k-ary n-trees: High Performance Networks for Massively Parallel Architectures

Fabrizio Petrini and Marco Vanneschi

Dipartimento di Informatica, Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy,
tel +39 50 887228, fax +39 50 887226
e-mail: {petrini,vannesch}@di.unipi.it

Abstract

The past few years have seen a rise in popularity of massively parallel architectures that use fat-trees as their interconnection networks. In this paper we study the communication performance of a parametric family of fat-trees, the k-ary n-trees, built with constant arity switches interconnected in a regular topology. Through simulation on a 4-ary 4-tree with 256 nodes, we analyze some variants of an adaptive algorithm that utilize wormhole routing with one, two and four virtual channels. The experimental results show that the uniform, bit reversal and transpose traffic patterns are very sensitive to the flow control strategy. In all these cases, the saturation points are between $35-40\%$ of the network capacity with one virtual channel, 55–60% with two virtual channels and around 75% with four virtual channels. The complement traffic, a representative of the class of the congestion-free communication patterns, reaches an optimal performance, with a saturation point at 97% of the capacity for all flow control strategies.

Keywords: Interconnection networks, fat-trees, k-ary n-trees, routing, virtual channels, flow control, congestion-free patterns.

1 Introduction

The fat-tree is an indirect interconnection network based on a complete binary tree [9]. 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 bidirectional channel between parent and child. A channel consists of a bundle of wires and the number of wires in a channel is called its capacity. These capacities are determined by how much hardware we can afford: this means that a fat-tree is parameterized not only in the number of processors, but also in the communication bandwidth it can support.

Routing on a fat-tree is relatively easy, since there is a unique minimal path between a pair of processors i and j: a message going from i to j goes up the internal switches of the tree until it finds the nearest common ancestor and then down to j. In the presence of channels with multiple capacity, the flow control algorithm can pick one of these wires in order to evenly distribute the messages and to minimize congestion.

Fat-trees have many nice theoretical properties. Leiserson [9] proved that, for an arbitrary message set M, *off-line* scheduling can be done optimally within a logarithmic factor of the number of processors. That is, M can be scheduled in a group of delivery cycles $M_1, M_2, \ldots M_d$ such that $d = O(\lambda \log N)$, where N is the number of processors and λ is the *load factor*. Also, fat-trees are hardware-efficient networks. The *universality theorem* from Leiserson states that a universal fat-tree of a given volume can simulate any other interconnection network of equal volume with only a polylogarithmic factor increase in the time required. These results have been extended to the more interesting *on-line* case [5]: if a set of messages has load factor λ on a fat-tree with N processors, the number of delivery cycles required is $O(\lambda + \log N \log\log N)$ with probability $1 - O(1/N)$.

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 [11]. 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. DeHon [3] provided an in-depth study of implementation problems, as wiring and packaging complexity and fault-tolerance. Orthogonal fat-trees are an alternative formulation that uses constant size elements. The elegant recursive construction developed by Valerio *et al.* [16] tries to maximize the number of processors when the degree of the internal switches and the diameter of the network are physically constrained.

Fat-trees have been adopted by several parallel computers as the Connection Machine CM-5 [10], the Data Diffusion Machine [15] and the Meiko CS-2 [14].

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 [7] [12] [13]. 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 [6] provided an analytical description of a class of such permutations, defined as *congestion-free*.

The remainder of this paper is organized as follows. Section 2 presents a recursive definition of fat-trees, and focuses on a parametric family of networks, the k-ary n-trees. Section 3 presents a detailed simulation model of the interconnection network for studying the impact of wormhole routing and virtual channels. The communication patterns adopted as benchmarks in the experimental evaluation are described in section 4. Using the simulation model and the benchmarks, in section 5 we evaluate the communication performance of a 4-ary 4-tree with 256 nodes. An overview of the experimental results is provided in section 6 and some concluding remarks are given in section 7.

2 k-ary n-trees

In this section we formalize the fat-trees, giving a recursive definition that is general enough to embed many different topologies that are often quoted as fat-trees. We then turn our attention to a particular subclass: the k-ary n-trees. As the k-ary n-cubes and the k-ary n-butterflies [2], the k-ary n-trees are a parametric family of regular topologies that can be built varying the two parameters k and n.

Definition 2.1 *A fat-tree is a collection of vertices connected by edges and is defined recursively as follows.*

- *A single vertex by itself is a fat-tree. This vertex is also the root of the fat-tree.*
- *If $v_1, v_2, \ldots, v_i$ are vertices and $T_1, T_2, \ldots, T_j$ are fat-trees, with $r_1, r_2, \ldots, r_k$ as roots (j and k need not to be equal), a new fat-tree is built by connecting with edges, in any manner, the vertices $v_1, v_2, \ldots, v_i$ to the roots $r_1, r_2, \ldots r_k$. The roots of the new fat-tree are $v_1, v_2, \ldots, v_i$.*

Definition 2.1 is extremely general and covers many existing examples in the literature. It includes ordinary trees, "full" fat-trees with variable-sized switches and multiple connections between vertices and irregular constructions. The only exception being the orthogonal fat-trees, that allow connections between the roots $v_1, v_2, \ldots, v_i$. Some examples are shown in Figure 1.

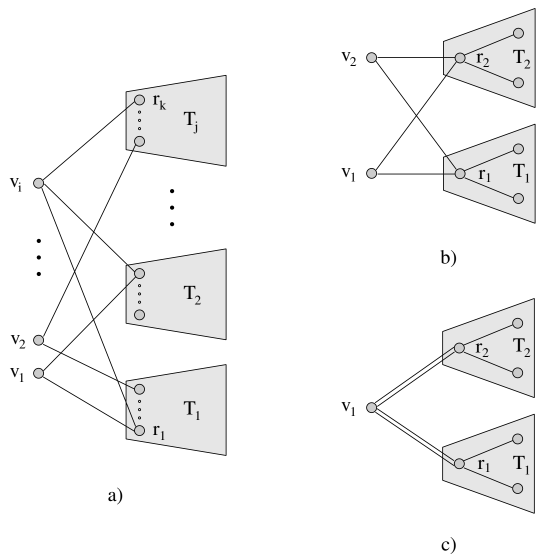

Figure 1. a) A fat-tree is recursively built connecting the new roots $v_1, v_2, \ldots, v_i$ to the roots $r_1, r_2, \ldots, r_k$ of the subtrees. b) A fat-tree with two roots. c) A fat-tree with multiple edges between the root v_1 and the roots of the subtrees T_1 and T_2.

Let's turn our attention to a particular class of fat-trees, the k-ary n-trees. k-ary n-trees borrow from a popular class of multistage interconnection networks, the k-ary n-butterflies [8] (or k-ary n-flies for short), the topology of the internal switches. The k-ary n-fly is a generalization of the butterfly and is useful to model those topologies that use communication switches with arity k greater than two. This topology has a recursive structure: one k-ary n-fly contains k butterflies of dimension $n-1$ as subgraphs. Also, each level 0 switch is linked to any level n switch by a unique path of length n, that is k-ary n-flies are *banyan* networks.

We can define the class of k-ary n-trees in the following way.

Definition 2.2 *A k-ary n-tree is composed of two types of vertices: $N = k^n$ processing nodes and nk^{n-1} $k * k$ communication switches* [1]. *Each node is an n-tuple*

[1] A k-ary n-tree of dimension $n = 0$ is composed of a single processing node

$\{0, 1, \ldots, k-1\}^n$, *while each switch is defined as an ordered pair* $\langle w, l \rangle$, *where* $w \in \{0, 1, \ldots, k-1\}^{n-1}$ *and* $l \in \{0, 1, \ldots, n-1\}$.

- *Two switches*

  $$\langle w_0, w_1, \ldots, w_{n-2}, l \rangle \quad and \quad \langle w'_0, w'_1, \ldots, w'_{n-2}, l' \rangle$$

  are connected by an edge if and only if $l' = l + 1$ *and* $w_i = w'_i$ *for all* $i \neq l$. *The edge is labeled with* w'_l *on the level* l *vertex and with* w_l *on the level* l' *vertex.*

- *There is an edge between the switch*

  $$\langle w_0, w_1, \ldots, w_{n-2}, n-1 \rangle$$

  and the processing node $p_0, p_1, \ldots, p_{n-1}$ *if and only if*

  $$w_i = p_i \quad for\ all \quad i \in \{0, 1, \ldots, n-2\}.$$

  This edge is labeled with p_{n-1} *on the level* n–1 *switch.*

It can be easily seen that a k-ary n-tree is a fat-tree, according to Definition 2.1. In fact, the level 0 switches $\langle w, 0 \rangle$ are the roots of a k-ary n-tree whose subtrees are $(n-1)$-dimensional k-ary n-trees. Also, the labeling scheme shown in Definition 2.2 makes the k-ary n-tree a delta network [8]: any path starting from a level 0 switch and leading to a given node $p_0, p_1, \ldots, p_{n-1}$ traverses the same sequence of edge labels $(p_0, p_1, \ldots, p_{n-1})$.

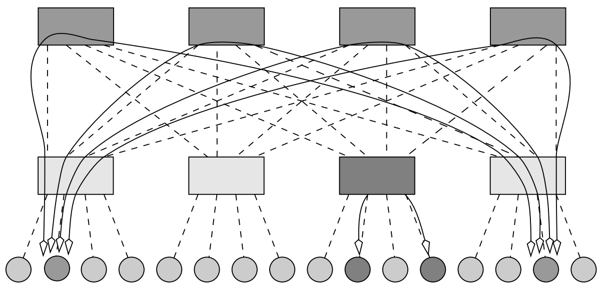

Figure 2. Routing on a 4-ary 2-tree.

As outlined above, minimal routing between a pair nodes on a k-ary n-tree can be slimly accomplished sending the message to one of the nearest common ancestors of both source and destination and from there to the destination. That is, each message experiences two phases, an *ascending* phase to get to a nearest common ancestor, followed by a *descending* phase. An example of 4-ary 2-tree is shown in Figure 2.

3 Relevant details of the network model

This section presents a router model and a simulation environment, that are used in the following sections to analyze the performance of the k-ary n-trees under various traffic loads and flow control strategies.

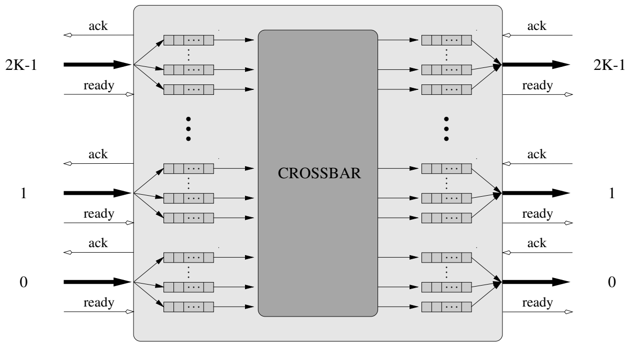

Figure 3. The internal structure of a routing switch.

Figure 3 outlines the internal structure of a $k * k$ 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 $2k$ 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.

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. 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 routing of a single header at a time every $T_{routing}$ cycles. The adaptive routing algorithm studied in this paper, in the ascending phase, picks the channel with the maximum number of free lanes [2].

[2] A free lane is an output lane not bound to any input lane.

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.

The experiments in this paper evaluate 4-ary 4-trees with 256 processing nodes. The number of virtual channels is varied between one, two and four and both input and output lanes have two buffer positions. Each node generates 20-flit packets with exponentially distributed inter arrival times; the destinations are distributed uniformly or according a static communication pattern, as explained in more detail in the following section. The simulator collects performance data only after 2000 cycles, to allow the network to reach steady state and each simulation is halted after 20000 cycles.

Chien in [1] has proposed a cost model to make fair comparisons between routing algorithms. It assumes a 0.8 micron CMOS gate array technology for the implementation of the routing chip. Using this model, as done in [4] for a 256 nodes hypercube, we can see that fat-trees are wire limited when we use up to four virtual channels. For this reason, in our experiments the delays $T_{routing}$ $T_{crossbar}$ and T_{link} are equalized to a single clock cycle and the clock cycle is the same for the three flow control strategies.

4 Message generation

In our model each node generates packets independently according to an exponential distribution and the destinations are chosen according to the following traffic patterns. To describe the patterns, let each node $p_0, p_1, \ldots, p_{n-1}$ also be labeled with a number in base k resulting from the concatenation of the p_i. The binary representation of $p_0p_1 \ldots p_{n-1}$ is $a_0a_1 \ldots a_{(n \log_2 k)-1}$[3]. Also, let $\overline{0} = 1$, $\overline{1} = 0$ and $m = \frac{n}{2} \log_2 k$.

- *Uniform traffic*. Destinations are chosen at random with equal probability between the processing nodes.
- *Complement traffic*. Each node sends only to the destination given by $\overline{a_0a_1 \ldots a_{(n \log_2 k)-1}}$.
- *Bit reversal*. Each node sends only to the destination given by $a_{(n \log_2 k)-1}a_{(n \log_2 k)-2} \ldots a_0$.
- *Transpose*. Each node sends only to the destination given by $a_m a_{m+1} \ldots a_{2m-1}a_0a_1 \ldots a_{m-1}$.

These traffic patterns illustrate different features. The uniform one is a standard benchmark used in network routing studies. This generation pattern can be considered representative of well-balanced shared memory computations. In the complement traffic all the packets cross the bisection of the network and traverse a path whose length is the diameter.

[3] We will assume that k is a power of two and n is even.

5 Experimental results

The performance of an interconnection network under dynamic load is usually assessed by two quantitative parameters, the *accepted bandwidth* or *throughput* and the *latency*.

Accepted bandwidth is defined as the sustained data delivery rate given some offered bandwidth at the network input. Two important characteristics are the saturation point and the sustained rate after saturation. Saturation is defined as the minimum offered bandwidth where the accepted bandwidth is lower than the global packet creation rate at the source nodes. It is worth noting that, before saturation, offered and accepted bandwidth are the same. The behavior above saturation is important because the network and/or the routing algorithm can become unstable, leading to a sharp performance degradation. We usually expect the accepted bandwidth to remain stable after saturation, both in the presence of bursty applications that require peak performance for a short period of time and applications that operate after saturation in normal conditions, e.g. when executing a global permutation pattern.

The *network* latency is the average delay spent by a packet in the network, from the insertion of the header flit in the injection lane till the reception of the tail flit at the destination. It does not include the source queuing delay. The *end-to-end latency* rises to infinity above saturation and is impossible to gain any information in this case. For this reason, the network latency is often preferred to analyze the network performance.

The experimental results of each traffic pattern are presented according to the Chaos Normal Form[4] (CNF). The CNF uses two graphs, one to display the accepted bandwidth and the other to display the network latency. In both graphs the x-axis corresponds to the offered bandwidth normalized with the unidirectional bandwidth of the links connecting the processing nodes to the network switches. This makes the analysis independent from the link bandwidth and the flit size. Also, it is worth noting that k-ary n-trees are not bisection-bandwidth limited as the k-ary n-cubes, whose CNF is normalized on the bisection bandwidth, the upper bound on the throughput for the uniform traffic.

[4] See http://www.cs.washington.edu/research/projects/lis/chaos/www/presentation.html for more details on the presentation of simulation results of network routing studies.

5.1 Uniform traffic

Under uniform traffic the adaptive routing algorithm saturates at 36% of the capacity with one virtual channel, 55% with two virtual channels and 72% with four virtual channels, as shown in Figure 4 a). In all cases the post saturation behavior is stable, with a constant throughput for any offered bandwidth.

These results confirm the importance of the flow control strategy. Wormhole routed networks do not achieve optimal throughput, due to blocking problems. When a packet is stopped at an intermediate switch on the descending phase, several links on the path from the node/switch where the tail flit is stored to the current switch are blocked. Other packets could profitably use these links. In fact, the use of four virtual channels doubles the accepted bandwidth, reaching a considerable 72%. This comes at a price. The condivision of the links between two or more packets slightly increases the network latency for moderate loads. In Figure 4 b) we can see that when the offered load is 20% of the capacity, with one virtual channel the network latency is 45 cycles and 49 cycles with 2 and four virtual channels. When the load is increased at 30% of the capacity, getting closer to the saturation point of the single virtual channel, the use of more virtual channels pays in term of network latency too: in this case we have 62, 56 and 58 cycles, respectively for one, two and four virtual channels.

5.2 Complement traffic

The CNF of the complement traffic shows a surprising behavior, at least at first glance. As can be seen in Figure 4 c), the saturation point is at about 97% of the capacity for all flow control strategies. This permutation pattern doesn't create any congestion in the descending phase. The use of more than a virtual channel is counterproductive in terms of network latency (Figure 4 d): this is mainly due to the link multiplexing, that increases the tail latency. At steady state, there are as many packets in progress as the number of virtual channels in each link. The network latency with one virtual channel remains stable until the offered load is 70% of the capacity and experiences a minor increase of the head latency after this point, which remains under 30 cycles. With two virtual channels the head latency has a similar behavior, while the tail latency converges to the upper bound after 70% of the capacity. The network latency with two and four virtual channels is mainly influenced by the tail latency.

The complement traffic belongs to a wide class of permutations that map a k-ary n-tree into itself. These permutations do not generate any congestion on the descending phase and are called *congestion-free* [6].

5.3 Bit reversal and transpose traffic

Bit reversal and transpose permutations are often generated by numerical programs and are considered as an interesting benchmark for the interconnection network and the routing algorithm. These permutations have a similar distribution of the destinations in terms of distance. It can be easily noted, looking at the numerical representation shown in section 4, that in both cases there are $k^{n/2}$ nodes at distance 0 (that is source and destination are on the same node) and $(k-1)k^{n/2+i-1}$ nodes at distance $n+2i$, for all $i \in \{1, \ldots, n/2\}$. The average distance d_m traversed by a packet is given by

$$d_m = \frac{k-1}{k^{n/2}} \sum_{i=1}^{n/2} (n+2i)k^i. \qquad (1)$$

For a 4-ary 4-tree $d_m = 7.125$, which is very close to the network diameter. The performance results of these communication patterns are very similar. From the CNF of the bit reversal shown in Figure 4 e) we can see that the saturation points are at 39%, 60% and 78% of the capacity for one, two and four virtual channels. An analogous behavior for the transpose can be seen in Figure 4 g), the only difference being the saturation point with one virtual channel at 38% of the capacity. After saturation, there is a linear increase of the accepted bandwidth, because there are $k^{n/2}$ nodes sending packets to themselves.

6 Discussion

The congestion-free communication patterns are an important characteristic of the k-ary n-trees. They can be routed reaching optimal performance with a simple routing algorithm and flow control strategy. They are analogous to local communication in direct topologies, as the k-ary n-cubes. The results obtained on the complement traffic generalize to the whole class of congestion-free patterns and are expected to scale with the number of nodes with an accepted bandwidth that approximates the network capacity. Message latency is only influenced by the flow control overhead and can be deterministically estimated with tight upper bounds. Using a sophisticated flow control strategy with several virtual channels is of little help in this case, because increases the network latency.

The remaining communication patterns, uniform, bit reversal and transpose, generate congestion in the descending phase and are very sensitive to the flow control strategy. As shown in Table 6, they all saturates at about $35-40\%$ of the capacity with one virtual channel, $55-60\%$ with two virtual channels and around 75% with four virtual channels. From these results we can argue that the expected performance of different communication patterns is mainly influenced by

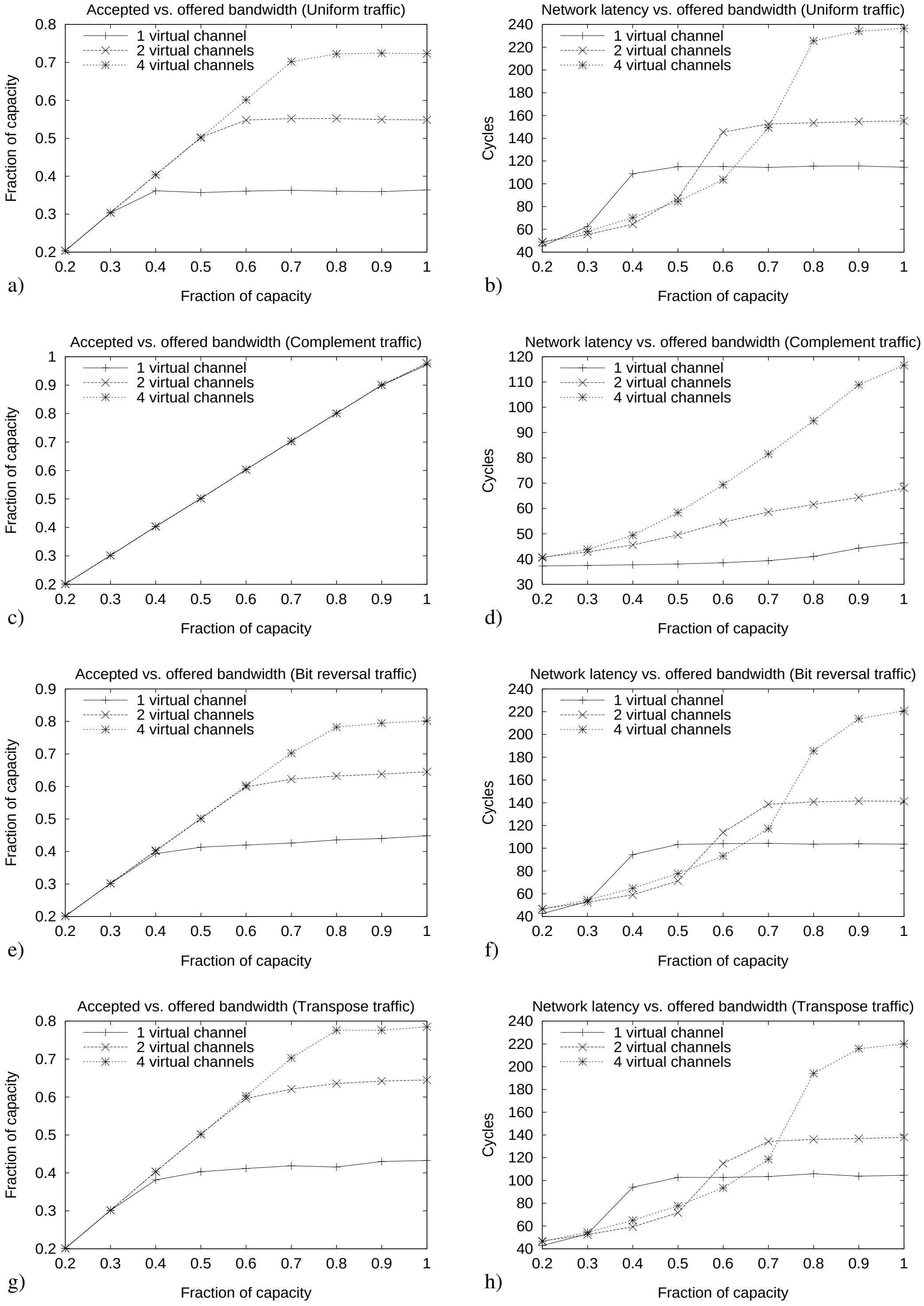

Figure 4. The Chaos Normal Form (CNF) for the uniform traffic (a-b), complement traffic (c-d), bit reversal traffic (e-f) and transpose traffic (g-h).

Traffic pattern	*1 vc*	*2 vc*	*4 vc*
Uniform	36	55	72
Complement	97	97	97
Bit reversal	39	60	78
Transpose	38	60	78

Table 1. Saturation points of the communication patterns.

the flow control strategy. Also, in all these cases, switching from one to four virtual channels doubles the accepted bandwidth. The performance of these patterns is not scalable. In fact, in a 4-ary 3-tree, the saturation points are about 5% above those shown in Table 6 and in a 4-ary 5-tree there is a further decrease of the same amount.

These results provide clear guidelines both for the compilation of parallel programs and for the implementation of the interconnection network. Congestion-free patterns are a powerful tool to reach optimal performance and guaranteed scalability. An important goal for a compiler is to factorize any given application in a collection of congestion-free communication patterns, when possible. In the remaining cases, virtual channels are needed to exploit the network bandwidth and to avoid the blocking problems of wormhole.

7 Conclusion and future work

In this paper we have presented a parametric family of regular topologies, the k-ary n-trees. k-ary n-trees are a particular type of fat-trees, built using processing nodes and constant arity switches interconnected in a butterfly-like topology. Adaptive routing on k-ary n-trees can be easily accomplished in two phases. In the ascending phase, each message reaches one the nearest common ancestors of source and destination and then begins the descending phase following the unique downward path to the destination. The adaptive algorithm has been extensively analyzed on a 4-ary 4-tree with 256 nodes under several traffic patterns, representative of shared memory computations and numerical algorithms. We have compared some variants of the routing algorithm using one, two and four virtual channels and the experimental results have outlined two main classes of communication patterns. The complement traffic, representative of the class of the congestion-free patterns, has reached an optimal performance, with a saturation point at 97% of the capacity for all flow control strategies. The remaining traffic patterns, uniform, bit reversal and transpose, are very sensitive to the flow control strategy.

References

[1] A. A. Chien. A Cost and Speed Model for k-ary n-cube Wormhole Routers. In *Hot Interconnects '93*, Palo Alto, California, August 1993.

[2] W. Dally. Network and Processor Architecture for Message-Driven Computers. In R. Suaja and G. Birthwhistle, editors, *VLSI and Parallel Computers*, chapter 3, pages 140–222. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1991.

[3] A. DeHon. Fat-Tree Routing for Transit. Technical Report 1224, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, February 1990.

[4] J. Duato and P. López. Performance Evaluation of Adaptive Routing Algorithms for k-ary n-cubes. In K. Bolding and L. Snyder, editors, *First International Workshop, PCRCW'94*, volume 853 of *LNCS*, pages 45–59, Seattle, Washington, USA, May 1994.

[5] R. I. Greenberg and C. E. Leiserson. Randomized Routing on Fat-Trees. *Advances in Computing Research*, 5:345–374, 1989.

[6] S. Heller. Congestion-Free Routing on the CM-5 Data Router. In K. Bolding and L. Snyder, editors, *First International Workshop, PCRCW'94*, volume 853 of *LNCS*, pages 176–184, Seattle, Washington, USA, May 1994.

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

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

[9] C. E. Leiserson. Fat-Trees: Universal Networks for Hardware Efficient Supercomputing. *IEEE Transactions on Computers*, C-34(10):892–901, October 1985.

[10] 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*, pages 272–285, June 1992.

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

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

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

[14] Meiko World Incorporated. *Computing Surface 2 reference manuals*, preliminary edition, 1993.

[15] 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*, pages 115–128, Cancun, Mexico, April 1994.

[16] M. Valerio, L. Moser, and P. Melliar-Smith. Recursively Scalable Fat-Trees as Interconnection Networks. In *Proceedings of the Thirteenth IEEE International Phoenix Conference on Computers and Communications*, pages 40–46, Phoenix, AZ, April 1994.