ELSEVIER Simulation Practice and Theory 6 (1998) 369–385

SIMULATION PRACTICE AND THEORY

Performance evaluation of message passing strategies and routing policies in multicomputers

M. Colajanni [a,*], A. Dell'Arte [b], B. Ciciani [b]

[a] *Dipartimento di Informatica, Sistemi e Produzione, Università di Roma 'Tor Vergata', Via della Ricerca Scientifica, Roma I-00133, Italy*

[b] *Dipartimento di Informatica e Sistemistica, Università di Roma 'La Sapienza', Via Salaria 113, Roma I-00198, Italy*

Received 1 July 1996; revised 1 May 1997

Abstract

This paper describes a discrete-event simulator designed for the analysis of communication switching techniques adopted in distributed memory multicomputers with asynchronous direct links. The motivation comes from the observation that, even if a wide set of routing strategies have been proposed, an extensive comparison analysis among routing strategies working under different workload characteristics is still lacking in the literature. For these purposes, we have designed and implemented a modular simulator, namely IntNetSim, that solves the models of a wide combination of message passing techniques, routing policies, link conflict resolution strategies, and traffic conditions. For each class of instances, a large variety of simulation runs permits us to select the best performance technique that reduces message transmission time and/or retards saturation of the network under definite workload conditions. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Message passing strategies; Interconnection networks; Distributed memory multiprocessors; Performance evaluation

1. Introduction

The performance of any parallel application on distributed memory multiprocessors strictly depends on the choice of the task granularity and the communication pattern. The interconnection network and the adopted message passing strategy play a fundamental role in determining the best choice for both these issues. In its turn, the performance of the communication support strictly depends on the characteristics of the application workload and the communication pattern among parallel tasks. This paper intends to explore the mutual dependence existing between performance of parallel applications and performance of communication support. Since this

* Corresponding author. Fax: +39-6-85300849; e-mail: ciciani@dis.uniroma1.it.

PII: S0928-4869(97)00022-0

correlation is not always taken into consideration, it is important to evaluate the performance indices of the communication network under different traffic conditions and network characteristics. In particular we focus on distributed memory machines based on asynchronous direct interconnection networks. We characterize the message passing algorithms on the basis of three attributes: *communication switching technique*, *routing policy*, and *link conflict resolution strategy*. Four main switching techniques have been adopted for these architectures: *store-and-forward*, *circuit-switching*, *virtual-cut-through*, and *wormhole routing*. In addition, each of them may follow a *deterministic* or *adaptive* routing policy, and select a particular strategy to solve link conflicts.

For these kinds of problems, simulation is the only way to obtain results when other approaches fail. The complexity of the systems here considered eludes analytical solutions unless one makes unrealistic simplifying assumptions that in any case lead to approximate results. The main difficulty resides in the fact that asynchronous communication networks are modelled by passive resources that prevent a product-form solution of the queuing networks including them. Conversely, the simulation has been demonstrated to be a very useful tool for studying many real situations thus avoiding the necessity of waiting for operational experience. Even if a large quantity of results were obtained through simulation [1,5,10,14,15], most of these papers either focus on one/two communication switching techniques or on a particular link conflict resolution strategy without carrying out a complete comparison analysis. A wider spectrum of routing strategies has been examined by Grunwald and Reed [7]. However, they do not cover several message passing techniques such as wormhole, adaptive store-and-forward, and adaptive virtual-cut-through.

The main goal of this paper is to carry out an exhaustive comparison among the routing techniques applied to several architectures and parallel applications. In particular, we take into account deterministic and adaptive routing policies under various link conflict resolution strategies such as blocking, drop one link, drop all links. We apply all of them by varying application features and interconnection network characteristics such as topology, dimension, router, link. An accurate discrete-event simulator, called INTNETSIM, has been developed for the purpose of carrying out an exhaustive performance analysis of all these switching techniques. The simulator has been implemented in the Simula language on Unix based platforms. Its design and implementation guarantee a considerable modularity and portability. A wide spectrum of applications and computational platforms can be characterized through the set of parameters passed as input to the simulator.

The paper is organized as follows. In Section 2, we introduce the main features of the communication support. In Section 3, we present the architectural design of the simulator. In Section 4, we describe the analysis for validating the simulator. In Section 5, we discuss some performance results obtained by INTNETSIM, while in Section 6 we give some final remarks and outline feasible directions for extending this simulator.

2. Message passing algorithms

Before passing to analyze the routing algorithms, let us give some definitions adopted throughout this paper:

• *switching technique* – the main algorithm adopted to move a message from a source to a destination node;

• *routing policy* – it specifies whether the path from source to destination node is unique (*deterministic policy*) or, in case of link conflict, the message can look for alternative routes (*adaptive policy*);

• *link-conflict resolution strategy* – it determines the actions that a message transmission has to carry out in the case of conflict for a link; some feasible strategies are: waiting in the node, retrying to build the entire path from the source node, changing the path.

Remarkable progress has been made in the development of message passing techniques since the appearance of the first generation of multicomputers such as nCUBE-1, and iPSC/1. These machines adopted *store-and-forward* (SF) switching techniques that collect the entire message or packet at each intermediate node of the path before requesting the next link.

A first reduction of the data transmission time was obtained by the *virtual-cut-through* (VCT) technique proposed by Kermani and Kleinrock [9]. Unlike the SF strategy, VCT divides the message into small parts, namely *flits*, and implements a pipeline switching technique. Once having obtained the first link, and before the message is entirely received at the adjacent node, the second link is required and, if obtained, the flits are sent out to the following node. This strategy has the advantage of reducing message buffering to the only instances in which a link is not available. Nevertheless, VCT was adopted only by prototype machines.

The second generation multicomputers, such as Intel iPSC/2 and iPSC/860, preferred *circuit-switching* (CS) techniques that avoid the large node buffers required by SF and VCT. In this case, the message header has to establish the entire path from the source to the destination node before data can be transmitted. Since each communication requires and holds all links of its path, link contention tends to increase, thus significantly affecting performance in case of heavy traffic.

Last generation multicomputers, such as Intel Paragon, nCUBE-2 and nCUBE-3, adopted *wormhole* (WH) routing techniques that are finely described in [13]. In a contention-free network, WH has the same pipeline behavior as VCT. However, in case of conflict, WH does not gather all the flits in a node, but blocks them in the *flit buffers* of the built path.

Any main switching technique builds the communication path from the source to the destination node through a step-by-step process, in which for each step the additional link is assigned by the local routing controller only after having verified that it is free. In case of link conflict, there are several possibilities that depend on the adopted *routing policy* and *link-conflict resolution strategy*.

The routing policies may be either *deterministic* or *adaptive*. All of the early commercial multicomputers used deterministic policies in which the route between sender and receiver nodes is fixed. Even if their implementation is simplified, they do not have the same ability to respond to dynamic network conditions such as the adaptive policies that can establish alternative routes and can preserve communications even in the presence of some faulty paths [6,9,11,16].

The *link-conflict resolution strategy* (for VCT, CS, WH strategies) concerns the action that has to be carried out on the already obtained links when a conflict occurs:

- The *blocking* (or *wait*) strategy leaves the link request pending at the node without releasing any link. This strategy is dependable and economic to implement. However, it does not show a good utilization of resources, because it leaves idle some allocated links.
- The *drop-all* (or *discarding*) strategy releases all the acquired links and retries to build the path from the sender node after a back-off time. Even if this strategy does not leave idle any link, the resulting message re-transmission may be burdensome.
- The *drop-one* (or *backtracking*) strategy releases the last acquired link and asks for a different link from the previous node. If no other link is available, it continues recursively to release another link and looks for an alternative path. Even if this strategy offers more flexibility, its implementation is more expensive and, if not very accurate algorithms are adopted, a communication request may enter a livelock cycle.
- The *virtual channel* strategy multiplexes the link utilization among all the communications in conflict. Throughout this paper we do not consider this possibility because a comparison among the various strategies, most of them proposed in combination with WH, would deserve a quite different study exclusively devoted to this purpose.

It should be observed that the *drop-one* strategy is intrinsically adaptive, while the *drop-all* may be combined both with a deterministic policy (drop-all after the first conflict) and an adaptive policy (drop-all after having verified that no link is available).

All the switching techniques both in *deterministic* or in *adaptive* form can use the *blocking* conflict resolution strategy. However, only CS and WH allow a wider spectrum of link conflict resolution strategies. In particular, CS may use blocking, drop-all and drop-one strategies, while WH has been implemented both in blocking and in drop-all form. Even if this latter has been adopted by the BBN Butterfly interconnection network, it is rarely used because of its unstable message transmission rate [8]. The drop-all combined with WH risks loosing packets. For this reason the best combination is with CS that moves only the header flit instead of the entire message.

Table 1 sums up the feasible message passing techniques that can be modelled by IntNetSim. The main goal of this paper is to compare all of these techniques under different workload conditions. In particular, our analysis will focus on

- parallel application features denoted by various message generation rates, selection of the destination nodes (uniform distribution, random, special sets), trace files;
- interconnection network characteristics (topology, dimension, router delays, link bandwidth, uni- or bi-directional channels);
- switching techniques (SF, VCT, CS, WH), routing policies (deterministic,

Table 1
A summary of analyzed message passing algorithms

Switching technique	Routing policy	Link-conflict resolution strategy
SF	deterministic	blocking and releasing
SF	adaptive	blocking and releasing
VCT	deterministic	blocking and releasing
VCT	adaptive	blocking and releasing
CS	deterministic	blocking
CS	deterministic	drop-all
CS	adaptive	drop-one
CS	adaptive	drop-all
WH	deterministic	blocking
WH	deterministic	drop-all
WH	adaptive	blocking

adaptive), and link conflict resolution strategies (blocking, drop-one, drop-all) in case of VCT, CS and WH.

In the present version of the simulator, we consider only strategies that build *optimal* paths that is, strategies that use the minimum number of links from source to destination node. Moreover, in this paper we are not interested in addressing fault-tolerant issues.

3. The simulator

The simulation model is discrete event driven. It has been implemented in the Simula language on Unix based platforms. The choices made in the design and implementation guarantee an appreciable modularity of the simulator that is organized on a three layer-basis:

- interconnection network architecture;
- message passing routing strategies;
- parallel computations and asynchronous events that are hardware independent.

The entire set of parameters passed as input to the IntNetSim simulator characterizes a wide spectrum of applications and computational platforms that we outline in the following sections.

3.1. Interconnection network level

IntNetSim can model any k-ary n-cube topology, and in particular commercial interconnection networks such as hypercube, 2-dimensional (2D) and 3-dimensional (3D) mesh, and torus. Each node of the system consists of three components: a processing element, a routing controller, and a crossbar switching network.

The processing elements are represented in the simulator by the average time to execute one floating-point operation, while each router carries out the following main functions: testing free-link, connecting a link to the path, releasing a link. These functions, depending on the choice of the user, can be denoted by deterministic or probabilistic execution times. The hypothesis of a DMA controller per each dimension guarantees simultaneous communications for each router through different links. The Simula *process class* construct has been used for DMA controllers and routers.

We have represented the interconnection network as an adjacency matrix, that is a bi-dimensional array with $n \times \mathrm{ch}$ elements, where n is the number of nodes and ch the number of links for each node. A non-zero $[i, j]$ entry of this matrix denotes the node i directly linked to node j. The values of these entries are determined through a mapping function that depends on the network topology chosen by the user.

In addition to topology and dimension, IntNetSim may model many other features of the interconnection network, such as link bandwidth, one/two directional channels, router at different levels of detail (no delay, constant delay, queuing server), node buffer of given dimensions starting from null. Although the present version of the simulator does not provide the possibility of modelling other policies such as the *virtual channel* strategy, we are working to release this limitation.

3.2. Message routing level

At present, we consider only strategies that build optimal paths, that is the number of links between source and destination node is minimum. However, IntNetSim can adopt any feasible combination of the switching techniques, routing policies, and link-conflict resolution strategies listed in Table 1. It should be noted that SF, VCT and WH strategies require an additional buffer link for each node. In the case of WH, the dimension of this buffer is limited because it stores flits and not entire messages. However, adaptive WH routing policies such as *p-cube* avoid deadlock risks by using double links.

In order to improve the modularity of the simulator, each function is implemented as a procedure included in the process class that implements the most important system activities. For example, the link connection process contains several subroutines that depends on the chosen communication switching technique and link-conflict resolution strategy. In particular, the *drop* strategies used in combination with CS require an additional process that simulates the waiting back-off time. In addition, the *backtracking* strategy uses a more complex process that has to manage the drop-one and retry function of the header message.

The potential conflicts on the same link are solved by a decision algorithm that is implemented by an *arbiter* process. The execution of this process requires a completion time that may be modelled through a generic distribution.

The messages, the router elements and some hand-shaking signal are implemented by means of the *link class* construct. This allows an efficient management of the queuing operations relative to routers, and requests for transmission of messages (or parts of it) to DMA controllers.

3.3. Application level

The application level aims to represent a parallel program running on the previously defined system. From the simulator point of view, a parallel application is an alternate sequence of computations and message generations. IntNetSim provides two possibilities for denoting this kind of input: *synthetic workload* and *trace workload.*

The synthetic workload does not correspond to an actual application, but it is useful to evaluate the scalability of the performance of a network under particular conditions. To this purpose, IntNetSim may model uniform traffic and parameterized workload with different message lengths. The message generation from a node is simulated through the activation of a *source* process. The inter-arrival time between two consecutive messages is a random variable given by a distribution chosen among constant, uniform, k-Erlang, and exponential functions. Each generated message has two attributes (*destination node* and *length*) that can be selected from a set of functions and distributions. For example, the destination node can be identified through several attributes such as random, uniform or decay distribution, bit complement, or bit-reversal.

In addition to the synthetic workload, IntNetSim is able to represent actual computations in which the alternation of computation and communication phases derives from a trace file. In this instance, the source processes do not generate random messages on the basis of some distribution, but they take normalized input values from a file. Each source has to be provided with a different trace file. However, no tool is yet provided to automatically obtain the trace file relative to an actual application, even if this problem is under study.

Another recent facility provided by IntNetSim is the possibility of representing not only node-to-node message transmissions, but also multicast and broadcast communications. In such a way, we are able to represent with more accuracy a wide set of actual parallel applications.

3.4. Output analysis

Several output parameters can be chosen to evaluate performance of message passing strategies. Among them, mean message latency time, mean service time of router, mean length path, probability of link conflict.

The output analysis adopts the independent replications method where the Jackknife estimator was used to produce confidence intervals at 95% for the throughput, mean queue length and mean response time. However, other parameters, such as machine utilization rate and waiting time in queue for each task, could be also evaluated.

The obtained confidence intervals generally have relative widths of 5–8%, but models with different parameters had requested different number of replications. For example, higher network dimensions and message traffic rates require more simulation runs. Besides, we observed higher costs of CPU time for a single simulation run in the instance of congested systems and, under the same traffic conditions, for the WH model more than for the CS model. This is mainly due to the greater functional complexity of the former system.

4. Validation of the simulator

Two kinds of validation were carried out that is, mathematical and experimental. We first validated IntNetSim against an analytical Markovian model explicitly solved for this purpose. To obtain a model which was analytically tractable we considered interconnection networks with symmetric topology such as the hypercube and torus with same number of nodes for each dimension. Moreover, we assumed unlimited memory capacity in each node; communication requests arising from each node independently and identically distributed in accordance with a Poisson process. In addition, since most of the performance parameters depend on the probability that a request for a link may conflict with another communication, an exact solution of the analytical model required as the main assumption that no conflicts occurred in the interconnection network. We obtained a wide set of results as a function of different communication request rates, data transfer sizes, and network dimensions. In all the instances, the analytical results for message communication time were demonstrated to fall inside the confidence intervals provided by IntNetSim.

Furthermore, we considered even the possibility of link conflicts for the CS and WH strategy. However, since in these instances the model has not an exact mathematical solution, we had to refer to the approximate technique proposed in [2,3], respectively. The difficulty of evaluating the probability of link conflict for these models is mainly due to a mutual dependence. The link utilization depends on the duration of the communication phase and, in its turn, the duration of the phase for acquiring all the links depends on the probability of contention/abort and on the routing controller overhead. An iterative solution approach was adopted to capture this feedback effect for CS [3], while we were able to achieve closed formulas for the model of deterministic WH [2]. The same iterative technique was applied to solve the analytical models for the following link conflict resolution strategies: deterministic blocking, deterministic drop-all, adaptive drop-all. In order to render these models tractable we had to introduce some further assumptions. In particular, we assumed that the probability of link conflict is independent of the number of links already obtained. In addition, for the deterministic blocking strategy, we assumed that a link conflict implies that no other requests are waiting for the same link, and the waiting times due to contention are independent variables with the same mean. For the adaptive strategy, we assumed that the probability of link conflict is independent of the number of aborts which occurred.

We observed that the simulation results matched the analytical results very well. In most experiments, the analytical values fall between the bounds of the confidence interval of the mean latency time estimated by the simulator, while some discrepancy exists only for very congested networks. However, all let us believe that this difference is not attributable to the simulator but is entirely due to the inaccuracy of the analytical model that is not able to adequately represent the system for high message generation rates.

To demonstrate that IntNetSim provides accurate predictions on the performance of applications running on multicomputers, we also carried out some simple experimental validations. These were based on simulations that use actual workload derived from traces collected on operational systems. In particular we carry out experiments following the single program multiple data (SPMD) paradigm which is the most widely adopted programming model for a large class of problems. By following this model, the programmer generates a single program that every node executes on a different portion of the data domain. The different evaluation of some predicates occurring in conditional statements allows each node to take different paths through the program and to operate on a different section of the data set. In particular, we modelled the kernels relative to two well known parallel algorithms for linear algebra: the *LU factorization* of a dense matrix decomposed by rows, and the multiply-roll algorithm for *matrix multiplication* [4].

The experiments were carried out on an Intel iPSC/2 with 16 nodes connected through a hypercube topology with full duplex channels. The system parameters of this machine have been settled to those reported in [12]: $t_L = 210$ μs, $t_C = 0.58$ μs/byte, $t_S = 0.5$ μs for the message latency, data transmission and floating point operation time, respectively.

The switching technique adopted by Intel iPSC/2 is CS with deterministic routing policy, namely *e-cube* [8]. The routers allow messages to route through intermediate nodes without interrupting processes executed on those nodes. The interval between two successive communications were settled to the constant values (dependent on the size of the matrix) for the matrix multiplication algorithm, while for the LU factorization these intervals were reduced after the completion of each main iteration step. In all of the test cases for matrix sizes equal to 500, 800 and 1000, the experimental and the simulation results were acceptably close that is, typically within 10% relative error.

5. Simulation results

In this Section we analyze and compare the performance of some of the switching techniques, routing policies and link conflict resolution strategies as a function of three parameters: traffic distribution, channel type, and network dimension. We tested a large variety of models. Since the set of variable parameters is very high and it is not possible to discuss here all the performance results obtainable through INTNETSIM, we fixed the architecture and message features and let the message passing strategies, that represent the main focus of this paper, vary. In relation to the architecture, we analyze the hypercube that has been adopted in several multicomputers such as Cosmic Cube, Connection Machine, nCUBE, iPSC/2, and iPSC/860. The network dimension is variable from 6 to 10, the channel type is usually chosen as single link, while for the adaptive WH we adopt a double link connection as well.

The communication router is modelled as a queuing server. The message generation rate is chosen as a Poisson distribution, while the message lengths are modelled through uniform distributions. The performance comparisons are based on *mean message latency* that includes path-set-up time, transfer message time, and link release time.

We distinguish uniform traffic that is, destination nodes are selected by uniform distribution, from other traffic conditions such as bit reversal, bit complement, multicast.

5.1. Uniform traffic

In the first experiment, we fix a link conflict resolution strategy (*blocking*), a routing policy (*deterministic*) and verify performance of the four communication switching techniques: SF, VCT, CS, WH. We first analyze the ideal instance of infinite buffers for VCT and SF. Fig. 1 shows that in this case VCT can sustain higher traffic loads than any other strategy. Nevertheless, in the more realistic instance of limited buffers (Fig. 2), both SF and VCT demonstrate a rough growth of the communication time when the traffic increases. Even if these results

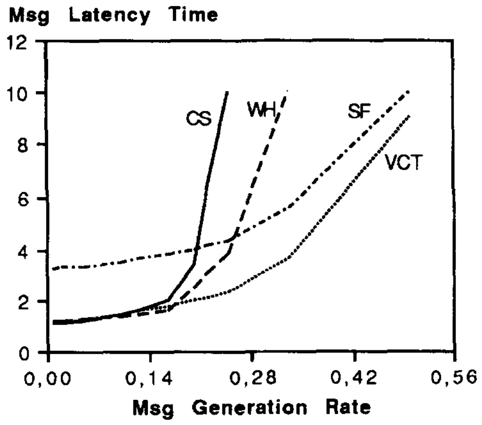

Fig. 1. Four switching techniques in case of deterministic routing (Case I: *infinite buffers*).

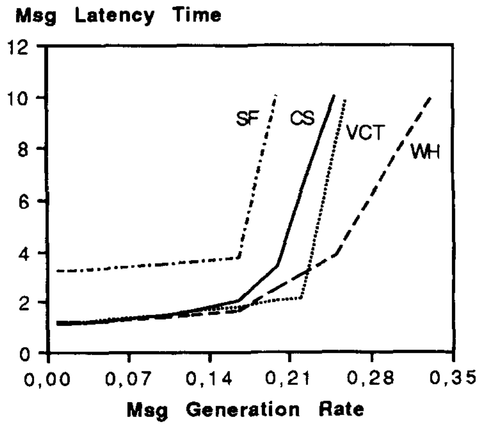

Fig. 2. Four switching techniques in case of deterministic routing (Case II: *finite buffers*).

demonstrate that WH achieves the best performance in actual conditions, VCT with large buffers or for short messages could overcome any technique.

In the second experiment, we fixed a communication switching technique and let the conflict resolution strategies vary for deterministic and adaptive routing. Here, we focus on the CS technique for a system with 64 and 256 nodes in Figs. 3 and 4, respectively. In both instances, the adaptive conflict resolution strategies — especially, in the *drop-one* version — are better than the deterministic policies. Moreover, by comparing the results of the two figures, we can observe that there is not a big difference among the deterministic policies in the two systems. On the other hand, when the system has more nodes and links, the adaptive policy is able to support much higher traffic loads. In fact, the number of alternative paths is a combinatorial number of the system dimension. However, this property of the adaptive policies is

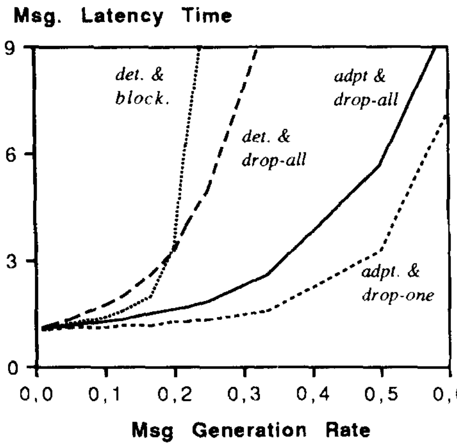

Fig. 3. CS techniques for a 64 node system.

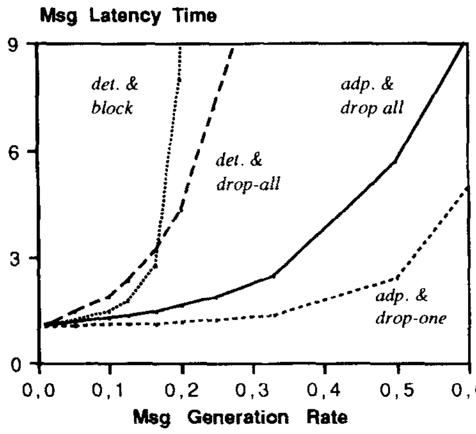

Fig. 4. CS techniques for a 256 node system.

valid only for hypercube topologies, while it does not hold when we augment the size of mesh/torus architectures without increasing the dimension for example from two to three.

For SF and VCT we obtained results similar to CS. More surprisingly, the adaptive WH routing has shown performance slightly worse than deterministic routing for a 64 node system. The reason could be found in two factors: the limited hypercube dimension does not allow many alternative routes; the uniform distribution for message destination prefers regular paths.

Having verified that in general the adaptive routing policies achieve better performance than deterministic strategies, we compared adaptive policies in the case of CS, SF, and VCT switching techniques; these last two with finite buffer. Even if Fig. 5, which refers to a system with 64 nodes, shows that *drop-one* CS outperforms all the other policies, it should be noted that this algorithm is more difficult to implement and requires more expensive routers.

In the last experiment we have fixed the message transmission rate and let its dimension vary. The communication time is evaluated as a function of the message-length/link-bandwidth ratio that varies from 0.2 to 2.0. Fig. 6 shows that WH always performs best, while VCT behaves better than CS for short messages. Conversely, CS can beat the other strategies for very large messages if compared to the available communication bandwidth of the system.

5.2. Nonuniform traffic

In this section we present the results related to a system subject to nonuniform traffic workloads, that tends to stress all the message-passing strategies. All the results are related to a hypercube with 256 nodes.

First, we consider the *bit-reversal* traffic. In such a case, each node generates a message towards the node that has the address which is the bit-reversal of the source address.

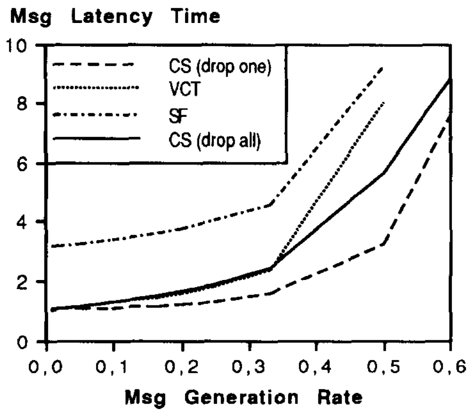

Fig. 5. Adaptive routing policies.

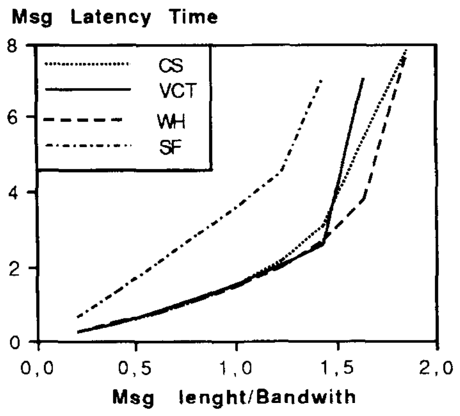

Fig. 6. Switching techniques with deterministic routing policy.

We compare the CS switching technique with the deterministic and adaptive policies. Fig. 7 shows that the drop-all adaptive strategy always performs best, while the blocking deterministic policy is better than the drop-all deterministic. This latter result modifies the conclusions achieved for uniform traffic conditions (see Fig. 4), where for message generation rates higher than 0.16 the drop-all deterministic strategies performed better than the blocking policy.

The WH switching techniques with the blocking link-conflict resolution strategy have been evaluated for a system with bi-directional links. Fig. 8 compares deterministic and adaptive policies in the instance of uniform and bit-reversal traffic. In the former case, the performance of the adaptive policy is very close to that of the deterministic policy and becomes worse only for very high traffic workloads. This result confirms that uniform traffic does not favor the search for alternative paths even in the case of system dimensions higher than 64 nodes. Conversely, the

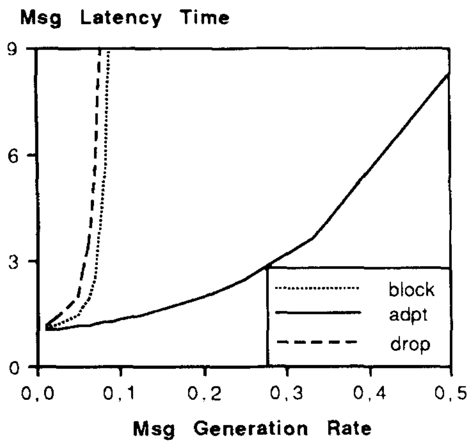

Fig. 7. CS techniques for bit-reversal traffic.

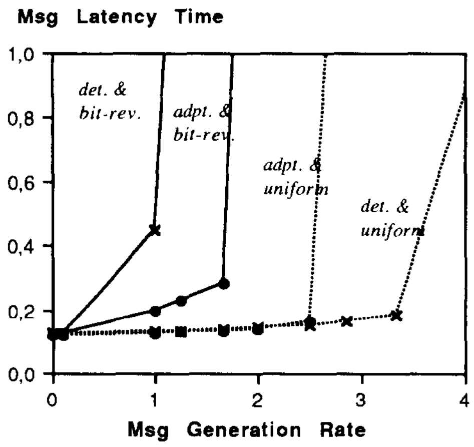

Fig. 8. WH techniques with deterministic and adaptive routing policy for uniform and bit-reversal traffic.

circumstances change in the case of bit-reversal traffic. Now, it is the adaptive policy which performs best and sustains almost double the quantity of traffic of the deterministic policy.

Other kinds of traffic are those generated by data transmissions from one node to multiple destinations, namely *multicast routing*. These communications are pretty common in parallel applications. For example, we used them in the algorithms that we adopted for the experimental validation of the simulator (see Section 4).

If multicast communications are not carefully used by the programmer, they tend to easily saturate the network. In our experiments we consider multicast from one node to their neighbors. In particular, we show the performance of the CS technique for various policies and link-conflict resolution strategies in the instance of multicast from one node towards three neighbors in a system with uni-directional channels. Fig. 9 shows that the behavior of the three strategies (blocking deterministic, drop-all deterministic and adaptive) do not differ substantially. The adaptive strategy, which for low message generation rates performs worse than the blocking deterministic, tends to sustain higher traffic workloads. Moreover, it should be noted that in all the cases the network saturates for very low traffic generation rates, especially if compared to the results of the experiments shown in Figs. 1–8. For this reason, in the last set of experiments we considered a system with bi-directional channels. Fig. 10 shows that in this case the network can support much higher traffic generation rates principally when the WH technique is adopted. For this kind of system we also consider multicast communications towards higher numbers of nodes than the previous experiments that is, all the 7 neighbors that each node of a 256 node hypercube has. The expected result is confirmed by the simulations because the WH technique behaves considerably better than CS (see Fig. 10).

Our research was directed to finding the message passing strategy with the best performance, in particular aiming at reducing the mean response time for different traffic workloads. The results confirm some intuitive elements and demonstrate other

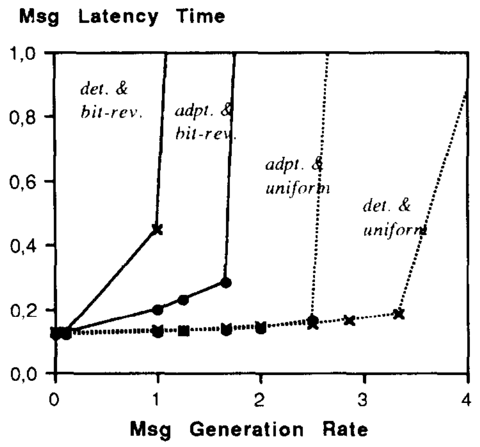

Fig. 9. CS techniques for multicast routing from each node to three neighbors.

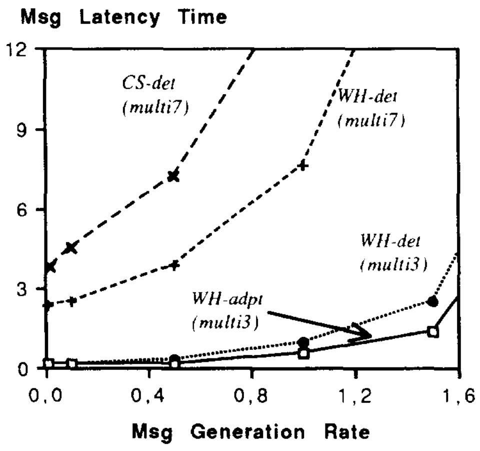

Fig. 10. Deterministic CS, deterministic WH and adaptive WH techniques in networks with bi-directional channels for multicast routing towards two sets of nodes that is, three and seven neighbors.

more interesting factors. There is not a strategy that always guarantees best performances, even if WH is the more promising strategy. The problem is whether or not to combine WH with adaptive strategies. Even if they often seem to perform better than the deterministic policies, some counter-example exists. See for example the WH technique with uniform traffic or other instances with low traffic workloads or network with small numbers of nodes. Moreover, the adaptive WH policies require sophisticated algorithms for avoiding deadlocks and livelocks [13].

If we distinguish the light traffic condition from the heavy traffic condition, our analysis shows that in the former case there are not big differences among WH, CS and VCT, while WH is confirmed to achieve best performance. In the latter case, as expected, *blocking* CS is the first strategy that breaks; while, more surprisingly, VCT with large router buffers or with small packets seems to be the strategy that reduces mean message latency. This leads us to believe that a strategy which combines VCT and WH could result in the best policy.

6. Conclusions

In this paper we have compared the most common communication switching techniques under different system and workload conditions. The performances were evaluated through a purpose designed discrete-event simulator, namely IntNetSim, that has been demonstrated to be very flexible and able to model several architectures, algorithms and traffic conditions. The results and the simulator itself are highly portable. In particular, this analysis presents two interesting aspects. Firstly, the simulation models can be used as a planning evaluation tool in the interconnection network design phase. Secondly, it can be the basis for obtaining the global response time and other general parameters of parallel applications, once the whole structure of the system has been determined.

In the near future, we intend to enrich IntNetSim with other two capabilities that should facilitate its use: a graphical input/output interface, and some automatic tools for extracting *trace workload* input from actual parallel programs. In addition, we are also including other wormhole techniques that have been recently proposed in the literature.

References

[1] R.V. Boppana, S. Chalasani, A comparison of adaptive wormhole routing algorithms, in: Proceedings of the 20th International Symposium on Computer Architecture, San Diego, CA, 1993, pp. 351–360.
[2] B. Ciciani, M. Colajanni, C. Paolucci, An accurate model for the performance analysis of deterministic wormhole routing, in: Proceedings of the IEEE 11th International Parallel Processing Symposium, Geneva, Switzerland, 1997.
[3] M. Colajanni, B. Ciciani, S. Tucci, Performance analysis of circuit-switching interconnection networks with deterministic and adaptive routing, Tech. Report RI.95.15, Università di Roma 'Tor Vergata', Oct. 1995, also submitted to Performance Evaluation.
[4] G.C. Fox, M. Johnson, G. Lyzenga, S.W. Otto, J. Salmon, D. Walker, Solving problems on concurrent processors, vol. 1, General Techniques and Regular Problems, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[5] C.J. Glass, L.M. Ni, The turn model for adaptive routing, in: Proceedings of the ACM 19th Int. Symposium on Computer Architecture, Gold Cost, Australia, 1992, pp. 278–287.
[6] L. Gravano, G.D. Pifarré, P.E. Berman, J.L.C. Sanz, Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks, IEEE Trans. Parallel Distrib. Syst. 5 (12) (1994) 1233–1251.
[7] D.C. Grunwald, D.A. Reed, Networks for parallel processors: Measurements and prognostications, in: Proceedings of the 3rd Conf. on Hypercube Concurrent Computer Applications, New York, NY, 1988, pp. 610–619.
[8] K. Hwang, Advanced System Architectures, Mc Graw Hill, 1994.
[9] P. Kermani, L. Kleinrock, Virtual Cut-Through: A new computer communication swithching technique, Comput. Networks 3 (1979) 267–286.

[10] C.A. Lamanna, W.H. Shaw, A performance study of the hypercube parallel processor architecture, Simulation 53 (3) (1991) 185–196.
[11] D. Linder, J.C. Harden, An adaptive and fault tolerant wormhole routing strategy for *k*-ary *n*-cubes, IEEE Trans. Comput. 40 (1) (1991) 2–12.
[12] O.A. McBryan, An overview of message passing environments, Parallel Comput. 20 (4) (1994) 417–444.
[13] L.M. Ni, P.K. McKinley, A survey of wormhole routing techniques in direct networks, IEEE Comput. 26 (2) (1993) 62–76.
[14] J. Peterson, E. Chow, H. Madan, A high-speed message-driven communication architecture, in: Proceedings of the ACM Int. Conference on Supercomputing, St. Malo, France, 1988.
[15] S. Ramany, D. Eager, The interaction between virtual channel flow control and adaptive routing in wormhole networks, in: Proceedings of the ACM Int. Conf. on Supercomputing, Manchester, U.K., 1994, pp. 136–145.
[16] C.-L. Wu, M. Lee, Performance analysis of multistage interconnection network configurations and operations, IEEE Trans. Comput. 41 (1) (1992) 18–26.