Are independent traffic sources suitable for the evaluation of interconnection networks?

J. Miguel-Alonso[1], F.J. Ridruejo[1], C. Izu[2], J.A. Gregorio[3] and V. Puente[4]

[1]Dep. of Computer Architecture and Technology, University of the Basque Country, 20080 San Sebastian, Spain

{miguel, acbripef}@si.ehu.es

[2]Department of Computer Science, University of Adelaide, SA 5005 Australia

cruz@cs.adelaide.edu.au

[3]Computer Architecture Research Group, Universidad de Cantabria, 39005 Santander, Spain

monaster@unican.es

[4]Computer Sciences Department, University of Wisconsin-Madison, USA

vpuente@cs.wisc.edu

Abstract. Many simulation-based performance studies of interconnection networks are carried out using synthetic workloads under the assumption of independent traffic sources. Although this assumption may be useful for some traffic patterns, it may lead to erroneous conclusions about the usefulness of some design proposals for heavy loads. In this work we show that some traffic patterns generated by independent sources introduce an undesired uneven utilization of resources that privilege some nodes (compared to others), and that affects performance measurements in unexpected ways. The limitation of these privileges, for example using burst-synchronized traffic that forces fast nodes to wait for slower ones, produces more realistic performance results. Just as an example, in this work we show that the utilization of a congestion control mechanisms looks

This work has been done with the support of the Ministerio de Educación y Ciencia, Spain, under grants TIN2004-07440-C02-01, TIN2004-07440-C02-02, and PR2004-0572, and also by the Diputación Foral de Gipuzkoa under grant OF-846/2004.

ineffective when traffic is generated this way. However, if we use other generation process in which traffic sources are coupled, then congestion control shows all its potential. We show that the latter traffic generation method is more realistic and with more coverage than the classic one using an analysis of a relevant set of actual applications.

1 Introduction

Performance studies of interconnection networks may be carried out via a choice of methods, ranging from the construction and measurement of hardware prototypes, to the utilization of overly simplified simulations. During the first stages of a new interconnection project, a fast simulation environment is critical, because it allows researchers to test and tune their design. It is important to have tools that allow designers to check the vast design space, to avoid the oversight of good approaches just because of tools' limitations. Once a good tradeoff between expected performance and cost has been attained, the design can be rounded off using more detailed simulators. The evaluation of expensive prototypes goes just before the manufacture (and, again, evaluation) of the final product. In all these stages, evaluation has to be done using some kind of workload that resembles, with the higher possible fidelity, the actual workload that will be processed by the final network.

We insist in this point: if during initial stages tools impose limits to out experiments, we may reach erroneous conclusions about the benefits of our proposals. As we will see throughout this paper, availability of traffic generators based only on independent sources may be one of those (often ignored) limits.

The definition of a synthetic workload requires the selection of the injection process, the spatial traffic pattern and the message size [6]. This can be done in a per-node basic, although very often all nodes share the same behavior. The spatial pattern determines the distribution of destinations for each source node. The injection process determines the temporal distribution (in other words, when a packet is generated). The size distribution determines the message length.

Traffic patterns include permutations such as bit-reversal or matrix transpose, uniform (also called random), hot-spot, and hot-region. Each of them represents a worst-case scenario: uniform has not locality, permutations

make an uneven use of resources, and hot-spot and hot-region embody patterns in which some areas or nodes receive a higher proportion of the traffic.

Regarding the injection process, nodes are “programmed” to inject packets using some probability distribution. Each node progresses *independently* of the others. Injection times usually follow a Poisson or Bernoulli distribution, which are smooth over large time intervals. Recent works added on-off models that better characterize the self-similarity of traffic in some applications [18]. However, the set of processes that compose a parallel application can advance tightly or loosely coupled, with all the possibilities in between. But, *they are never totally uncoupled*. This is not reflected in an injection process that assumes each node is an independent traffic source.

The purpose of this paper is to show that performance results obtained under heavy loads with the usual injection process are misleading because they do not reflect the way actual parallel applications make use of the communication subsystem. The utilization of independent sources leads (for most traffic patterns) to an unfair access to network resources: some nodes inject as fast as they can, while others must compete with intensive passing-by traffic. This results in an uneven injection rate. Actual applications do not work that way: an exhaustive analysis of actual applications (of very different nature, ranging from scientific applications to on-line transaction processing) shows that all processes inject traffic at a similar rate, and no network or application-induced unfairness is observed.

To better illustrate this, we define an experimental setup designed to evaluate the impact on network performance of a restrictive injection mechanism, and we compare the results obtained using independent sources with those obtained using burst-synchronized traffic—a very simple traffic-generation method that incorporates some characteristics of actual applications, such synchronization among participating processes. Results with this alternative method clearly show the advantages of congestion control mechanisms, while when using independent sources researchers may conclude exactly the contrary.

The rest of the paper is organized as follow: Section 2 defines all relevant parameters of our experimental setup. In Section 3 we carry out a performance study of some interconnection networks under a collection of

traffic generation mechanisms and patterns, and also analyze the behavior of real applications. Finally, Section 4 summarizes the findings of this work.

2 Evaluation Environment and Example Architectural Proposal

When carrying out a performance study in order to analyze the advantages (or disadvantages) of a given architectural proposal, the choice of workload is critical, because if it does not reflect the characteristics of the actual workload that will use the system, extracted conclusions may be erroneous. This is valid for computer architecture in general, and for interconnection network design in particular.

In this section we describe an experimental setup that would be very familiar for researchers in our area of interest. It is used to analyze, via simulation, the performance of a restrictive injection mechanism aimed to prevent (or reduce) network congestion. Note that this is *just an example*: the focus of this paper is on the experimentation methodology, not in the architectural proposal. Next, we describe the simulators we have used. Then we describe the router architecture, traffic patterns and the rest of simulation parameters.

2.1 The Architectural Proposal: Congestion Control via Restrictive Injection

The example proposal for this study is the utilization of a congestion control mechanism that prioritize in-transit traffic that, supposedly, increase network performance for high loads.

Congestion control mechanisms restrict new injections when the network reaches a given level of congestion, which can be estimated locally or globally. Local methods are simple because each node restricts its own injection based on the congestion level of its own router. Global methods estimate network congestion on the network as a whole, so that a mechanism is needed to gather and distribute this information. For this paper, we apply a local method called *in-transit-priority restriction* (IPR): for a given fraction P of cycles, priority is given to in-transit traffic; in those cycles, injection of a new packet is only allowed if it does not compete with packets already in the network. P may vary from 0 (no restriction) to 1 (absolute priority to in-transit traffic), although in this paper we will consider only the two extreme cases. This kind of mechanism has been implemented in recent

interconnection networks, such as the torus network of IBM's BG/L [2] and the Alpha 21364 network [12]. A more detailed discussion of congestion control mechanisms can be found in [8].

2.2 The Simulators

For this work we have used two simulators that differ in the detail of the model they work with. The simplest one is FSIN (Functional Simulator for Interconnection Networks) [17], a cycle-driven simulator which is designed to deal with large networks (thousands of nodes) while using a minimal collection of resources (memory and execution time). The other one is SICOSYS, (SImulator of COmmunication SYStems) [14]. This simulator allows us to take into consideration most of the VLSI implementation details with high precision, but with much lower computational requirements than hardware-level simulators. The maximum error observed with respect to a standard hardware simulator is around 2%, providing in all cases pessimistic estimations. Its main drawback is the speed and memory utilization, so it can be used only for small networks.

We will start with studies performed with FSIN [17], an in-house developed simulator, very similar in design to ChaosSim [4], specifically designed to simulate adaptive virtual cut-through (VCT) networks based on router architectures like the one shown in Fig. 1. We could have used ChaosSim or FlexSim[1] [19], but the availability of an internally developed tool allowed us to easily modify the code to get more detailed information of the network behavior.

Both simulators can be parameterized in many ways: topology (mesh, torus, midimew), dimensions (1, 2 or 3), router architecture (switching technique, number of virtual channels, buffer sizes, number of injection channels, routing function, crossbar arbitration, etc.), applied workload (message size, traffic pattern, generation process), etc.

[1] In fact, we have used these two simulators to run some of the experiments reported in this paper, and they validate the data obtained from FSIN.

2.3 Router Architecture

The router architecture employed is shown in Fig. 1. It represents a virtual cut-through (VCT) router with three virtual circuits (VCs) per physical channel, each one with capacity for 8 packets (128 phits). These three VCs allow to map a deadlock-free oblivious (dimension-order routing) sub-network and a minimal adaptive sub-network. One of the VCs is used for the escape sub-network, relying on Bubble Flow Control (BFC) [16] to avoid deadlock in each dimension. The adaptive sub-network uses the other two virtual channels. Any blocked packet in the adaptive sub-network can resort to an escape path to break a potential deadlock cycle [6]. Such combination provides low-cost, deadlock-free adaptive routing.

Although BFC needs only two VCs to provide deadlock free, adaptive routing, we have chosen an architecture with three in order to compare with alternative, classic architectures that require this number of VCs to supply the same functionality [5][7]. It is important to remark that this work is focused on BFC, but results and conclusions presented here are also valid for other router architectures. The interested reader may check [11].

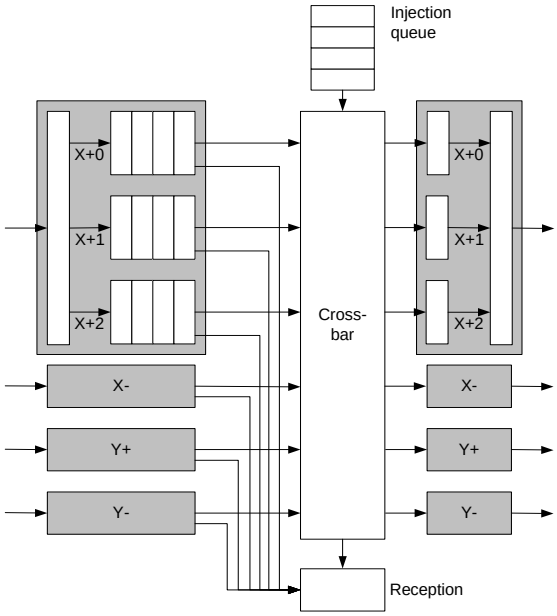


Figure 1. Architecture of the adaptive VCT router used in the experiments.

2.4 Synthetic traffic patterns and message size

We have considered fixed-size packets of 16 phits. In general, we cannot assume that applications running on a parallel computer use fixed-size messages. However, networks often impose a maximum packet size and messages have to be segmented to fit in several of those packets. For this reason, most studies are done with fixed-size messages of 8-32 phits [3][6][16][18]. In some cases, message length follows a bimodal distribution which reflects network workload for a cc-NUMA system [15]. In this study, we will limit our discussion and experiments to fixed-size packets, although conclusions are valid for other length distributions.

The traffic patterns used in the experiments are the following, although for the sake of clarity results for only a few of them will be shown in plots and tables:

- **BR**: bit-reversal permutation. The node with binary coordinates (a_{k-1}, a_{k-2}, ..., a_1, a_0) communicates with node (a_0, a_1, ..., a_{k-2}, a_{k-1}).
- **SH**: perfect-shuffle permutation. The node with binary coordinates (a_{k-1}, a_{k-2}, ..., a_1, a_0) communicates with node (a_{k-2}, a_{k-3}, ..., a_0, a_{k-1}) – rotate left 1 bit.
- **TR**: transpose permutation. In a 2-D network, the node with coordinates (x, y) communicates with node (y, x).
- **BC**: bit-complement permutation. Each node sends messages to a partner node that is identified by the bit complement of the bit-string representing the sender ID, modulo the total number of nodes.
- **UN**: uniform traffic. Each node selects destinations randomly in a packet-by-packet basis.
- **HR**: hot-region traffic. The destinations of 25% of the packets are chosen randomly within a small “hot” contiguous sub-mesh region consisting of 12.5% of the machine. The remaining 75% of the packets choose their destinations uniformly over the entire machine [2].

3 Performance analysis for independent and non-independent traffic sources

In this section we discuss the impact that the choice of workload—actually, the choice of injection process—has on results reported by simulators. The evaluation of the influence on performance of a restrictive injection

mechanism (IPR) on our adaptive VCT torus network is provided only to illustrate this issue. We insist in that could have selected different router architecture, topology, congestion-control mechanism… It would not matter because conclusions would be the same: performance results vary widely with the choice of applied workload.

3.1 Performance of the network under independent sources

As we stated before, most performance studies assume a network in which nodes are continuously generating packets, following the same injection process. Under this assumption, the figures of merit for performance are **latency** (time from packet generation until its delivery) and **throughput**, which is measured as the number of packets delivered in a given time interval divided by the interval length and the network size. In other words, this is the average load accepted by the network (i.e., the network throughput), which is expected to be even amongst the network nodes. Note that all nodes have an independent generator and try to inject at the same rate, which depends on the *applied load*. However, above saturation, the network may not be able to accept all the packets generated by nodes, so the *accepted load* (or throughput) may be different.

Fig. 2 represents a typical plot of one of these studies. We show the results of running the FSIN simulator under three different traffic patterns (UN, TR and SH), without and with IPR, using the "normal" injection process. "Normal" means **independent traffic sources**, each one following a Bernoulli distribution with a parameter that depends on the applied load. This load is varied from 0 to 1 phit/cycle/node. The simulator runs for a warm-up period of 100 kcycles, plus a measurement period of 100 kcycles.

For the UN pattern, results show that utilization of a restrictive injection mechanism eliminates the throughput loss for loads beyond congestion. However, we cannot extend this conclusion to the permutations. In fact, results indicate that restrictive injection is counterproductive for TR and SH traffic under heavy loads.

However, another indicator of network performance of widespread use is **channel utilization**: the higher the channel utilization, the better, because more resources are being productive. Let us focus on TR traffic without/with IPR. Fig. 2 indicates that, in saturation, throughput is higher without IPR. However, simulation results also indicate that channel utilization is higher with IPR. So, a question arises: does IPR increase performance, or not?

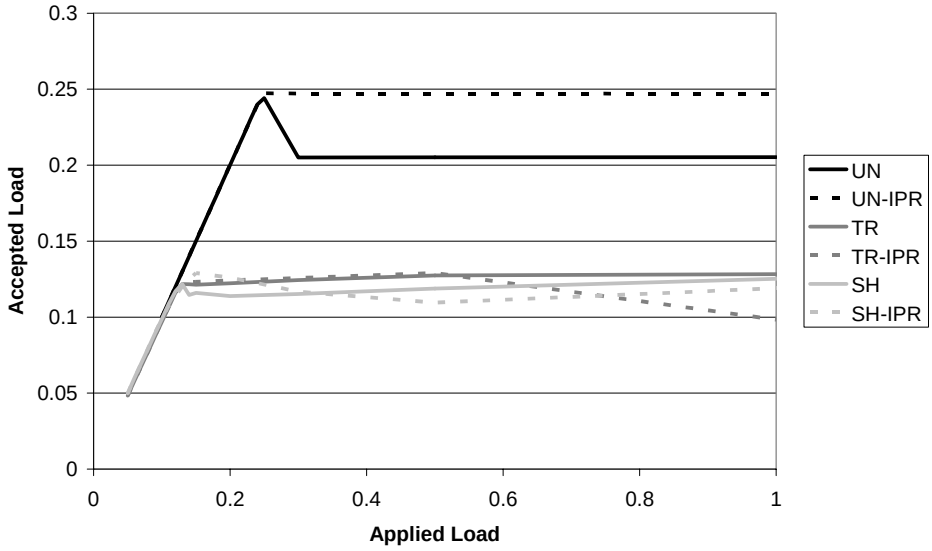


Fig. 2. Applied load vs. throughput (phits/cycle/node) for UN, TR and SH patterns, without/with IPR.

3.2 Discussion of results

In [6], Dally & Towles suggest that performance of a network for a given traffic pattern in which the node injection rate is not the same for all nodes should be reported as the lowest injection rate that matches the desired workload. Following this approach, in Table 1 we report maximum, minimum and average injection rates for the six configurations under study. Notice the vast differences between these values for the TR and SH permutations—and take into account that, in many experimental studies, only the average values are considered.

Table 1. Maximum, minimum and average network throughput (phits/cycle/node), for applied loads beyond saturation, for UN, TR and SH patterns, without/with IPR.

	UN		*TR*		*SH*	
	IPR off	*IPR on*	*IPR off*	*IPR on*	*IPR off*	*IPR on*
Max.	0.219	0.267	0.559	0.716	0.973	0.974
Min.	0.194	0.217	0.013	0.000	0.002	0.000
Avg.	0.205	0.243	0.132	0.098	0.125	0.119

Dally & Towles also state that average and minimum rate differ in some routers due to their unfair design; however, the differences shown in Table 1 are not caused by an unfair routing or arbitration method, but by the

fact that network resources are used unevenly by the applied workload. We should note that the time a packet awaits in an injection buffer before entering the network depends not only on the arbitration method, but also on the local router state. Under UN traffic, the network load is evenly distributed, so that all nodes have a similar view of network status and are able to inject packets at a similar rate. However, under non-uniform loads the degree of utilization of resources (buffers, output channels) may vary widely from one router to another. Therefore, at high loads, nodes connected to busy routers[2] have lower chances to inject their load than nodes in less used areas—a difference that causes wide variations in the number of packets injected by each node.

Let us focus again on the TR permutation. In a 32x32 network, and assuming that all nodes inject at the same rate, the average distance packets traverse is 16.516. In fact, this is what the simulator reports when network operates below the saturation point. The map of packets injected per node is flat (except for those nodes in the diagonal, which do not generate traffic for themselves), as shown in Fig. 3a.

The scenario changes drastically when saturation is reached. Some nodes can inject packets in their routers at very high rates, while others can hardly access the network, because their routers devote most resources to passing-by packets. The simulator reflects this in a change in average distance (17.12) and in a very different map of injected packets (Fig. 3c). Note that “lucky” nodes (those that have more opportunities to inject packets) are located close to the diagonal and in a pair of bands parallel to it. It gives the impression that the network is *unfair* for TR traffic.

[2] “Busy” routers are those that are traversed by numerous in-transit packets.

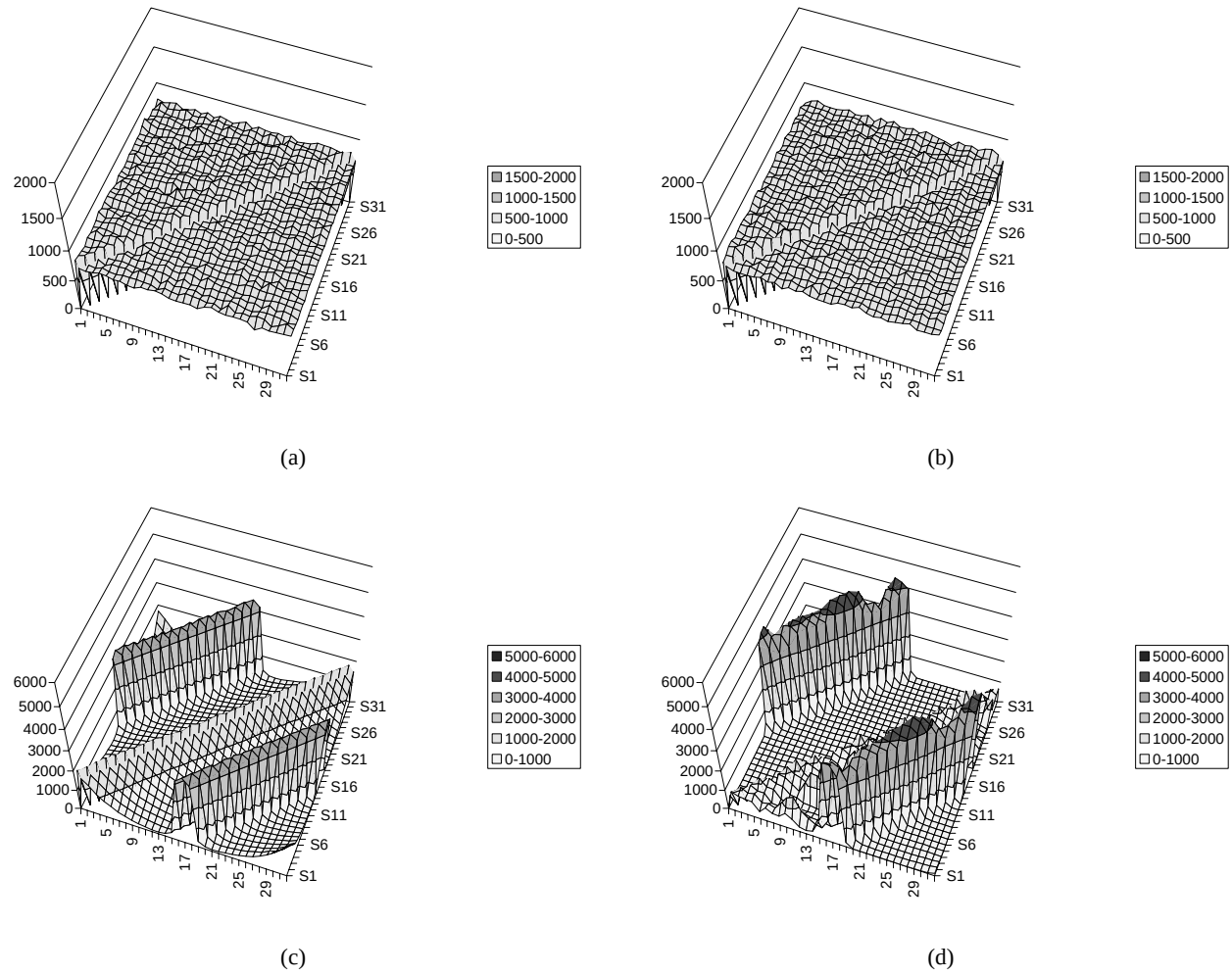


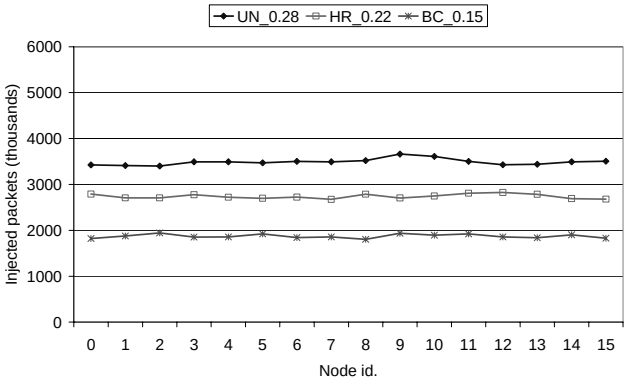
Fig. 3. Maps of injected packets for TR traffic. Each surface point (x, y) represents the number of packets a node with coordinates (x, y) injected in 100.000 cycles. (a) Below saturation, no IPR. (b) below saturation, IPR. (c) beyond saturation, no IPR. (d) beyond saturation, IPR.

When the IPR congestion control mechanism is added to the network, nothing changes below saturation (Fig. 3b). However, beyond saturation the unfairness appears again, magnified (Fig. 3d): the “lucky” area in the diagonal disappears, and the parallel bands are smaller and higher than without IPR. As nodes in these bands are injecting packets addressed to distant destinations, the average distance rises up to 23.47.

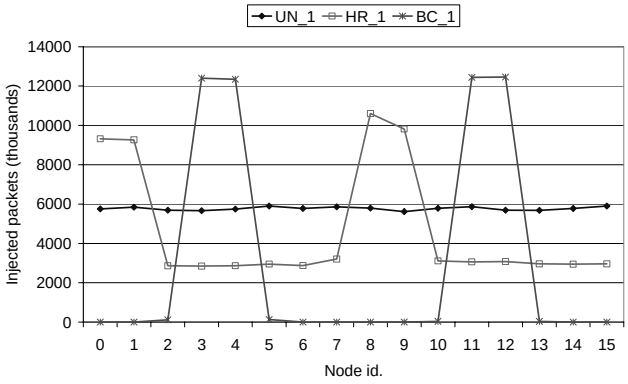
This behavior for non-uniform patterns is independent of the number of nodes and of network topology. For example, if we perform a similar analysis in a 16-node ring, we observe the same phenomenon, with and without IPR. In Figure 4 we plot the number of packets injected by each one of the 16 nodes during 200 kcycles (after a

warm-up period of 200 kcycles) using three patterns: UN, HR and the BC permutation[3], without IPR. Fig. 4a corresponds to a lightly loaded system: 16 nodes are injecting at a pre-established rate (0.28 phits/cycle/node for UN traffic, 0.22 for HR and 0.15 for BC) which is well below the network capacity—actually, a 40% below the saturation point. Fig. 4b corresponds to a situation where nodes *try* to inject at 1 phit/cycle/node, well above saturation. As we see, when network is congested, unfairness appears for non-uniform patterns: some nodes are able to inject at a very high rate, obstructing the remaining ones. In this context, an "average network throughput" for BC traffic is meaningless.

This experiment is useful to show an undesired network behavior (the uneven distribution of resources), but is utterly unrealistic because, in a real parallel application, faster nodes would need to synchronize with slower ones in such a way that, instead of greedily using its advantageous position, they will eventually stop (or reduce) packet injection, thus giving chances to other nodes to access the network. These considerations would be completely different if we were simulating another network, such as the Internet, where traffic sources are *really* independent. However, our focus is on networks for parallel computers able to run applications such as those studied in the next subsection.



(a)



(b)

Fig. 4. Graphs of injected packets for UN, HR and BC traffic, for independent sources, without IPR. Each point represents the number of packets injected by the corresponding node during 200 kcycles. (a) For applied load 40% below saturation. (b) For applied load above saturation.

[3] BC is used as a representative permutation. Results are very similar for others, such as SH and BR.

3.3 Behavior of Actual Applications

The huge difference in ability to inject packets of some nodes, compared to others (that suffer a situation close to starvation) led us to perform an analysis of real applications, to provide evidence to verify our hypothesis (that this situations cannot happen, because of the synchronization and coupling among processes). We selected a collection of applications of different characteristics, running them in the full-system simulator GEMS [10] in which SICOSYS simulates the interconnection network. In Fig. 6 we represent the number of messages sent by each processing element of a 16-node cc-NUMA directory based machine when running a subset of such benchmark applications.

The applications range considered in this analysis go from typical scientific workloads derived from NPB suite [9] (“LU”, “BT”, “CG”, “FT” and “MG”) to more server oriented workloads. In particular here we are using a database benchmark (“OLTP”), web server benchmark based on SURGE (“Apache”) and SPECjbb2000 (“Java”).

For most applications, we observe that all participating processes generate a similar number of packets. In some cases, some nodes generate appreciably more traffic that the remaining ones[4], but no node is close to starvation.

[4] This characteristic is common in many applications, where a node act as the “master” of some tasks.

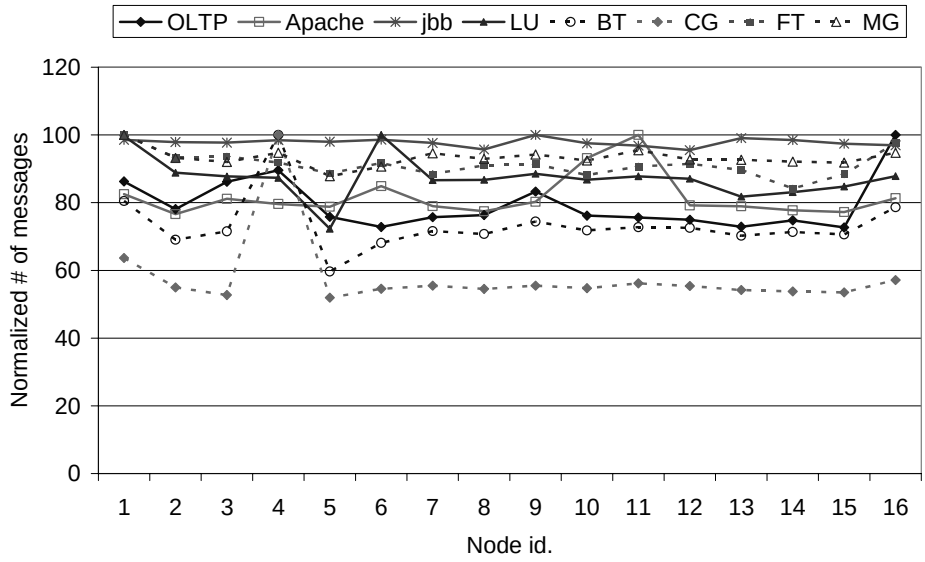


Fig. 5. Normalized number of packets sent by processes for a set of parallel applications running in a cc-NUMA machine with 16 nodes.

In parallel applications, processes are *coupled* one way or another, because they work to perform a given task in a cooperative way. Most (if not all) applications use synchronization barriers, perform collective operations or use other mechanisms that make all the processes advance at a similar rate. It is true that worst-case performance for data exchanges is important (as shown in [13]) because it may halt progress of computation nodes, which are not able to perform additional operations, or communicate any further, until the data exchange has been completed. However, we cannot conceive a realistic scenario in which, *in the same parallel application*, a process is sending packets to its selected destination *ad infinitum* while other nodes do the same at a *much* smaller rate. In a tightly-coupled application, processes will advance at a similar rate; in a loosely-coupled one some nodes *may* advance much faster than the others (even more if the network favors them), but at some point they will stop and spend some time waiting a sort of “go ahead” signal—freeing network resources that will be used by the other nodes. Consequently, in our simulations, we must incorporate a coupling mechanism in the traffic generation process, instead of relying on dependent traffic sources.

3.4 Performance under Burst-Synchronized Traffic

We have made a complete performance analysis similar to that reflected in Fig. 2, but using burst-synchronized traffic, i.e. non-independent sources, to reflect the synchronized nature of parallel applications. The injection method is similar to that described in [3]. The same workload (*b* packets) is assigned to each source of traffic. A burst starts with an empty network. Nodes inject their *b* packets as fast as the network accepts them. The burst ends when all packets of all the traffic-generating nodes have been consumed.

Fig. 6 shows the time to complete 5 bursts of 1K packets for the six scenarios under consideration. For comparison purposes, Table 2 shows their throughput computed as the total workload delivered divided by the completion time. For this workload, maps of injected packets are meaningless (all nodes inject exactly the same number of packets), and the reported average distance traversed by packets is always the expected one[5]. Now we observe that, under burst-synchronized workload, the use of restricting injection policies is positive for the three traffic patterns: the time to deliver the 5 bursts of packets is lower with IPR than without it.

This conclusion contradicts that of Section 3.1 and it is clear which one is correct. Both are based on synthetic workloads and both are just approximations to the reality, but burst-synchronized traffic results more appropriate for the kind of evaluation we are performing, because it introduces a synchronization mechanism, as real applications do, that avoid unfairness in the use of network resources.

[5] In this context starvation is not an issue: if the network somehow favors some nodes, they will send their workload faster than others, but will eventually stop, allowing the rest of the nodes to progress faster, until all of them have sent their packets.

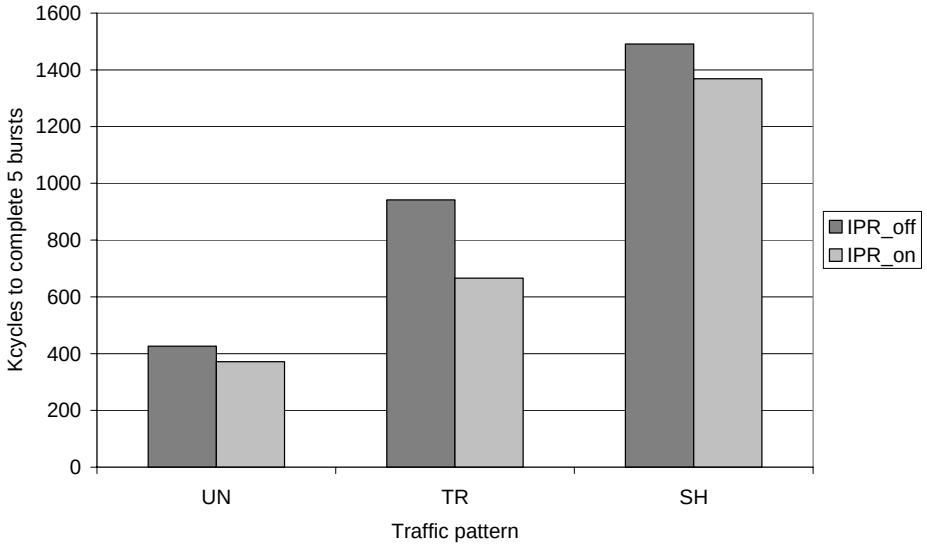


Fig. 6. Time to deliver 5 bursts of 1K packets for UN, TR and SH patterns, without/with IPR.

Table 2. Network throughput (phits/cycle/node) averaged for 5 bursts under UN, TR and SH patterns, without/with IPR.

	UN		*TR*		*SH*	
	IPR off	*IPR on*	*IPR off*	*IPR on*	*IPR off*	*IPR on*
	0.192	0.220	0.087	0.123	0.055	0.060

4 Conclusions

The main conclusion of this paper is that synthetic workloads with the underlying assumption of independent sources, widely used in the initial stages of interconnection network design, may be less than appropriate in many cases. In particular, we have shown that, for loads beyond saturation, this workload model can lead to incorrect conclusions. It is essential to check, when performing an analysis of a design proposal, that applied workloads do reflect the basic aspects of parallel applications. We need to understand how each workload makes use of the network resources, and draw conclusions using more than a single figure of merit. In the case described in this paper, an uneven utilization of resources produced meaningless throughput results at heavy loads. Without any further analysis, we could have concluded that congestion control mechanisms based on restrictive injections have a negative impact on performance for most traffic patterns, so we would not consider

its implementation in actual routers. However, the fact is that restrictive injection is actually useful in most scenarios.

The best choice would be to apply real workloads, but this presents a series of difficulties. One is to establish which those workloads would be. Then, if we use traces, they have embedded some characteristics of the system where they were captured, that may be inappropriate for the system under test. The alternative of using execution-driven simulation is extremely costly. Thus, the bulk of interconnection network research is done using synthetic workloads that allow us to test the network capabilities in many ways.

Our solution was to switch our simulations to burst mode, with large bursts. This workload introduces some synchronization points (something typical of parallel applications) and is able to saturate the network for large periods of time, with the advantage of assuring that all nodes inject the same number of packets—something that, while still unrealistic, is a better approximation to actual applications' behavior.

References

[1] A. R. Alameldeen, M. M. K. Martin, C. J. Mauer, K. E. Moore, M. Xu, D. J. Sorin, M. D. Hill, and D. A. Wood. "Simulating a $2M Commercial Server on a $2K PC". IEEE Computer, 36(2):50–57, Feb. 2003.

[2] M. Blumrich, D. Chen, P. Coteus, A. Gara, M. Giampapa, P. Heidelberger, S. Singh, B. Steinmacher-Burrow, T. Takken, P. Vranas. "Design and Analysis of the BlueGene/L Torus Interconnection Network" IBM Research Report RC23025 (W0312-022) December 3, 2003.

[3] T. J. Callahan, S.C. Goldstein. "NIFDY: A Low Overhead, High Throughput Network Interface". Proc. of the 22nd Annual International Symposium on Computer Architecture, ISCA '95, Santa Margherita Ligure, Italy. pp. 230-241, June, 1995.

[4] The Chaotic Routing Project at the U. of Washington. Chaos Router Simulator. Available at http://www.cs.washington.edu/research/projects/lis/chaos/www/chaos.html

[5] W.J. Dally, C.L. Seitz, "The Torus Routing Chip" *Distributed Computing* vol 1 pp. 187-196. 1987.

[6] W.J. Dally & B. Towles. Principles and Practices on Interconnection Networks. Morgan Kaufmann, 2004.

[7] J. Duato. "A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks". IEEE Trans. on Parallel and Distributed Systems, vol. 7, no. 8, pp. 841-854, 1996.

[8] C. Izu, J. Miguel-Alonso, J.A. Gregorio. "Packet Injection Mechanisms and their Impact on Network Throughput". Technical report EHU-KAT-IK-01-05. Department of Computer Architecture and Technology, The University of the Basque Country. Available at http://www.sc.ehu.es/acwmialj/papers/ehu_kat_ik_01_05.pdf.

[9] H. Jin, M. Frumkin and J. Yan. “The OpenMP Implementation of NAS Parallel Benchmarks and its Performance”. Technical Report NAS-99-011, NASA Ames Research Center. October 1999.

[10] Milo M.K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood,, “Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset”, available in www.cs.wisc.edu/gems/

[11] J. Miguel-Alonso, C. Izu, J.A. Gregorio. "Improving the Performance of Large Interconnection Networks using Congestion-Control Mechanisms". Technical report EHU-KAT-IK-06-05. Department of Computer Architecture and Technology, The University of the Basque Country. Submitted.

[12] S. Mukherjee, P. Bannon, S. Lang, A. Spink and David Webb, “The Alpha 21364 Network Architecture”, IEEE Micro v. 21, n. 1 pp 26-35, 2002.

[13] F. Petrini, D. Kerbyson and S. Pakin. “The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q”. In IEEE/ACM SC2003, Phoenix, AZ, November 2003.

[14] V.Puente, J.A. Gregorio, R.Beivide, SICOSYS: An Integrated Framework for studying Interconnection Network in Multiprocessor Systems, Proceedings of the IEEE 10th Euromicro Workshop on Parallel and Distributed Processing. Gran Canaria, Spain. January 2002.

[15] V. Puente, J.A. Gregorio, R. Beivide and C. Izu, “On the Design of a High-Performance Adaptive Router for CC-NUMA Multiprocessors”, IEEE Trans. on Parallel and Distributed Systems, Vol. 14, NO. 5, May 2003.

[16] V. Puente, C. Izu, R. Beivide, J.A. Gregorio, F. Vallejo and J.M. Prellezo (2001). “The Adaptative Bubble Router”. Journal of Parallel and Distributed Computing. Vol 61 - n. 9.

[17] F.J. Ridruejo, J. Miguel-Alonso. "INSEE: an Interconnection Network Simulation and Evaluation Environment". Technical report EHU-KAT-IK-04-05. Department of Computer Architecture and Technology, The University of the Basque Country. Available at http://www.sc.ehu.es/acwmialj/papers/ehu_kat_ik_04_05.pdf.

[18] J Shin, TM Pinkston. “The Performance of Routing Algorithms under Bursty Traffic Loads”. Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications. Las Vegas (USA), Jun. 2003.

[19] SMART group at the U. of Southern California. FlexSim 1.2. Available at http://ceng.usc.edu/smart/FlexSim/flexsim.html