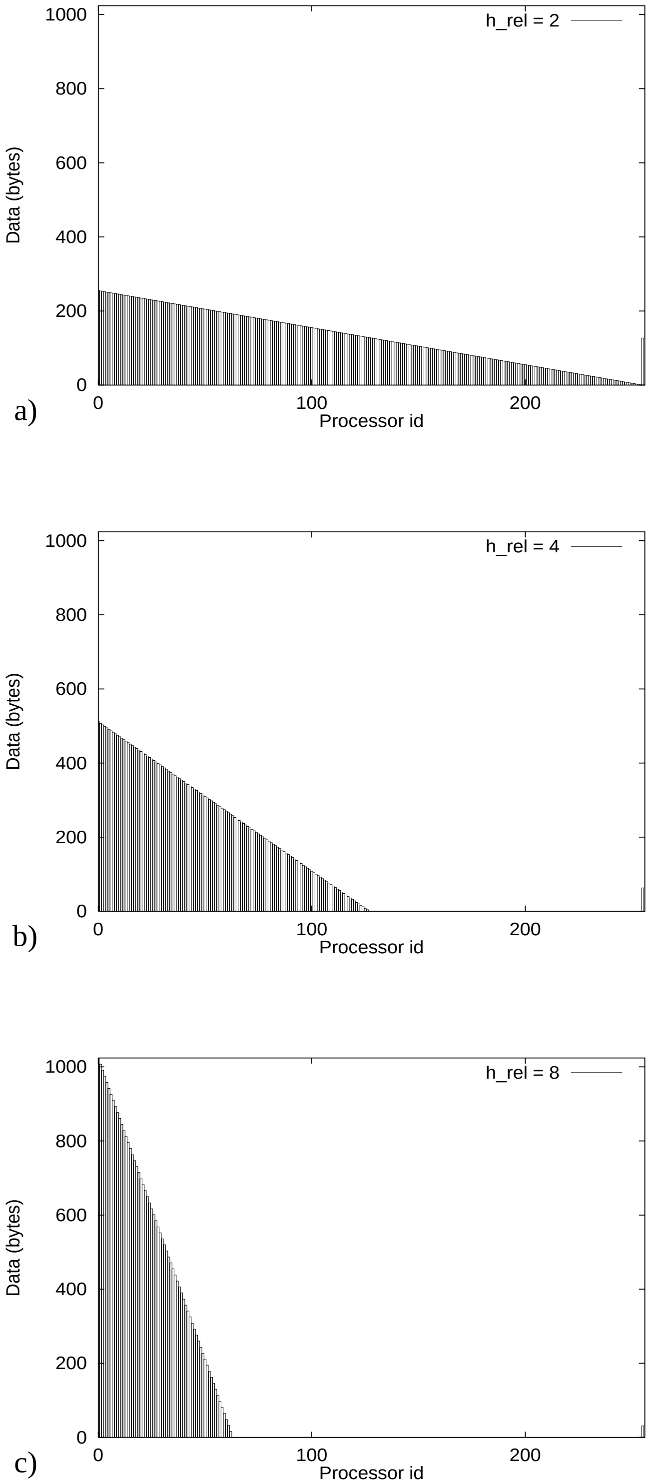


Figure 5: The final distribution of data when $h\_rel = 2$, 4 and 8.

by the packet size and the amount of data available, is sent to all nodes that have a non null entry in the communication vector.

- The entry is decremented by the corresponding packet size upon sending a packet. The algorithm terminates when all entries are null.

6 Experimental Results

In our experimental analysis we considered two interconnection networks with 256 nodes, a 16-ary 2-cube and 4-ary 4-tree.

6.1 Cube

In Figure 6 we report the execution times obtained using the deterministic and the Duato's algorithm.

The h-relation factor $h\_rel = 2$ indicates a moderate imbalance. This communication pattern is limited by the bisection bandwidth in one direction. The lower bound in Figure 6 a) is computed by considering that most of the information initially stored in the nodes with index > 127 is directed to the nodes in the other half of the topology. The asymptotic behavior shows that the ratio between the execution time and the lower bound is about 1.6 and 2.0, respectively for the deterministic and adaptive algorithms.

When the total amount of input data is small the execution time is dominated by the overhead in the initial and final phase and the steady state period is relatively short. In Figure 7 b) we can see that there is a further problem with the Duato's algorithm when $m = 32768$ bytes: after an initial spike, the network utilization falls at 10%. More insight on this problem is provided in Figure 8. Figure 8 a) displays the final distribution of data, by representing the torus as a flattened grid: the hot spot is represented by the nodes in the first half. In Figure 8 c) we can see that the adaptive algorithm generates congestion between rows 3 and 7, while the deterministic algorithm distributes the congestion along the columns by forcing the packets first along the rows until they are aligned with the destination and then along the columns.

When $h\_rel = 4$ the lower bound is represented by both the bisection bandwidth and the input bandwidth of the hot node 0 that receives h bytes. As in the previous case the execution time is less than optimal for small data sets, but converges to 1.6 times the lower bound when we use the adaptive algorithm. With the deterministic algorithm the ratio is 1.7.

When the degree of imbalance is high, the gap between the actual execution times and the lower bounds is narrow. In fact the performance is limited by the hot node. When $h\_rel = 8$ the asymptotic ratio with the lower bound is only 1.22.

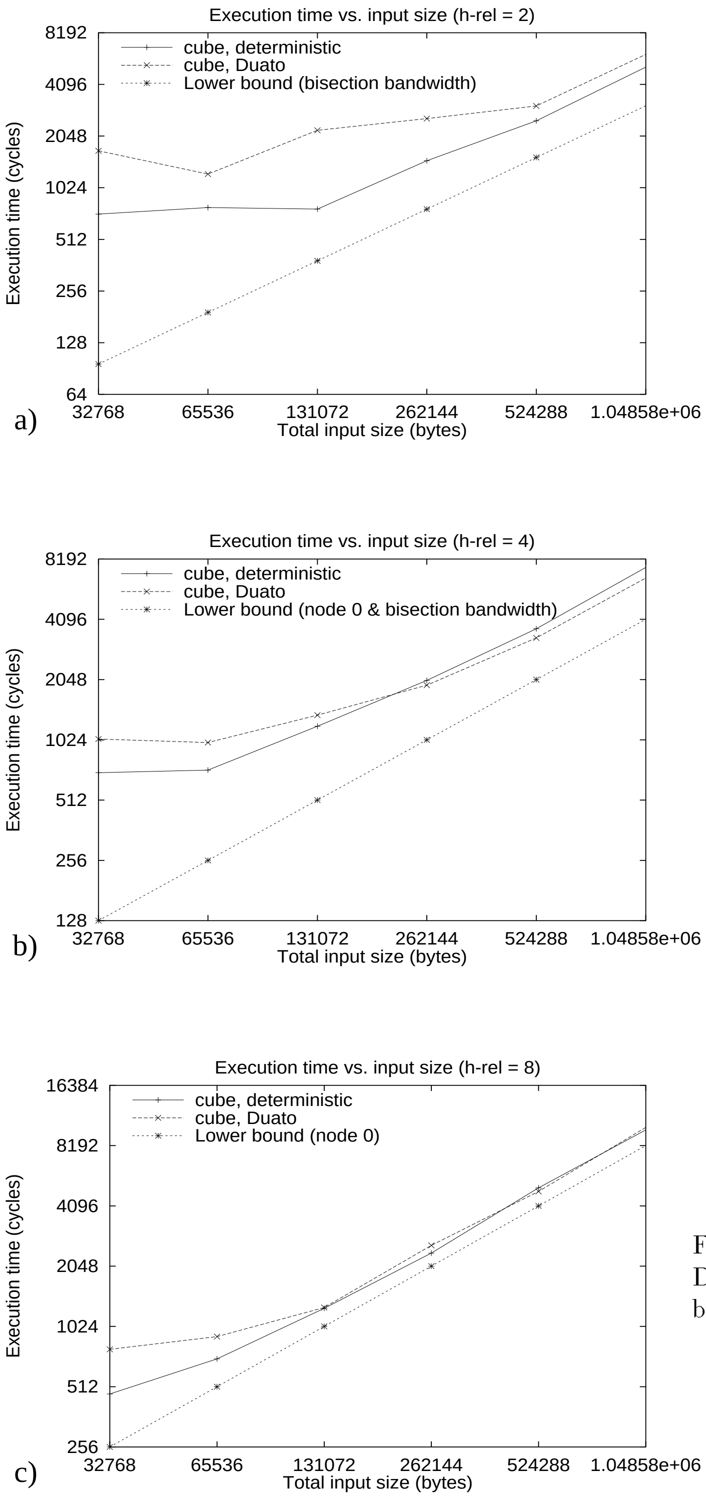


Figure 6: Execution time of the h-relations using the deterministic and the Duato's algorithm. Both axes use a logarithmic scale.

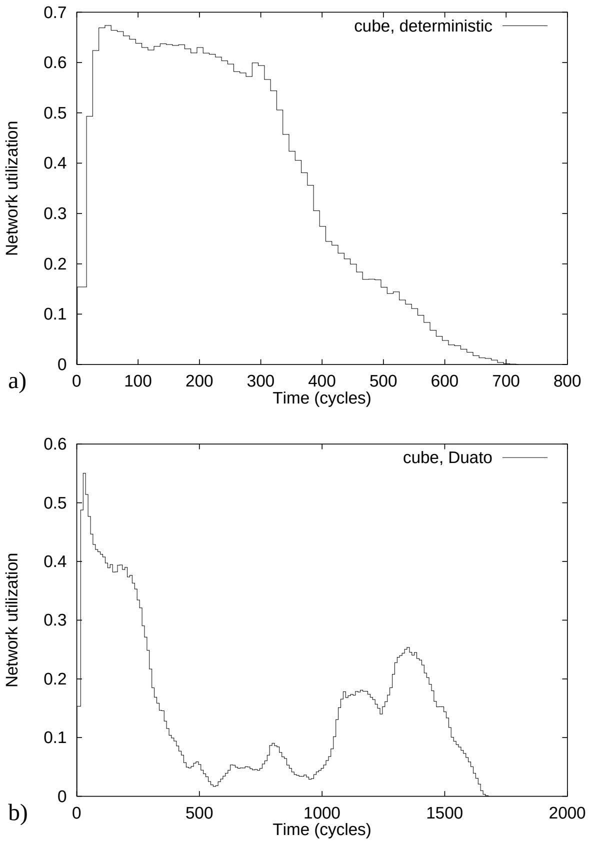


Figure 7: Network utilization of the deterministic and Duato algorithms with an input data set size $m = 32768$ bytes and h-relation factor $h\_rel = 2$.

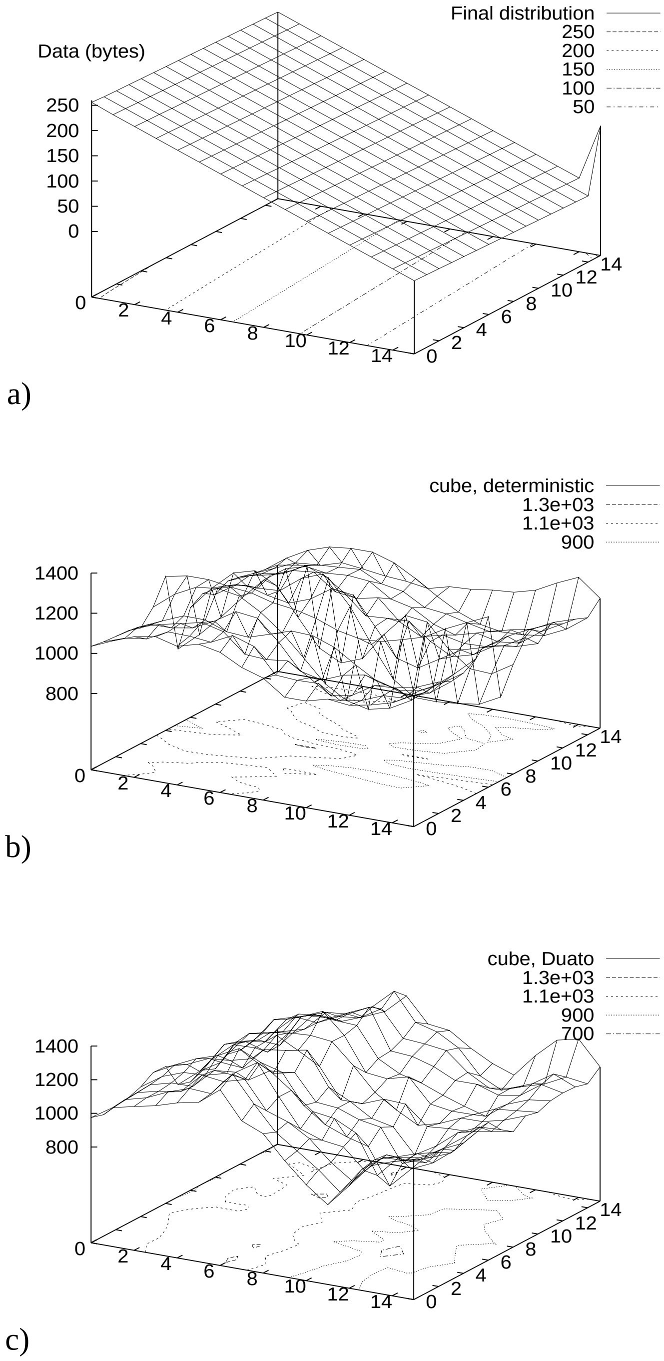


Figure 8: a) The final data distribution when $h\_rel = 2$ and $m = 32768$. b) Number of packets passing through each node with the deterministic algorithm. c) Number of packets passing through each node with the Duato's algorithm.

6.2 Fat-Tree

In Figure 9 we can see the performance of the fat-tree under the three types of h-relations, that are routed using one, two and four virtual channels. The lower bounds in the graphs are simply computed by considering the load at the hot node 0. In the fat-trees the topological limitation of any h-relation is simply given by the most loaded node. This is a very interesting property, because we can determine the lower bound without knowing any information other than the maximum amount of information exchanged. We have seen that in the cubes the topological limitation is the bisection bandwidth when the communication pattern is fairly balanced and the input bandwidth of hot nodes in other cases. This means that we need more information to provide a precise estimate of the lower bound.

The performance of the fat-tree is very sensitive to the number of virtual channels. For example, the asymptotic ratio between the execution time with one and four virtual channels is about 1.7 in all the three h-relations. An interesting cost/effective solution is represented by two virtual channels. In this case the ratio between one and two virtual channels is 1.4 and, again, this value is not sensitive to the characteristics of the h-relation. It is worth noting that virtual channels can be expensive because they complicate routing decision and channel control, increasing the router complexity and delay significantly, with a consequent cost/performance trade-off [40]. So, the number of virtual channels per physical channel should be kept small.

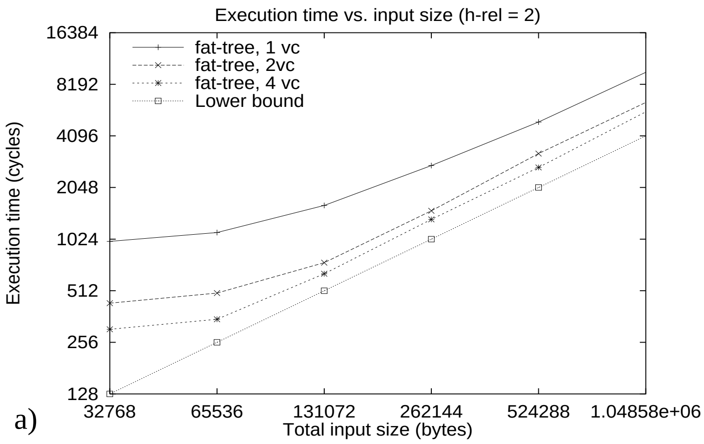
When we use four virtual channels and $h\_rel = 2$ the ratio between the execution time and the lower bound is only 1.38. When $h\_rel = 4$ and $h\_rel = 8$ the ratio is 1.21 and 1.13, respectively.

From these results we can see that the fat-tree can route h-relations very efficiently. More information on the network behavior when we use four virtual channels is shown in Figure 10. The network reaches the steady state after few cycles and then works without problems until the topological bottleneck limits the network throughput. In this case the routing algorithm does not introduce any substantial overhead.

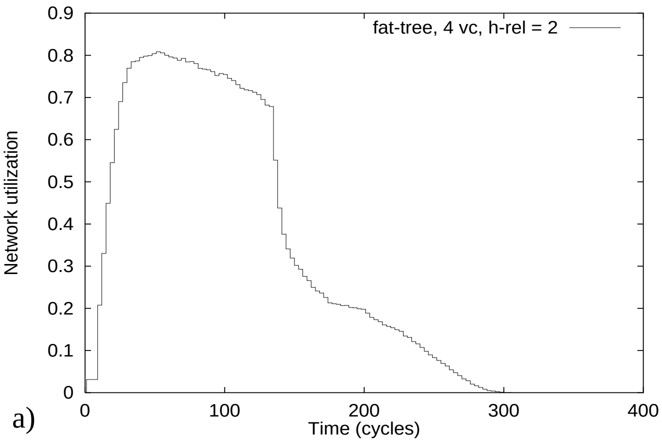
7 Conclusion

In this paper we have analyzed in depth the performance of the toroidal cube and the fat-tree with an important collective communication pattern, the h-relation.

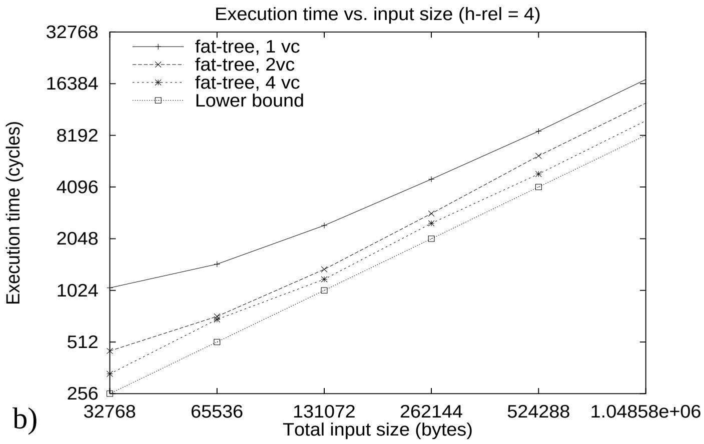
We have examined three families of h-relations, representative of three increasing degrees of imbalance. In both topologies we used a simple randomized algorithm to schedule the h-relations.



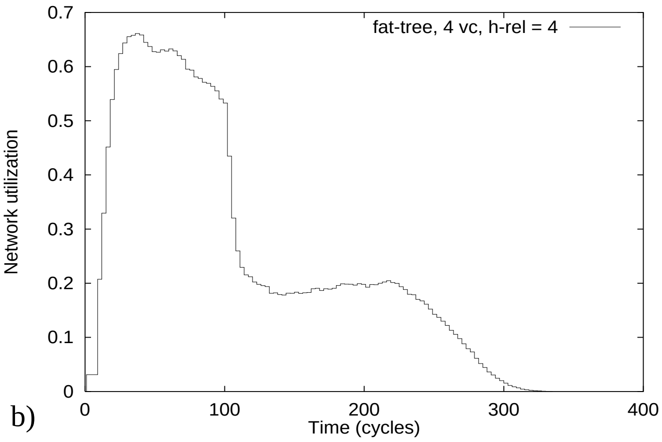
a)



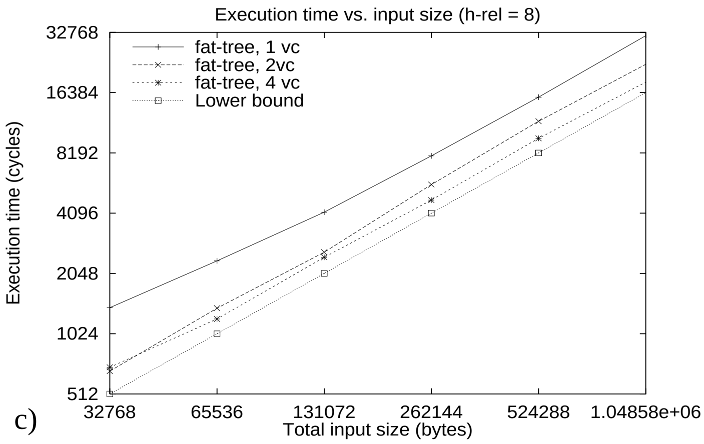
a)



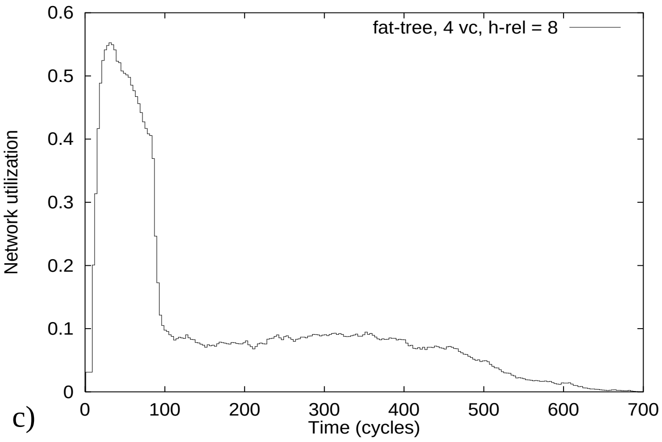
b)



b)



c)



c)

Figure 9: Execution time of the h-relations using one, two and four virtual channels. Both axes use a logarithmic scale.

Figure 10: Network utilization with four virtual channels and an input data set size $m = 32768$ bytes.

On the fat-tree, for any value of h, a strict upper bound can be determined by considering that the injection and reception bandwidth of the processing nodes are the limiting factors. The time $T(h)$ needed to route an h-relation can be simply estimated, with a good approximation, as

$$T(h) = h * g * \alpha \tag{3}$$

where the permeability g is the inverse of the unidirectional link bandwidth and α a parameter that represents the routing and flow control overhead. α belongs to a narrow range, between 1.4 and 1.1. The more imbalanced is the h-relation, the smaller is α. For balanced h-relations the value is approximately 1.4, using four virtual channels and the router architecture shown above. We consider this result as a sort of "missing link" between the BSP programming model and the computer architecture. We can provide a strict, reliable and realistic cost calculus to the programming model. The result does not rely on any simplification or compromise: we have used a detailed register-level simulation of the routing chip. The largest α, representing the worst case, is obtained with a balanced h-relation. This result can be used to determine a strict upper bound to permeability of the BSP model, that is $1.4 * g$. In the case of a highly imbalanced h-relation we overestimate the completion time of only 25%.

Unfortunately, the same propriety does not hold for the toroidal cubes. This can be explained in the following way. First of all, there isn't a unique topological limiting factor for any h-relation. In the cube, balanced h-relations are limited by the bisection bandwidth, e. g. when $h\_rel = 2$. In a highly imbalanced h-relation ($h\_rel = 8$), the bottleneck is the input bandwidth of the hot node. With uniform traffic, in a radix k toroidal cube, each node can inject at capacity only $4/k$ flits per cycle[1]. In a 16-ary cube this means that only 1/4 of the injection/reception bandwidth can be effectively utilized. In this case the lower bound is four time higher than the full injection bandwidth.

Also, there are problems in the routing algorithms, that introduce an overhead α that can be as high as 2.0. In this case we cannot provide a tight estimate of the completion time of the h-relation (it can be an order of magnitude bigger). We need more information on the type of communication pattern: for example the degree and the "type" of imbalance can help to determine the lower bound and, consequently a good estimate of the completion time. The permeability parameter g can be conservatively estimated by considering the bisection bandwidth as the limiting factor of the h-relation (that is we assume that the the pattern is fairly balanced). But, this can be potentially inaccurate.

On the other hand, cubes are favored over the fat-trees by the wider data paths on h-relations with high skew. In fact, when the communication patterns is imbalanced the bottleneck becomes the injection, or reception, bandwidth of the most loaded nodes. The small difference between the deterministic algorithm and the one with four virtual channels when $h\_rel = 2$ becomes wider when we move to $h\_rel = 4$ and $h\_rel = 8$.

[1] The network capacity under uniform traffic can be determined by considering that 50% of the traffic crosses the bisection of the network. Thus, if a cube has bisection bandwidth B, each of the N nodes can inject $2B/N$ traffic at the maximum load.

References

[1] D. B. Skillicorn and D. Talia, "Models and Languages for Parallel Computation." submitted to ACM Computing Surveys, 1996.

[2] A. Bar-Noy and S. Kipnis, "Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems," in *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pp. 13–22, June 1992.

[3] J. Bruck, L. D. Coster, N. Dewulf, C.-T. Ho, and R. Lauwereins, "On the Design and Implementation of Broadcast and Global Combine Operations Using the Postal Model," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, pp. 256–265, March 1996.

[4] D. Culler, L. T. Liu, R. P. Martin, and C. Yoshikawa, "Assessing Fast Network Interfaces," *IEEE Micro*, vol. 16, pp. 35–43, February 96.

[5] A. C. Dusseau, D. E. Culler, K. E. Schause, and R. P. Martin, "Fast Parallel Sorting Under LogP: Experience with the CM-5," *IEEE Transactions on Parallel and Distributed Systems*, vol. 7, pp. 791–805, August 1996.

[6] J. R. Larus, "Compiling for Shared-Memory and Message-Passing Computers," *ACM Letters on Programming Languages and Systems*, vol. 2, pp. 165–180, March–December 1994.

[7] L. G. Valiant, "A Bridging Model for Parallel Computation," *Communications of the ACM*, vol. 33, pp. 103–111, August 1990.

[8] W. F. McColl, "Universal Computing," in *Second International Euro-Par Conference, Volume I*, no. 1123 in LNCS, (Lyon, France), pp. 25–36, August 1996.

[9] M. Goudreau, K. Lang, S. Rao, T. Suel, and T. Tsantilas, "Towards Efficiency and Portability: Programming with the BSP Model," in *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, (Padova, Italy), June 1996.

[10] D. B. Skillicorn, J. M. D. Hill, and W. F. McColl, "Questions and Answers about BSP," Tech. Rep. PRG-TR-15-96, Oxford University Computing Laboratory, August 1996.

[11] J. M. D. Hill and D. B. Skillicorn, "Practical Barrier Synchronization," Tech. Rep. PRG-TR-16-96, Oxford University Computing Laboratory, 1996.

[12] J. M. D. Hill and D. B. Skillicorn, "Lessons Learned form Implementing BSP," Tech. Rep. PRG-TR-21-96, Oxford University Computing Laboratory, 1996.

[13] G. Bilardi, K. T. Herley, A. Pietracaprina, G. Pucci, and P. Spirakis, "BSP vs. LogP," in *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, (Padova, Italy), June 1996.

[14] S. Ranka, R. V. Shankar, and K. A. Alsabti, "Many-to-many Personalized Communication with Bounded Traffic," in *The Fifth Symposium on the Frontiers of Massively Parallel Computation*, (McLean, VA), February 1995.

[15] R. V. Shankar, K. A. Alsabti, and S. Ranka, "The Transportation Primitive." Posted to comp.parallel, August 1994.

[16] D. R. Helman, D. A. Bader, and J. Jájá, "Parallel Algorithms for Personalized Communication and Sorting with an Experimental Study," in *Proceedings of the 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, (Padova, Italy), June 1996.

[17] S. Rao, T. Suel, T. Tsantilas, and M. Goudreau, "Efficient Communication Using Total-Exchange," in *Proceedings of the 9th International Parallel Processing Symposium, IPPS'94*, (Santa Barbara, CA), April 1995.

[18] W. J. Dally, "Virtual Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, pp. 194–205, March 1992.

[19] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, pp. 1055–1067, October 1995.

[20] 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.

[21] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Adaptive Routing in Wormhole Networks," in *International Conference on Parallel Processing*, vol. I - Architecture, pp. I–142–I–149, 1994.

[22] S. L. Scott and G. M. Thorson, "The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus," in *HOT Interconnects IV*, (Stanford University), August 1996.

[23] F. Petrini and M. Vanneschi, "Minimal Adaptive Routing with Limited Injection on Toroidal k-ary n-cubes," in *Supercomputing 96*, (Pittsburgh, PA), November 1996.

[24] W. J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, pp. 466–475, April 1993.

[25] C. E. Leiserson and B. M. Maggs, "Communication-Efficient Parallel Algorithms for Distributed Random Access Machines," *Algorithmica*, vol. 3, pp. 53–77, 1988.

[26] C. E. Leiserson *et al.*, "The Network Architecture of the Connection Machine CM-5," in *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 272–285, June 1992.

[27] H. L. Muller, P. W. A. Stallard, and D. H. D. Warren, "The Data Diffusion Machine with a Scalable Point-to-Point Network," Tech. Rep. CSTR-93-17, Department of Computer Science, University of Bristol, October 1993.

[28] H. L. Muller, P. W. A. Stallard, and D. H. D. Warren, "An Evaluation Study of a Link-Based Data Diffusion Machine," in *Proceedings of the 8th International Parallel Processing Symposium, IPPS'94*, (Cancun, Mexico), pp. 115–128, April 1994.

[29] Meiko World Incorporated, *Computing Surface 2 reference manuals*, preliminary ed., 1993.

[30] S. Haridi and E. Hagersten, "The Cache Coherence Protocol of the Data Diffusion Machine," in *PARLE'89, Parallel Architectures and Languages Europe*, vol. I, pp. 1–18, June 1989.

[31] Kendall Square Research, *Technical Summary*, 1^{th} ed., 1991.

[32] T. T. Kwan, B. K. Tatty, and D. A. Reed, "Communication and Computation performance of the CM-5," in *Supercomputing'93*, pp. 192–201, November 1993.

[33] M. Lin, R. Tsang, D. H. C. Du, A. E. Klietz, and S. Saroff, "Performance Evaluation of the CM-5 Interconnection Network," Tech. Rep. AHPCRC Preprint 92-111, University of Minnesota AHPCRC, October 1992.

[34] A. Martin and D. Bader, "Performance of the CM-5 ENEE 646." unpublished, January 1994.

[35] S. Heller, "Congestion-Free Routing on the CM-5 Data Router," in *First International Workshop, PCRCW'94* (K. Bolding and L. Snyder, eds.), vol. 853 of *LNCS*, (Seattle, Washington, USA), pp. 176–184, May 1994.

[36] F. Petrini and M. Vanneschi, "k-ary n-trees: High Performance Networks for Massively Parallel Architectures," in *Proceedings of the 11th International Parallel Processing Symposium, IPPS'97*, (Geneva, Switzerland), pp. 87–93, April 1997.

[37] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* San Mateo, CA, USA: Morgan Kaufmann Publishers, 1992.

[38] F. Petrini and M. Vanneschi, "SMART: a Simulator of Massive ARchitectures and Topologies," in *International Conference on Parallel and Distributed Systems Euro-PDS'97*, (Barcelona, Spain), June 1997.

[39] A. Agarwal, "Limits on Interconnection Network Performance," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, pp. 398–412, October 1991.

[40] A. A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers," in *Hot Inteconnects '93*, (Palo Alto, California), August 1993.