Impact of Adaptivity on the Behavior of Networks of Workstations under Bursty Traffic *

F. Silla, M. P. Malumbres and J. Duato
Dpto. Inf. de Sistemas y Computadores
Universidad Politécnica de Valencia
Camino de Vera, 14, 46071–Valencia, Spain
{fsilla,mperez,jduato}@gap.upv.es

D. Dai and D. K. Panda
Dept. of Computer and Information Science
Ohio State University, Columbus, OH 43210-1277
{dai,panda}@cis.ohio-state.edu

Abstract

Networks of workstations (NOWs) are becoming increasingly popular as an alternative to parallel computers. Typically, these networks present irregular topologies, providing the wiring flexibility, scalability, and incremental expansion capability required in this environment. Similar to the evolution of parallel computers, NOWs are also evolving from distributed memory to shared memory. However, distances between processors are longer in NOWs, leading to higher message latency and lower network bandwidth. Therefore, we can expect the network to be a bottleneck when executing some parallel applications on a NOW supporting a shared-memory programming paradigm.

In this paper we analyze whether the interconnection network in a NOW is able to efficiently handle the traffic generated in a DSM with the same number of processors. We evaluate the behavior of a NOW using application traces captured during the execution of several SPLASH2 applications on a DSM simulator. We show through simulation that the adaptive routing algorithm previously proposed by us almost eliminates network saturation due to its ability to support a higher sustained throughput. Therefore, adaptive routing becomes a key design issue to achieve similar performance in NOWs and tightly-coupled DSMs.

1. Introduction

Research in parallel computers has focused on multicomputers and multiprocessors during the last decades. Large-scale parallel computers evolved from multicomputers to distributed shared-memory multiprocessors, either with cache coherence or without cache coherence. The higher architectural complexity required to provide a single memory address space is worth its cost. It provides a simpler programming model in which communication is achieved by accessing shared memory locations, and synchronization is performed by using barriers. Additionally, sequential applications that require a large amount of memory to be executed efficiently can be directly ported to DSMs. In general, this is not possible in a multicomputer because each processor has a relatively small local memory.

Due to the increasing computation power of microprocessors and the high cost of parallel computers, networks of workstations (NOWs) are being considered as a cost-effective alternative for small-scale parallel computing. NOWs do not provide the computing capacity available in multicomputers and multiprocessors, but they meet the needs of a great variety of parallel computing problems at a lower cost. Moreover, the nature of NOWs makes them scalable and allows an incremental expansion of the system.

Recent network products, like Autonet [17], Myrinet [1], and ServerNet [11], use point-to-point links between switching elements instead of the traditional shared mediums (like Ethernet) used in computer networks. These NOWs usually present an irregular topology as a consequence of the needs in a local area network. Moreover, instead of being a direct network, NOWs are often arranged as switch-based interconnects, thus reducing the number of switches required for a given number of processors.

Research in NOWs is advancing relatively fast because the research effort made on parallel computers is now being transferred to this raising environment.

As a natural evolution from local area networks, most interconnects for NOWs only provide support for message-passing. These networks consist of a network interface card that is plugged into the I/O bus of each workstation, and one or more switches interconnecting the interface cards through point-to-point links [1]. In general, the bandwidth provided by currently available interface cards and switches is high enough for the requirements of message-passing ap-

* This work was supported by the Spanish CICYT under Grant TIC97–0897–C04–01

plications. However, the performance of some parallel applications is limited by message latency. Most of this latency is due to the software messaging layer. Several attempts have been made to reduce this bottleneck [8].

As parallel computers, NOWs are also evolving from distributed memory to shared memory. Some commercial interface cards support the shared-memory programming model. This is the case for the SCI-PCI adapter from Dolphin [5]. However, this card does not support cache coherence in hardware because memory traffic cannot be seen from the I/O bus. Also, interface cards are reaching the limits of I/O buses[1]. In order to provide a higher bandwidth and lower latency, some researchers have started to develop interface cards that are plugged into the memory bus. Some experimental NOWs provide support for shared-memory [16], also implementing a cache-coherence protocol that allows out-of-order delivery of messages.

2. Motivation

A decade ago, the invention of wormhole switching led to very fast interconnection networks. As a consequence, the interconnection network was no longer the bottleneck in a multicomputer. However, DSMs[2] require faster interconnection networks than multicomputers because messages are shorter (a typical message consists of a cache line and some control information) and much more frequent. While the interconnection network is not the bottleneck yet, some researchers began to report that a lower latency [21] and a higher network bandwidth [2] may significantly reduce the execution time of several parallel applications. As processor clock frequency is increasing at a faster rate than network bandwidth, the interconnection network may become a bottleneck within the next few years [7].

The situation is more critical for NOWs supporting a single memory address space. Physical distance between processors is higher in NOWs than in DSMs, leading to higher latency (due to the propagation delay) and lower bandwidth (due to the use of narrower links)[3]. Additionally, the use of irregular topologies makes routing and deadlock handling much more complex. Existing routing algorithms are either deterministic [1] or provide some degree of adaptivity [17]. They provide non-minimal paths and an unbalanced use of physical links. As a consequence channel utilization is low, reducing the effective network bandwidth even more. Therefore, we can expect the network to be a bottleneck when executing some parallel applications on a NOW supporting a shared-memory programming paradigm. Taking into account that NOWs are evolving from distributed memory to shared memory, it is important to analyze the behavior of parallel applications on these machines, also providing solutions to alleviate the bottleneck.

In this paper, we take on such a challenge. We analyze whether the interconnection network in a NOW is able to handle the traffic supported by the network in a DSM when executing some parallel applications. In particular, we evaluate different routing algorithms on NOWs with irregular topology, using traces from parallel applications. These traces were captured from the execution of SPLASH2 applications (Barnes-Hut, Water, Radix, and LU) on a DSM simulator with a hardware cache-coherence protocol. The main goal of this study is to determine whether the interconnection network in a NOW becomes a bottleneck when executing the same applications. Therefore, we fed the network in a simulated NOW with the same traces captured from the execution of those applications in a DSM simulator. We show that the slower interconnection network available in a NOW becomes a bottleneck, saturating during some periods of time. The use of adaptive routing algorithms, derived by using a methodology previously developed by us [20] helps to alleviate such bottlenecks.

The main contribution of this paper is a study of the network behavior in a NOW under the traffic generated by several shared-memory parallel benchmarks executed on a DSM simulator with the same processors. This study provided the following insights:

- Network traffic is bursty, as other studies showed. Shared-memory parallel applications are usually programmed by splitting computation into several steps. At the end of each step, processes update global variables and synchronize, leading to bursty traffic.
- Peak traffic saturates the network for all the applications we analyzed when using routing algorithms from commercial systems. The lower effective network bandwidth available in a NOW compared to a DSM leads to saturation. Therefore, execution time will be longer in a NOW, even if it uses the same processors, than in a DSM with the same number of nodes.
- Architectural improvements that increase channel utilization and throughput, like adaptive routing and virtual channels, considerably reduce the time duration when the network is saturated. Therefore, this architectural improvements allow NOWs to achieve performance similar to a DSM with the same number of processors.

Section 3 presents the trace mechanism used to evaluate interconnection networks. In Section 4, the performance of

[1] For example, link bandwidth in Myrinet is 160 Mbytes/s. ServerNet II [9] provides links with 1 Gigabit/s peak bandwidth. However, a 32-bit PCI bus running at 33 MHz only achieves a peak bandwidth of 133 Mbytes/s.

[2] Throughout this paper, the terminology 'DSM' is used to refer to tightly-coupled distributed shared memory systems like DASH, FLASH, SGI Origin, etc.

[3] For example, Cray T3E routers [19] use 14 data wires per link. ServerNet II [9] and Myrinet [1] use serial and 8-bit links, respectively.

different routing algorithms is evaluated using application traces. Finally, in Section 5 some conclusions are drawn.

3. A Simulation Model Based on Application Message Traces

In general, the interconnection network is not the bottleneck in current DSM systems. Therefore, techniques that improve network throughput are of little interest in this environment [21]. Nowadays, DSMs and NOWs are designed by using the same processors. The main architectural differences between these machines are the interconnection network and communication assisting circuitry such as the node controller. In this paper, we focus on the network. In a NOW, links are usually longer and narrower than in a DSM, therefore increasing latency and reducing bandwidth. However, the lower message latency achieved by the network in a DSM may have a very small impact on the execution time of parallel applications. In general, performance could be reasonably comparable in a DSM and a NOW when using the same processors, unless the interconnection network saturates. If it saturates, messages are queued in the injection queue at the source nodes, drastically increasing message latency and seriously degrading performance. The following question arises: Is the interconnection network the bottleneck in a NOW? Will the network in a NOW be able to handle the traffic supported by the network in a DSM without saturating? Note that only a fraction of the network bandwidth is used in a DSM during the execution of parallel applications [21]. Therefore, despite the lower effective bandwidth provided by a NOW, this bandwidth may be enough for the requirements of parallel applications.

To answer the above question, we gathered message traces at the node controller during the execution of several parallel applications on an execution-driven DSM simulator. Then, we fed the interconnection networks of a NOW with those traces in order to analyze if it can handle the same amount of traffic without saturating. In particular, we evaluated the performance of several routing algorithms for irregular networks by using message traces generated by an execution-driven simulation of several SPLASH2 benchmark applications.

The simulated hardware DSM system used an architecture similar to the FLASH machine [12]. The system had 64 nodes connected by an 8×8 mesh with a bandwidth of 400 MB/s/link and 15 ns per-hop delay. The processor in each node was assumed to be a 200 MHz single-issue microprocessor with a perfect instruction cache and a 128 KB 2-way set associative data cache with a line size of 64 bytes. The cache was assumed to operate in dual-port mode using write-back and write-allocate policies. The instruction latencies, issue rules, and memory interface were modeled based on the DLX design [10]. The memory bus was assumed to be 8 bytes wide. On a memory block access, the first word of the block was assumed to be returned in 30 processor cycles; the successive words follow in a pipelined fashion. The machine used a full-mapped, invalidation-based directory coherence protocol [13]. The node controller took 14 cycles to process an incoming (or outgoing) request or reply. The network interface took 15 cycles to construct and 8 cycles to dispatch a message. The synchronization protocol assumed was queuing based similar to the one used in DASH [14]. Our simulated architecture used the sequential memory consistency model. A cache miss stalls the processor until the critical word of data is returned.

Four applications Barnes, LU, Radix, and Water, all ported from the SPLASH2 suite [22] were used in this study. The problem sizes (listed in Table 1) for the applications were selected with two considerations. First, they are reasonably large to generate realistic network behavior. Second, the sizes of resulting trace files are manageable.

Application	Problem Size
Barnes-Hut	$8K$ particles, $\theta = 1.0$, 4 time steps
LU	512×512 doubles, 8×8 blocks,
Radix	$1M$ keys, $1K$ radix, max $1M$
Water	512 molecules, 4 time steps

Table 1. Applications and their input sizes.

Figure 1 shows the total number of messages injected into the network per time unit. Note that the plot shows the total number of messages generated by all the processors during each 500,000-cycle time interval. As can be seen in Figure 1(a), there are several bursts of traffic during the execution of the Barnes-Hut application. This bursty traffic may instantly saturate the network. If saturation occurs, messages are stored in an injection queue at the source node. Thus, depending on the ability of the network to process this kind of traffic, the overall execution time may be considerably affected. Note that the shared tree is completed at the end of each burst and the processors are not able to proceed until this phase finishes. Figure 1 also shows the traffic pattern of the Water, LU, and Radix applications. Bursty traffic can also be observed in these applications as well.

4. Performance Evaluation

In this section, we compare the performance of the up/down routing scheme (UD) [17] and our minimal adaptive (MA-2VC) routing algorithm [20]. The latter requires two virtual channels per physical channel. In order to make a fair comparison, we have also evaluated the up/down routing algorithm with two virtual channels (UD-2VC).

Our simulator models the network at the flit level. We used traces to drive the simulator. The performance mea-

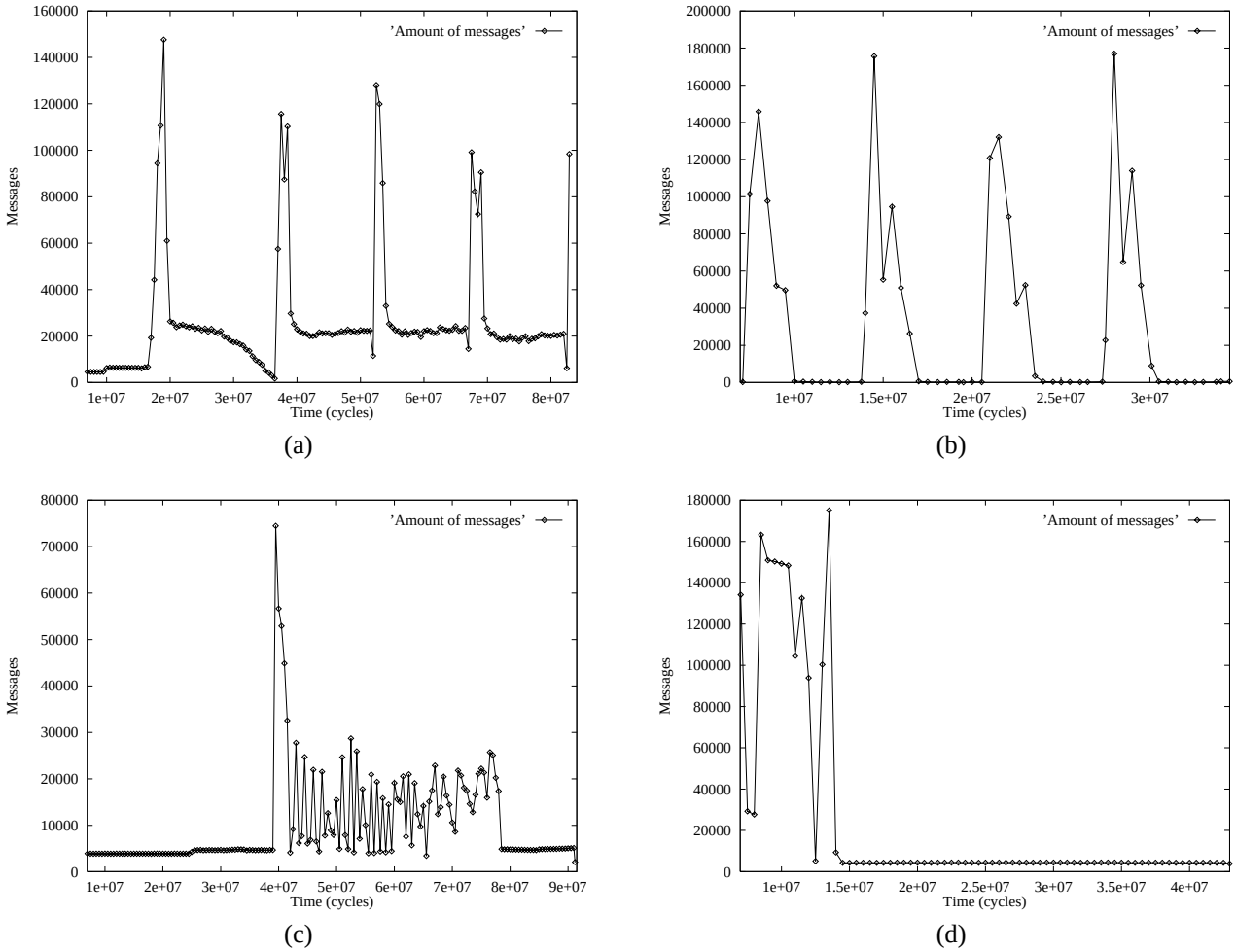

Figure 1. Number of messages injected during the execution time of (a) Barnes, (b) Water, (c) LU, and (d) Radix applications.

sure is message latency because total execution time is determined by the trace file. Message latency lasts since the message is queued at the network interface until its last flit is received at the destination node. It is measured in clock cycles. We obtained the average message latency for each set of 50,000 received messages during program execution. When traffic is intense, a more detailed analysis is required. Thus, when presenting simulation results we will make a zoom of the plot in the critical points by gathering measurements every 5,000 received messages.

As can be seen in Figure 1, there are time intervals in which the message injection rate increases considerably. At these points some messages would be lost if the queues were not deep enough. The time spent waiting on the source queue is also considered in the message latency. We consider the network becomes saturated when the average number of messages stored in these queues is higher than one.

With respect to the network model, the network is composed of a set of switches. Network topology is completely irregular was generated randomly. For the sake of simplicity, we assumed that there are exactly 4 processors connected to each switch. Also, two neighboring switches are connected by a single link. Finally, all the switches in the network have the same size. We assumed that each switch has 4 ports available to connect to other switches.

4.1. Simulation Results

Figure 2 shows the average message latency in each time interval versus time during the Barnes-Hut execution for the three routing algorithms (UD, UD-2VC, and MA-2VC) on a network with 64 processing elements and 16 switches. Each processor has one injection channel. As seen, message

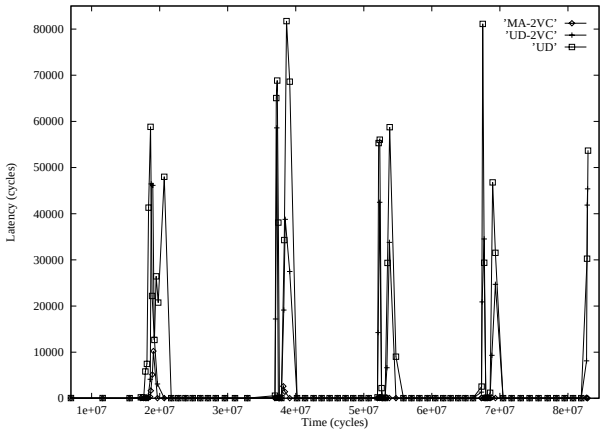

Figure 2. Average message latency versus time during Barnes-Hut execution.

latency remains very low for the three routing algorithms during the whole execution except for five short periods of time, where it increases considerably. Comparing this figure with Figure 1(a), it can be seen that these saturation points correspond to the bursty points mentioned in Section 3. Let us analyze saturation in more detail. The UD routing algorithm produces a very high latency at these points, as well as the UD-2VC routing scheme, which achieves a slightly lower latency. However, the MA-2VC routing algorithm reduces latency drastically in the first two bursty points and even avoids saturation during the remaining ones.

The scale in Figure 2 does not allow us to study the differences in latency when the interconnection network is not heavily loaded. Also, the behavior during the saturation points cannot be clearly seen. We could have used logarithmic scale in the Y axis in order to show these issues. However, after trying with logarithmic scale, we realized that they were difficult to study and, in any case, they did not clearly show the behavior of the routing algorithms during saturation. Therefore, the plots in Figure 3 are zooms that display all these issues properly. Note that in these plots, measures are taken every 5,000 received messages instead of every 50,000 messages as in Figure 2. The consequence is that sudden spikes in latency are not hidden because latency data represented in the plots are the average from a fewer amount of messages. Thus, these plots show the response of the interconnection network more accurately than the plot in Figure 2. This remark is also valid for the zooms of the other SPLASH2 applications shown in this paper.

Figure 3(a) — a detail of the bottom part of Figure 2 — shows the differences in latency when the network is not heavily loaded. As can be seen in this figure, the three routing schemes behave similarly for low loads, as it was expected after the study in [20]. The UD routing algorithm achieves the highest latency, as expected, while the lowest latency is achieved by the MA-2VC routing algorithm. However, this difference is not significant because the highest difference is about five clock cycles. It is interesting to compare these latencies with the theoretical base latency for wormhole in the absence of contention, which is about 50 cycles for the average distance between nodes and the average message length in the traces. The similarity between the theoretical base latency and the latency obtained in the simulations indicates that during these periods of time the network load is really low.

Figure 3(b) is a zoom of the first saturation point in Figure 2. It can be seen that the UD routing scheme needs seven times more time to process this traffic peak than the MA-2VC algorithm. Also, with respect to the UD-2VC algorithm, MA-2VC behaves more than four times faster. Figures 3(c) and 3(d) show similar results for the second and third bursty points. These results confirm the conclusions in [20], where the MA-2VC algorithm achieved a throughput several times greater than the UD routing algorithm using a uniform distribution of message destinations. However, differences in throughput are more noticeable when application traces are used. This result indicates that there are hot spots in the network due to repetitive communication patterns. In this situation, the MA-2VC routing algorithm behaves comparatively better than with uniform traffic, since it is a fully adaptive algorithm.

An issue that needs some specific study is why the bursty traffic spikes are so hard to handle by the different routing algorithms, specially the UD algorithm. From Figure 1(a), we can see that the injection rate never exceeds 200,000 messages per interval of 500,000 cycles, with 64 processors and 16 switches. This is only a rate of 1 msg/proc/160 cycles. Moreover, messages in a DSM machine are small (they are control messages and cache lines), so why the spikes are so difficult to handle?. We have analyzed the message length distribution in the trace file. Short messages (5 bytes) represent 61.42% of the messages, while long messages (69 bytes) account for 38.57% of the total number of messages. Therefore, the average message length is 29.68 bytes. Taking measures at 5,000 cycle intervals, the maximum amount of information injected into the network per interval is 96,861 bytes. This means an injection rate of 1.21 bytes/switch/cycle. However, simulations carried out with synthetic load and the message length distribution mentioned above show that a network consisting of 16 switches using the UD routing algorithm saturates at an injection rate of 0.37 bytes/switch/cycle. As a consequence, the network becomes completely saturated.

In the case of the other SPLASH2 applications, similar results were obtained when their traces were used to drive the simulator. Figures 4 and 5 show the simulation results

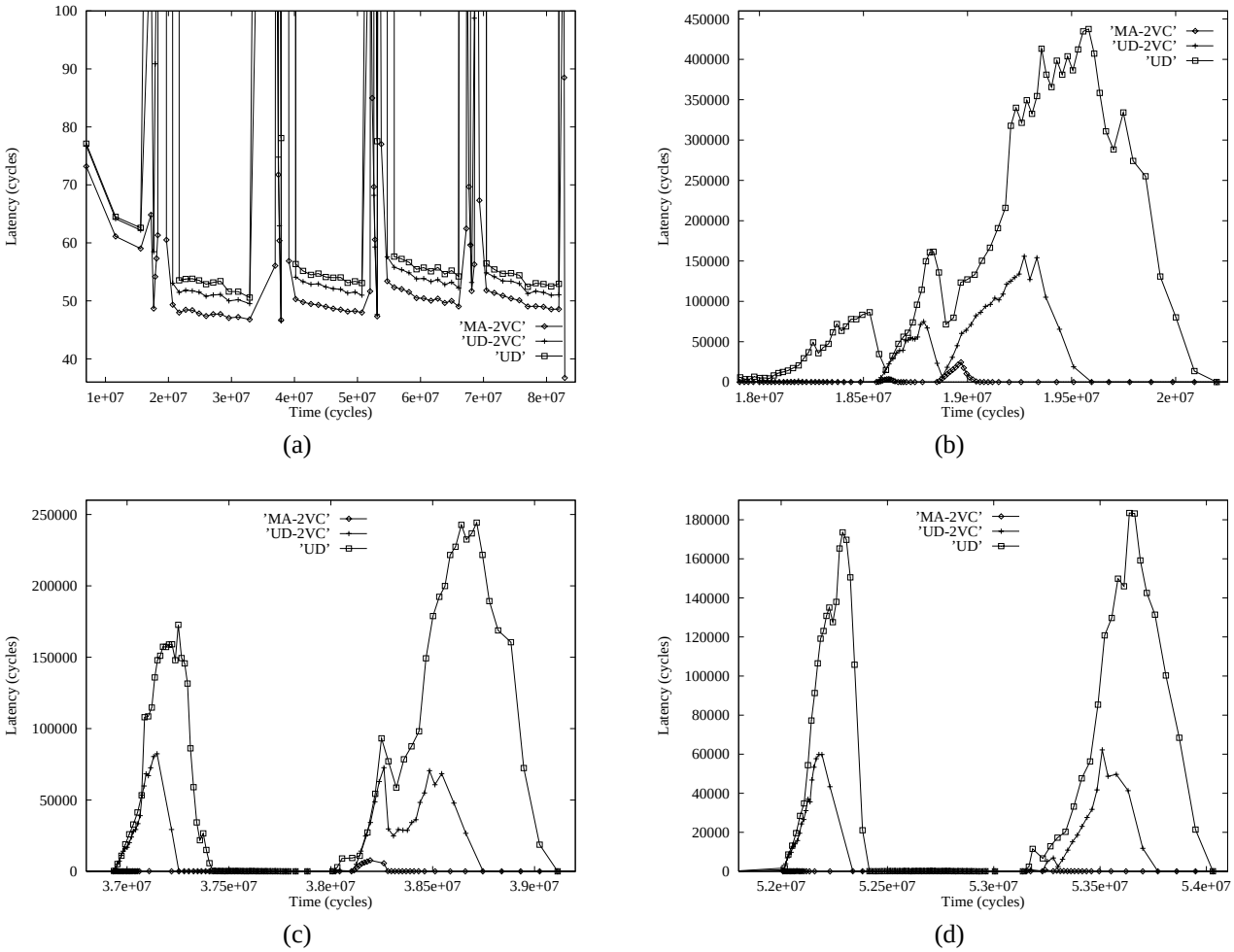

Figure 3. Different views of Figure 2.

for the Water and LU applications, respectively (results for Radix are not shown to save space). Two plots are shown for each application. The first one displays the overall application behavior, and the second one presents a detail of one of the bursty points. In general, it can be seen that the UD routing algorithm is not able to manage the total traffic generated by the processors in the saturation periods, reaching excessive latency values. On the other hand, the UD-2VC behaves better than UD, but the network also saturates. Thus, the use of virtual channels, by itself, does not solve the problem. However, the MA-2VC routing algorithm avoids the network saturation at these points for all the applications under study. Although the figures do not accurately show the behavior of MA-2VC at the bursty points, this routing algorithm never saturates the network for these three applications. In particular, at Figure 4(b), we can see the second bursty point, and the maximum latency values achieved by MA-2VC, UD-2VC, and UD are 60, 31,700, and 300,000 respectively. This shows the importance of using a minimal adaptive routing scheme.

These results are very important from an architectural point of view. As mentioned, NOWs are migrating from distributed memory to shared memory. This evolution needs faster interconnection networks. As shown above, current NOWs are not ready to support the traffic generated by shared-memory applications. The introduction of virtual channels and a minimal routing algorithm like the MA-2VC algorithm contribute to meet the requirements of NOWs.

5. Conclusions

Networks of workstations are becoming increasingly popular as a cost-effective alternative to parallel computers. Typically, these networks connect processors using irregular topologies [1, 11]. Irregularity provides the wiring flexibility, scalability and incremental expansion capabil-

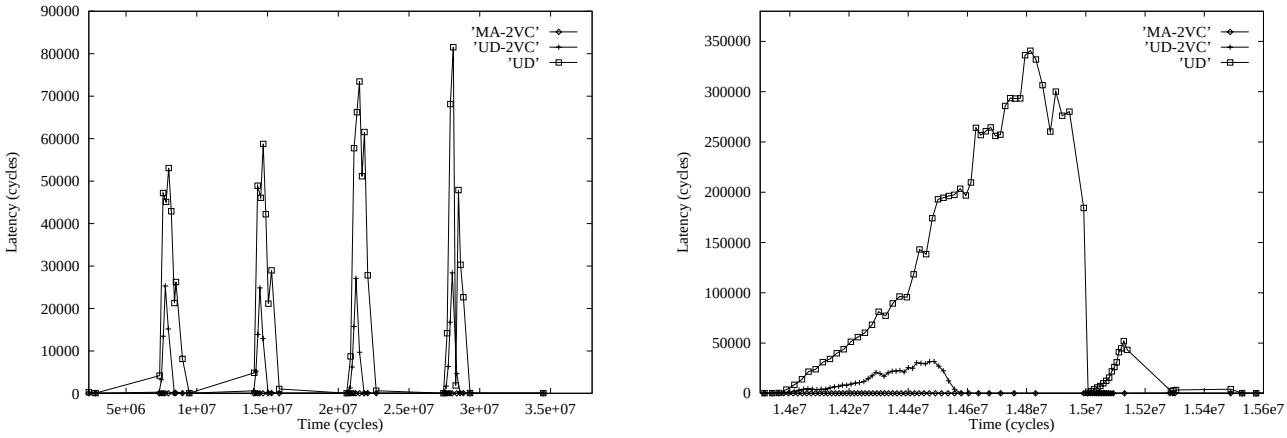

Figure 4. Average message latency versus time during Water execution.

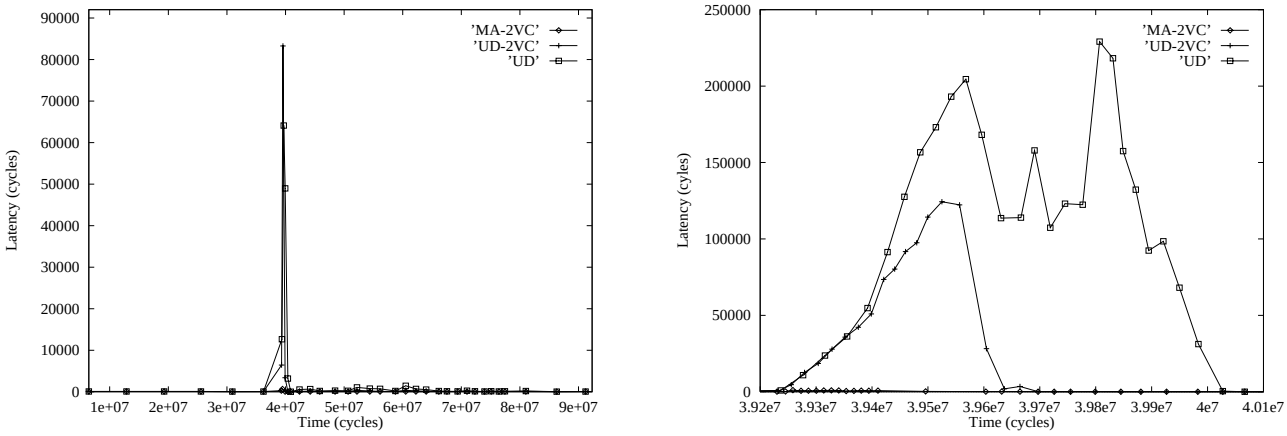

Figure 5. Average message latency versus time during LU execution.

ity required in this environment. Similar to the evolution of parallel computers, NOWs are also evolving from distributed memory to shared memory. Some commercial interface cards support the shared-memory programming model [5]. Some experimental NOWs provide support for shared-memory [16], also implementing a cache-coherence protocol that allows out-of-order delivery of messages.

In this paper we analyzed whether a NOW with irregular topology is able to handle the traffic supported by the interconnection network in a DSM with the same number of processors. This analysis has been performed by simulating the behavior of networks with irregular topology, using traces from parallel applications. These traces were captured from the execution of SPLASH2 applications (Barnes-Hut, Water, Radix, and LU) on a distributed shared-memory multiprocessor (DSM) simulator with 64 processors and a hardware cache-coherence protocol. We studied the network behavior using different routing algorithms, highlighting the application requirements with respect to the services provided by the network. This study provided the following insights:

- Network traffic is bursty, as already known.
- In the parallel applications we analyzed, peak traffic saturates the network for all the applications when using routing algorithms from commercial systems. Therefore, the lower effective bandwidth provided by the network in a NOW is not enough to achieve the same performance as in a DSM.
- Architectural improvements that increase channel utilization and throughput, like adaptive routing and

virtual channels, considerably reduce network contention and message latency. Therefore, these architectural improvements allow a NOW to achieve performance similar to that of a DSM with the same number of processors.

In summary, this paper shows that several parallel applications for shared-memory machines produce bursty traffic, and that our fully adaptive routing algorithm handles bursty traffic much more efficiently than the up/down routing algorithm with the same network parameters. Therefore, the use of fully adaptive routing algorithms and virtual channels may considerably improve performance when the applications tend to saturate the network at some point during the execution. As a consequence, the use of fully adaptive routing and virtual channels may considerably reduce the total application execution time in a NOW.

As for future work, we plan to develop a complete DSM testbed that includes our irregular network simulator, so that we could estimate the reduction of total application execution time when fully adaptive routing algorithms are used.

References

[1] N. J. Boden et al., "Myrinet - A gigabit per second local area network," *IEEE Micro*, Feb. 1995.

[2] G. T. Byrd et al., "Evaluation of communication mechanisms in invalidate-based shared memory multiprocessors," in *Proc. of the 1997 PCRCW*, June 1997.

[3] D. Dai and D. K. Panda. "Reducing Cache Invalidation Overheads in Wormhole DSMs Using Multidestination Message Passing, in *Int. Conf. on Parallel Processing*, Aug 1996.

[4] D. Dai and D. K. Panda, "How we can design better networks for DSM Systems?," in *Proceedings of the 1997 Parallel Computer Routing and Communication Workshop*, June 1997.

[5] Dolphin, *The Dolphin SCI Interconnect*, available at http://www.dolphinics.no.

[6] J. Duato, "On the design of deadlock-free adaptive routing algorithms for multicomputers: Design methodologies," in *Proc. of Parallel Architectures and Languages Europe 91*, June 1991.

[7] J. Duato, S. Yalamanchili and L. M. Ni, *Interconnection Networks: An Engineering Approach*. IEEE Computer Society Press, 1997.

[8] T. von Eicken et al., "Active messages: A mechanism for integrated communication and computation," in *Proc. of the 19th Int. Symp. on Computer Architecture*, June 1992.

[9] D. Garcia, "Servernet II," in *1997 Parallel Computer Routing and Communication Workshop*, June 1997.

[10] J. L. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, 1990.

[11] R. Horst, "ServerNet deadlock avoidance and fractahedral topologies," in *Proc. of the Int. Parallel Processing Symp.*, Apr. 1996.

[12] J. Kuskin et al, "The Stanford FLASH Multiprocessor, in *Proc. of the Int. Symp. on Computer Architecture*, 1994.

[13] D. Lenoski et al, "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, in *Proc. of the 17th Annual Symp. on Computer Architecture*, May 1990.

[14] D. Lenoski, et al., "The Stanford DASH multiprocessor," *IEEE Computer*, vol. 25, no. 3, March 1992.

[15] J. Miguel, et al., "Assessing the performance of the new IBM SP2 communication subsystem," Technical Report 96–06–01, Department of Electrical and Computer Engineering, University of California, Irvine, June 1996.

[16] A. G. Nowatzyk, et al., "S-Connect: From networks of workstations to supercomputer performance," *Proc. of the 22nd Int. Symp. on Comp. Architecture*, June 1995.

[17] M. D. Schroeder et al., "Autonet: A high-speed, self-configuring local area network using point-to-point links," Technical Report SRC research report 59, DEC, April 1990.

[18] S. L. Scott and G. Thorson, "Optimized routing in the Cray T3D," *Proc. of the Workshop on Parallel Computer Routing and Communication*, May 1994.

[19] S. L. Scott and G. Thorson, "The Cray T3E networks: adaptive routing in a high performance 3D torus," in *Proc. of Hot Interconnects IV*, Aug. 1996.

[20] F. Silla and J. Duato, "Improving the Efficiency of Adaptive Routing in Networks with Irregular Topology," in *1997 Int. Conf. on High Perf. Computing*, Dec. 1997.

[21] A. S. Vaidya et al., "Performance benefits of Virtual Channels and Adaptive Routing: An Application Driven Study," in *Int. Conf. on Supercomputing*, 1997

[22] S. C. Woo et al., "The SPLASH-2 Programs: Chracterization and Methodological Considerations, in *Int. Symp. on Computer Architecture*, 1995.