Realistic Evaluation of Interconnection Network Performance at High Loads

F.J. Ridruejo[1], J. Navaridas[1], J. Miguel-Alonso[1], Cruz Izu[2]
[1] *Dep. of Computer Architecture and Technology, UPV/EHU, Spain*
[2] *School of Computer Science, The University of Adelaide, Australia*
{franciscojavier.ridruejo, javier-navaridas, j.miguel}@ehu.es, cruz@cs.adelaide.edu.au

Abstract

Any simulation-based evaluation of an interconnection network proposal requires a good characterization of the workload. Synthetic traffic patterns based on independent traffic sources are commonly used to measure performance in terms of average latency and peak throughput. As they do not capture the level of self-throttling that occurs in most parallel applications, they can produce inaccurate throughput estimates at high loads. Thus, workloads that resemble the varying levels of synchronization of actual applications are needed to study the performance of interconnection networks. One approach is to use simple, burst-synchronized synthetic workloads that emulate the self-throttling of many parallel applications. To validate this approach, we compare the gains achieved by a restrictive injection mechanism under this workload with those obtained using traces from the NAS Parallel Benchmarks. This study confirms that the burst-synchronized traffic model provides reasonable performance estimates, which could be improved by taking into account dependency chains between messages.

1. Introduction

The interconnection network (IN) is a key element of any parallel computer, more so when executing communication-intensive applications. Network performance has been evaluated using many techniques, including: network simulation with synthetic loads, analytical models, trace-driven simulation and full-system simulation. Traditionally, network simulators measure performance under a well-known range of synthetic traffic patterns such as uniform, hot-spot and permutations. These patterns model worst-case scenarios of little locality and unbalanced usage of network resources [4]. Analytical models have been proposed for simple networks but they rely on unrealistic assumptions, such as uniform traffic or infinite injection and delivery queues, so they have limited use. A full system simulation, in which traffic is provided by an execution-driven simulator such as Simics [16], provides more meaningful results but limits the evaluation to small networks.

Congestion is a well-known problem in standard computer networks [8], but most INs in the literature (most with a standard size of 256 nodes) did not exhibit throughput degradation at loads beyond saturation. Such network proposals had single injection queues and, when the network is small, the head-of-line blocking at injection is enough to throttle the network and control congestion [7]. This is not the case for large networks with multiple injection sources such as IBM's BlueGene/L [2], which are prone to suffer from congestion at heavy loads. Consequently, new congestion control techniques have been proposed and evaluated for wormhole [1,15] and virtual cut-through INs [9,11,14]. Most of these works carry out evaluations using synthetic traffic patterns that assume that nodes generate traffic independently of each other. Although they ignore the different levels of coupling and synchronization that exist in parallel applications, synthetic loads seemed to provide reasonable indicators of network performance for a range of parallel benchmarks [12]. Note that most parallel applications will apply some level of self-throttling as nodes synchronize and may stop sending new messages as they wait for messages delayed by congestion. Thus, any evaluation of an IN at heavy loads, and in particular any evaluation of a congestion control technique, should be done under loads that reflect the synchronization and coupling among application processes.

In [6] burst-synchronized synthetic traffic was proposed to make a fair evaluation of the congestion control technique IPR (In-transit Priority Restriction), the one used in the torus network of the IBM BlueGene/L [2]. In this paper we will consider another congestion control technique called Local Buffer

Restriction (LBR), which uses local information to regulate the admission of new traffic. We will compare results obtained using burst-synchronized loads with those obtained by applying actual loads taken from traces of the NAS Parallel Benchmarks (NPB). If the synthetic loads reflect the synchronized nature of parallel applications, both results should lead to the same conclusions regarding LBR, and we would have a means to evaluate very large systems. Note the main goal of the work is not to evaluate LBR but to find out if burst-synchronized loads are a good approximation of real loads.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the experimental, simulation-based workbench. Section 4 compares and contrasts the results of the different experiments. Finally, Section 5 summarizes the findings of this work.

2. Motivation and Related Work

Most IN studies rely solely on synthetic traffic patterns, which include uniform traffic, hot spot traffic and permutations [4]. The figures of merit are latency at low loads and peak network throughput. The need to characterize network workload and produce better synthetic models was identified long time ago [3], although there has been little progress since. Instead of developing new synthetic loads, some IN studies combine the standard evaluation with real workload evaluation [12]. Other studies use synthetic loads that mimic the bursty nature of network traffic [14] extending the standard packet generation, which followed a Poisson or Bernoulli distribution, with a sequence of ON/OFF states, so that packets are generated only during the ON state. Besides, most studies normalized the applied load to the network bisection limit, so that the networks were not tested for loads beyond their theoretical capacity. As most network proposals sustained their maximum throughput after saturation, the evaluation of the network at heavy loads was not considered of interest until recently. However, congestion is a problem for large INs with multiple injection sources, and congestion control mechanisms are tested at loads beyond saturation [1,9,14] in order to see if they prevent traffic bursts from degrading network performance.

Synthetic loads used to evaluate congestion control techniques fail into two categories: static loads, which use the same pattern and injection rate over the time each experiment runs, and dynamic loads which alternate between phases of high and low injection rates. For static loads, only the steady-state performance is studied, and the figure of merit is average sustained throughput. For dynamic loads the figure of merit is total execution time, which is a better indication of the network ability to cope with communication intensive phases.

The evaluation of congestion control techniques using static loads has shown that unbalanced traffic loads may result in network unfairness at saturation [6]: some nodes have less chance to inject traffic than others. The source of this unfairness is the unbalanced utilization of resources, derived from the traffic pattern. The more in-transit packets a router has to manage, the less opportunity it has to inject its own traffic. Besides, an unbalanced load distribution results in the formation of persistent zones of congestion at high loads. In this context, average throughput as reported in [1,8,15] is not representative of application loads, because “fast” injecting nodes will eventually wait for the slow ones to catch up. Therefore, in order to obtain useful performance figures at heavy loads, a static synthetic traffic pattern should reproduce the level of coupling that exists amongst traffic sources in actual parallel applications.

Non-static loads reflect the fact that communication in many parallel applications is not constant over time, so that an intensive communication phase (with high traffic volume) will be followed by a computation intensive phase (with low traffic volume), Recent studies using non-static traffic patterns include [1,14] in which traffic is uniform but load alternates between low and high phases, and [15] in which each high phase uses a different pattern. As mentioned before, the figure of merit is total execution time. As the high phases will exhibit network unfairness, a computing node located in a less clogged area will be able to advance to the next phase ahead of the rest, an unlikely scenario in a parallel application. In other words, although non-static synthetic loads reflect the temporal variations that occur in application loads, they still fail to model the synchronization and coupling amongst application nodes.

Burst-synchronized traffic deals with this issue by modeling the barrier synchronization primitives used in many parallel applications, either explicitly (in the form of collective operations) or implicitly. This synchronized traffic has been sparingly used in studies focused on injection issues [3,6], claiming that it represents realistic workloads, but there is no formal study confirming or denying this fact. Our work tries to fill this gap by comparing the insights obtained from synthetic loads with those from actual workloads taken from application traces.

3. Experimental Setup

3.1 The Simulation Environment

Experiments have been performed using the evaluation environment described in [13]. It consists of an IN simulator and a traffic-generation module, which provides traffic from one of these sources: synthetic generation, traffic as recorded in traces or interfacing with Simics [16] to perform a full system simulation [11].

We perform most of our experiments using a small network of only 64 nodes. This is because the trace-capture setup imposes us limits that are close to this value (due to availability of resources in a production machine, and to the sizes of the trace files). As bisection bandwidth does not increase linearly with the radix, networks with large radix reach saturation at lower injection rates, and they are more likely to suffer overloads during intensive communication phases. Thus we focus our study on 64-node rings, instead of using 8x8 tori. The ring is adequate to experiment with congestion, and results with this topology can be extended to multidimensional Ins, because congestion causes messages to rely on the escape sub-network, in which they must traverse the x-ring first.

The models of routers used in the experiments are depicted in Fig. 1. Each physical channel in the router is split into three virtual channels (VCs): an Escape channel (governed by the bubble routing rules [12]), and two adaptive channels. Note that a ring network has just one minimal path from source to destination, so packets cannot adapt. Thus, the only difference between the Escape VC and the other two is that access to the "adaptive" VCs is not restricted by the bubble rules. In the case of 2D tori, packets in adaptive VCs can use any minimal path to reach their destinations.

Each node is able to simultaneously consume several packets arriving to the reception port. There are two injection ports, and the interface should perform a pre-routing decision: packets moving towards the X+ axis are stored in the I+ injection port, and those towards X– go to the I– injection port. Transit and injection queues are able to store 4 packets of 16 phits each. Phit length is 4 bytes (32 bits).

Each experiment has been repeated 10 times, using different random seeds. We measure execution times in terms of simulation "cycles". As these times vary between patterns or benchmarks we have represented the average value (obtained from 10 simulation runs) normalized to the Base case.

Regarding synthetic traffic we use UN (uniform) and HR (hot-region). In both cases destinations are chosen randomly; in the case of HR [2], 1/4 of the traffic goes to the first 1/8 nodes, and the remaining traffic is uniform.

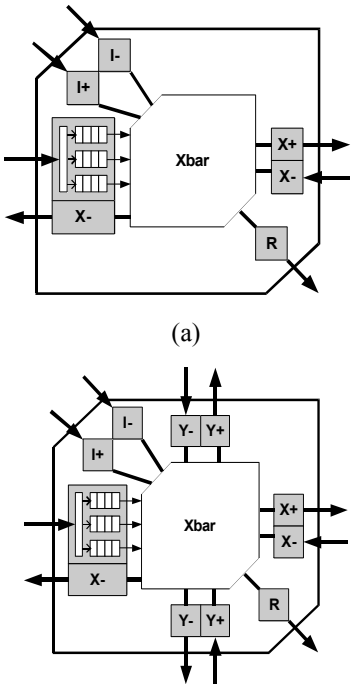


Fig. 1. Models of the simulated routers, (a) router for rings, and (b) router for 2D tori, with a detailed view of the X+ input port showing the 3 virtual channels that share its link

Traces used in this work have been obtained from clusters of commodity PCs and from the Mare Nostrum, running some of the NAS Parallel Benchmarks (NPB) [10]. To obtain these traces we modified the trace capture tool included in MPICH, in order to register all the point-to-point operations—including those that implement the collective operations [13]. As we want the network to become the bottleneck, we provided workload to the simulator as fast as we can, regardless of the timestamps found in the traces. However, in order to preserve the causal relationship among messages, we maintained the order shown between arrivals and new injections: when the trace file indicated that a message was received at a given node, injection of packets from that node sent later in the trace is delayed until that message arrives. This way, we measure the number of network cycles that, in the simulated network, are necessary to complete all the interchanges of messages stored in the trace files.

3.2 Congestion Control Techniques

As congestion is caused by overloading the network with too many packets, congestion control techniques deal with it by limiting packet injection as soon as the network exhibits signs of being congested. They differ in the way congestion is diagnosed.

Global methods estimate congestion by examining the status of the whole network (for example, the number of packets held in the routers, as in [15]); thus a mechanism is needed to gather and distribute that information. Local methods are simpler because each node restricts its own injection based on its own congestion level. Multiple congestion control methods are evaluated in [8]. As this is not a work on congestion control, we consider only one simple local method that has shown good performance, LBR.

Most routers split each physical link into several VCs in such a way that the combination of an escape sub-network with one or more adaptive sub-networks provides deadlock-free adaptive routing [5]. The LBR mechanism has been designed specifically for adaptive routers that rely on Bubble Flow Control to avoid deadlock in the escape sub-network [12]. A previous study showed how the bubble restriction also provides congestion control for the escape sub-network [7]. LBR extends this mechanism to the rest of the VCs. That is, a packet can only be injected into an adaptive VC if such action leaves room for at least B packets in the transit buffer associated to that VC. Parameter B indicates the buffer space reserved for in-transit traffic. In other words, congestion is estimated by the current buffer occupancy. We can vary the degree of restriction in the injection by modifying this parameter. In this work we use B=3 (out of 4-packet buffer), so packets are injected in an adaptive channel only when its queue is empty or almost empty.

4 Analysis of the Experiments

4.1 Experiments using Burst-Synchronized Traffic

The utilization of burst-synchronized traffic to model application loads was proposed in [6]. With this traffic, each node tries to inject a burst of b packets as fast as the network is able to accept them; then the node stops. Nodes will start injecting another burst only when all the packets of the previous one have been consumed. The figure of merit in these experiments is the time to consume a burst. We have considered a collection of values for b, trying to emulate different degrees of coupling among application processes, from 10 (the most tightly-coupled) to 10.000 (the most loosely-coupled). For applications that synchronize using barriers, we can interpret b to be the number of packets sent between two barriers. Note that, in a VCT network, long messages are packetized, so a long message generates a burst of packets to be injected in the network. Therefore, message size and b are directly related.

Execution times have been normalized to the time of the Base case (LBR deactivated). The time relation between base case and LBR is plotted in Fig. 2 for both random traffic and hot-region traffic. We can see that LBR is effective in reducing the time the network requires to deliver one burst. As expected, the gains increase with the number of packets sent per burst. LBR is more effective in the 2D network, as adaptive routing increases pressure on network resources and causes higher contention than in the 64-ring counterpart. Thus, the evaluation of LBR using a 64-ring give us a conservative estimate of the gains achieved by LBR on multidimensional network with similar radix.

4.2 Experiments Using Application Traces

We now explore the relationship between b and the problem size of the NPB. This suite can be compiled and run for a variety of problem sizes, denoted S, W, A, B, C and D—where S is the smallest and D is the largest. Larger problems use larger data structures, and this entails interchanges of larger messages, although the pattern of interchanges remains the same. There is, though, an exception to this rule: for LU, as the problem size increases, the number of messages interchanged also increases, in addition to its size.

For a second set of experiments, we have generated traces for a variety of problem sizes for all the NPB, and processed them through the network simulator. We have used the official sizes defined for the applications, and added some intermediate cases when necessary (for example, the difference in problem size from S to W for IS is too wide, so we have added those cases denoted as “T”). Figures 3(a) and 3(b) show the results obtained for IS and CG respectively. Compare with the curves in Fig. 2 for the synthetic patterns UN and HR. Again, as the problem grows, the benefits of LBR are more visible. Although congestion control can be counterproductive in some cases (CG), its penalty is small.

4.3 Analysis of Application Traces

Results from previous sections predict that LBR is clearly beneficial in some applications, but may have a negative impact in others.

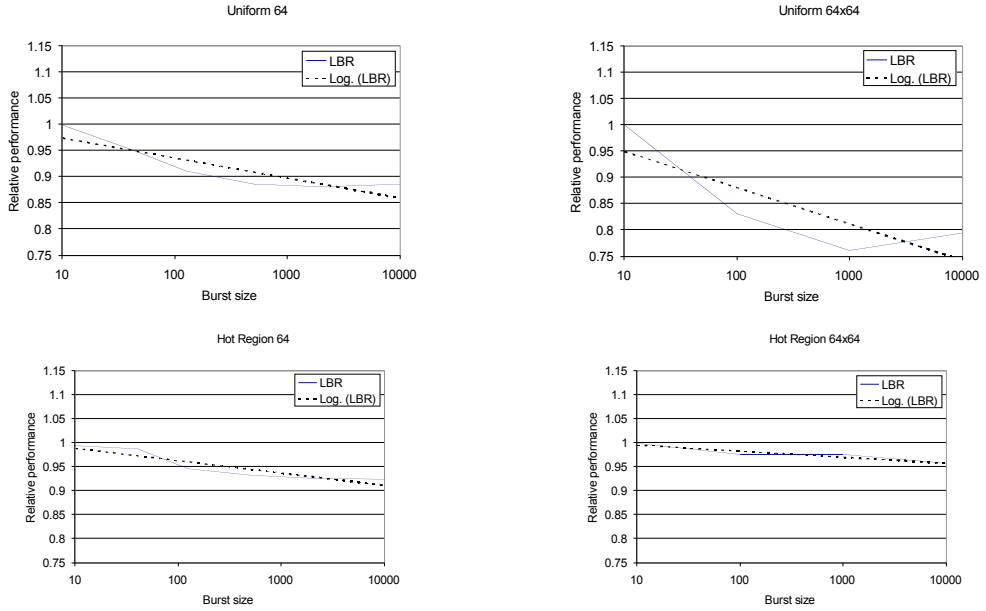
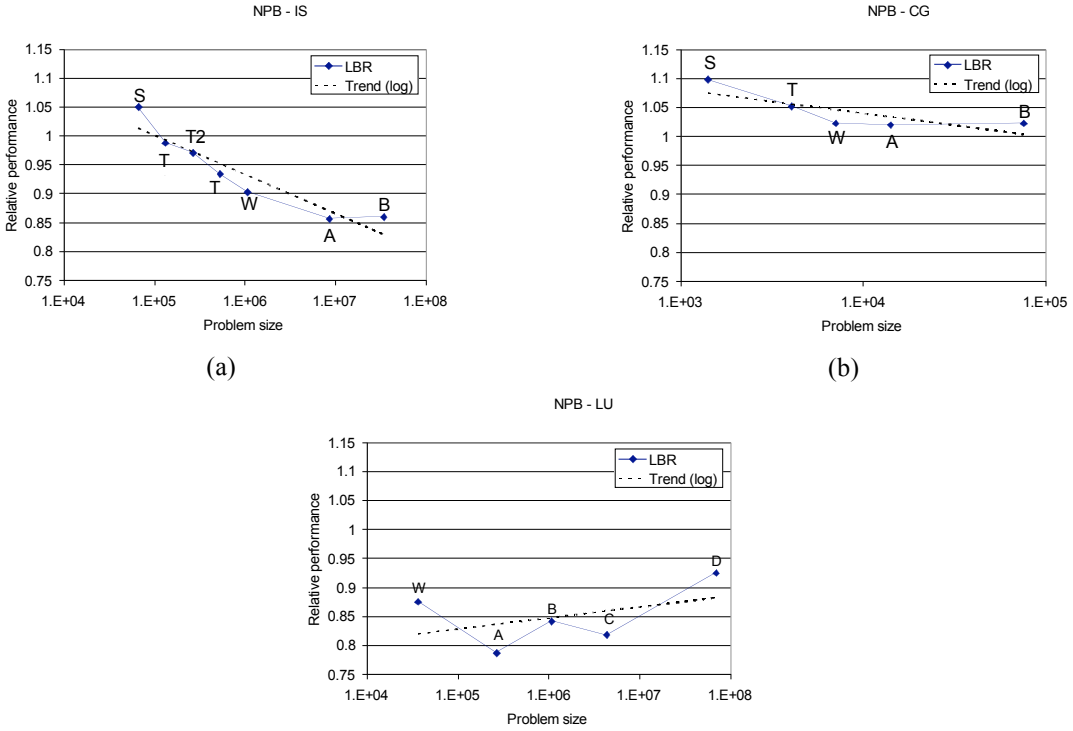


Fig. 2. Performance of 1D and 2D tori dealing with bursts of uniform and hot-region traffic. Normalized times to consume a burst, and trend lines (X axis is logarithmic).



(a) (b)

(c)

Fig. 3. Relationship between problem size and efficiency of LBR for benchmarks IS (a), CG (b) and LU (c). Measured values and trend lines. Scale in the X axis is logarithmic. Y axis represents the time to consume the applied workload, relative to the base case without LBR.

We have related this to the problem size and to the burst size b in our synthetic model, but we have also observed that this approach is not valid for all cases. In this section we analyze the traces of CG and LU (size A, 64 processors), obtained in the Mare Nostrum, to better understand the relationship between the traffic patterns and the effect of congestion control.

CG is the only application of the NPB suite that does not benefit from congestion control—just the opposite. A visualization of the trace of CG.A.64 (see Fig. 4) shows that this application consists of a series of iterations, each of them with these phases:

(1) An interchange of messages: each node sends and receives 7 messages. The first 4 are long (~14 KB) and the remaining 3 are very short (8 B). The first two long messages go to nodes nearby, while the remaining two must traverse longer distances. The first short message goes to a distant node, while the remaining two go to nearby nodes.
(2) A short computation phase.
(3) A second interchange of three very short messages, between nearby nodes.
(4) A longer computation phase.

In short, this benchmark exhibits long communication-synchronization chains: a message is sent only when triggered by the reception of another one. These chains are of length 7 in phase 1, and of length 3 in phase 3. The number of messages traversing the network simultaneously is never very large, and in most cases they go to close destinations. Any delay injecting a given message (for example, to prioritize in-transit traffic, as LBR does) results in additional delays injecting messages than depend on it. We can expect to obtain maximum benefit from congestion control in communication-intensive phases with a mixture of short-distance and long-distance packets, without interdependencies. CG does not fulfill these requirements, so the observed performance drop.

LU is the application that benefits most from congestion control. However, we expected more improvements for larger problem sizes, and this does not happen. The traces help us understanding the reasons. A study of the trace of LU.A.64 shows that this application also consists of a series of iterations, each of which has 7 phases:

(1) A cascade of short, chained messages (600 B) initiated at node #0, flowing downward to the remaining nodes; messages go to destinations at distance 1 or 8.
(2) A similar, upward cascade, started at node #63.
(3) A computation phase.
(4) An interchange of a long message (~40 KB) with a neighbor.
(5) A computation phase.
(6) Another interchange of a long message, with a node at distance 8.
(7) A computation phase.

Fig. 5 illustrates the 7 phases for LU.A.16. The size was reduced to 16 for the sake of clarity, but the patterns are similar for LU.A.64. Phases 1 and 2 are very demanding in terms of network utilization, and they take a sizeable portion of the total running time. In those phases there are dependency chains, of length 7, among messages. All messages in each chain go to distance 1 or 8. As LBR prioritizes the in-transit traffic, messages to distance 1 are injected after those going to distance 8 have passed trough, thus the total time for these phases is increased greatly. It is important to remark that when the problem size increases, both the message size and the number of chains in the cascades increase.

In phase 4 only neighbor-to-neighbor links are used, so network routers do not observe passing-by traffic. In this phase, dependency chains are of length 2. Phase 6 is similar to 4, having chains of length 2, but message destinations are at distance 8. This results in conflicts between in-transit traffic and injected traffic.

Congestion control techniques are very effective in accelerating phase 6; in fact, a micro-benchmark that reproduces only this phase reports gains using LBR over a 40%. However, they are harmful in phases 1 and 2—a micro-benchmark for this phase reports drops of a 20%. When the problem size of LU is increased, phase 6 does not experiment further acceleration, but phases 1 and 2 are longer in number of messages and, thus, the negative impact of congestion control is more noticeable. This explains the results of Fig. 3c, showing performance decreases for larger problem sizes.

The conclusion of this section is that a simple, burst-synchronized traffic model may not adequately describe the characteristics of any possible application, but can do so for some phases of the application. We need to improve the model to include the dependency chains that are present in actual applications and that interact negatively with congestion control mechanisms.

5 Conclusions

Synthetic workloads are useful during the initial design stages of an IN, as they allow exploring the network design space and providing initial performance estimates. For large networks, the synthetic model should take into account, at least, the self- throttling that real applications have implicit.

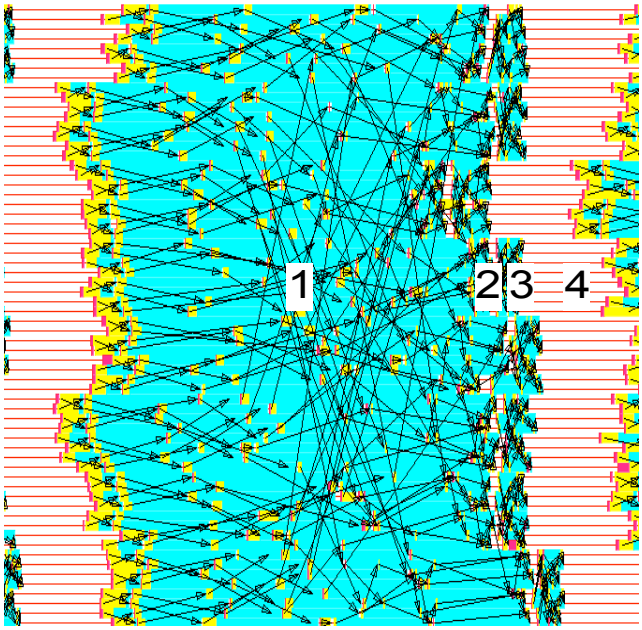


Fig. 4. Visualization of a portion of the traces for CG.A.64 (a): a yellow block (state) in the timeline represents a "send", pink means "receive", and cyan means "wait". The remaining is computation time. Arrows represent messages.

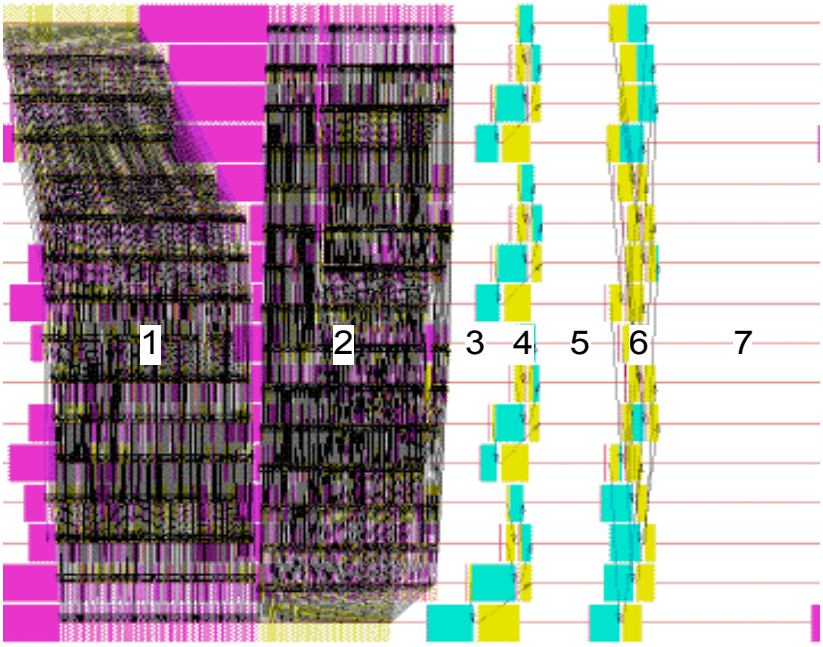


Fig. 5. Visualization of a portion of the LU.A.16 trace: in the last phase of data interchanges messages are sent to distance 4, whilst in the LU.A.64 this distance is 8. (yellow block (state) represents a "send", pink means "receive", and cyan means "wait". The remaining is computation time.

We propose the utilization of burst-synchronized traffic that, although still simple, models application behavior better than the traffic generated by independent sources. Scientific applications advance in alternating phases of computation and communication-synchronization. In our model, a burst represents a phase of intensive packet interchanges followed by synchronization primitives such as a barrier. The burst size b indicates the number of packets sent by each node in each phase.

We have used this traffic to evaluate a local congestion control mechanism (Local Buffer Restriction), after showing that the utilization of independent sources may provide misleading results. Using burst-synchronized traffic, LRB shows its potential to accelerate message interchange in a VCT network. This benefit is larger for larger values of b. In order to validate this result, we have performed additional experiments using real traffic, taken from traces of the NPB (class A, 64 nodes). For most of the experiments, congestion control shows its good performance, confirming our findings. Furthermore, the class (problem size) of the NPB is directly related to b and, again, larger problems benefit more from congestion control.

There are some exceptions, though, to this rule. For some applications, congestion control is counterproductive and, for some others, obtained results are not as good as we could expect, especially for large problems. This is because the applications, or some phases within them, have long chains of dependencies between messages. This behavior is not adequately characterized by burst-synchronized traffic, which models all traffic interchanges inside a phase as independent communication events.

In summary, this study has proven that the utilization of burst-synchronized traffic is a reasonable, although not perfect, alternative to the utilization of actual traffic, and can help in the evaluation of large networks for which the use of real traffic loads is not viable. Future lines of work include the introduction in this model of some sort of "reactiveness" to describe message chains, and also a method to find the best value of b that characterizes a given application.

References

[1] E. Baydal, P. Lopez and J. Duato, "A Family of Mechanisms for Congestion Control in Wormhole Networks" IEEE Trans. on Parallel and Distributed Systems, V. 16, N. 9, Sept. 2005, pp 772-784.

[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 Dec. 2003.

[3] T. Callahan and S.C. Goldstein, "NIFDY: A Low Overhead, High Throughput Network Interface", in Proc. 22nd Annual Int. Symp. on Computer Architecture (ISCA), Italy, June 1995.

[4] W.J. Dally, B. Towles. "Principles and Practices of Interconnection Networks". Morgan-Kaufmann, 2004.

[5] 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, v. 7, n. 8, 1996.

[6] C. Izu, J. Miguel-Alonso, J.A. Gregorio. "Evaluation of Interconnection Network Performance under Heavy Non-uniform Loads". Lecture Notes in Computer Science, Volume 3719 / 2005 (Proc. ICA3PP 2005), pp. 396 - 405.

[7] C. Izu, J. Miguel-Alonso, J.A. Gregorio. "Effects of Injection Pressure on Network Throughput", in Proc. PDP 2006 14th Euromicro Conference on Parallel, Distributed and Network based Processing. Montbéliard-Sochaux - France-February 15-17 2006.

[8] R. Jain. "Congestion control in computer networks: issues and trends". IEEE Network, v.4 n.3, May 1990.

[9] J. Miguel-Alonso, C. Izu, J.A. Gregorio. "Improving the Performance of Large Interconnection Networks using Congestion-Control Mechanisms". EHU-KAT-IK-06-05. Dep. of Computer Architecture and Technology, UPV/EHU. Submitted.

[10] NASA Advanced Supercomputing (NAS) division. "NAS Parallel Benchmarks" Available (May 2006) at http://www.nas.nasa.gov/Resources/Software/npb.html

[11] J. Navaridas, F.J. Ridruejo, J. Miguel-Alonso. "Evaluation of Interconnection Networks Using Full-System Simulators: Lessons Learned". Proc. 40th Annual Simulation Symposium, Norfolk, VA, 2007.

[12] V. Puente, C. Izu, J.A. Gregorio, R. Beivide, and F. Vallejo, "The Adaptive Bubble router", Journal on Parallel and Distributed Computing, vol 61, no. 9, Sept. 2001.

[13] F.J. Ridruejo, J. Miguel-Alonso. "INSEE: an Interconnection Network Simulation and Evaluation Environment". Lecture Notes in Computer Science, Volume 3648 / 2005 (Proc. Euro-Par 2005).

[14] Y.H. Song, T.M. Pinkston. "Distributed Resolution of Network Congestion and Potential Deadlock Using Reservation-Based Scheduling". IEEE Trans. Parallel and Distributed Systems, v.16, N.8, 2005.

[15] M. Thottethodi, A.R. Lebeck, S.S. Mukherjee. "Exploiting Global Knowledge to Achieve Self-Tuned Congestion Control for K-Ary N-Cube Networks". IEEE Trans. on Parallel and Distributed Systems, Vol. 15, No. 3, March 2004, pp 257-272.

[16] Virtutech Inc. "Simics page". Available (June 2006) at http://www.virtutech.se/products/

Acknowledgements

This research has been supported by the Spanish Ministerio de Educación y Ciencia, under grant TIN2004-07440-C02-01. Mr. Navaridas is supported by a pre-doctoral grant from the University of the Basque Country. We also acknowledge the Barcelona Supercomputing Center (BSC) for supplying computing resources for our research.