PARALLEL DISCRETE EVENT SIMULATION OF WORMHOLE ROUTING INTERCONNECTION NETWORKS

DIRK O. KECK
Institute of Communication Networks
and Computer Engineering (IND)
University of Stuttgart
Pfaffenwaldring 47, D-70569 Stuttgart, Germany
Tel.: (+49)-711-685 7964, Fax: (+49)-711-685 7983
keck@ind.uni-stuttgart.de

MICHAEL JURCZYK
Department of Computer Engineering and Computer Science
University of Missouri - Columbia
121 Engineering Building West
Columbia, Missouri 65211, USA
Tel.: (+1) 573-884-8869, Fax: (+1) 573-882-8318
mjurczyk@cecs.missouri.edu

ABSTRACT: *Simulation is inevitable for dimensioning interconnection networks for parallel computers. In this study, we present the application of a parallel event-driven simulator for k-ary 2-cube wormhole-routing interconnection networks running on the Intel Paragon and discuss simulator speedup behavior and influence of the operation modes of the simulated network (synchronous, asynchronous, and mixed mode) on simulator and network performance. With our simulator, similar speedups of up to 6.7 for 8 processing nodes have been achieved for the different modes. The simulator shows only a slight increase in execution time if problem (network) size and number of PEs are scaled up proportionally. Furthermore, it is shown that synchronous and asynchronous networks can be simulated most efficiently and realistically with a synchronous simulator, while the mixed-mode network operation cannot be simulated synchronously but has to be simulated in a mixed-mode to obtain accurate network performance results.*

KEYWORDS: k-ary 2-cube networks, network operation modes, parallel simulation, speedup, wormhole routing

1 INTRODUCTION

In many parallel processing systems, direct interconnection networks are used to interconnect the processing elements (PEs), or to connect the processors with memory modules [1, 2], e.g., Intel Paragon, Maspar MP-1, or the Cray T3E. Many implementation parameters have a major influence on network performance and must be evaluated by simulation and/or analysis prior to network implementation to achieve an effective system [3]. To keep the mathematical analysis tractable, simplified network and data traffic models have to be used that most often do not describe the real system very accurately. Therefore, simulation is often the only realistic approach to the network performance assessment. Even if the network topology is simple and highly regular, its size and the needed accuracy for the simulation results lead to high performance requirements for the executing simulator, creating a need for parallel simulation as a means to reduce turnaround times. Thus, the use of a massively parallel system (the Intel Paragon MIMD parallel computer [4]) for the discrete-event simulation of k-ary 2-cube direct networks is studied in this paper. A few parallel network simulators based on the discrete event simulation technique have been implemented on MIMD and SIMD machines, e.g., [5]. This work differs from those in the study of different network operation on the network and on the parallel simulator performance.

In the next section, the network model is presented. Section 3 discusses parallel simulation and simulator implementation aspects. The simulator speedup achievable on the Intel Paragon parallel machine is discussed in Section 4, together with a simulator performance comparison of the different network operation modes. In Section 5, the influence of the different network operation modes on the network performance is studied and conclusions on the use of the most efficient simulation modes are drawn.

2 INTERCONNECTION NETWORK MODEL

2.1 Network Topology and Routing

The network model assumed in this paper is a wormhole-routing k-ary 2-cube direct network (i.e., mesh and torus topologies) [6]. Each *node* of the parallel system consists of a processing element (processor/memory pair) and a router, as shown in Figure 1 for a two-dimensional mesh. The router handles all message transmissions (including message segmentation and routing decisions) among the nodes. In each direction, two neighboring nodes are connected via two unidirectional physical links (one in each direction). Because a node is not connected to all other nodes in a direct network, a message transfer from a source to a destination node might require several hops through intermediate routers to reach its destination node. In this model, it is assumed that if several hops are needed to transfer a message, the message will not be delivered to the intermediate nodes but will stay in the intermediate routers. Furthermore, wormhole-routing [7] is assumed in the network. Also, a deterministic dimension-ordered and an adaptive routing algorithm for meshes and tori are currently implemented in the proposed parallel network simulator.

The router architecture assumed in this paper is depicted in Figure 2. Flit buffers are present at each router input to temporarily buffer flits that cannot be transferred through the router due to busy output links. Also, *virtual channels* (*VC*) are present at each physical router port to increase the performance of wormhole-routing networks [8]. VCs allow a physical link to be shared among several messages. Thus, a blocked message can be overtaken by another message. If more than one message is able to be transmitted at the same time on the same physical link, they are usually time-multiplexed flit-by-flit on this link. Multiplexing leads to logically independent connections between two nodes, each of which requires its own flit buffering queue, as depicted in Figure 2. When a message is generated by a node it is stored in a message queue in the processor/router interface. The router then segments the message at the head of the queue into one or more flits (P→F in Figure 2) and routes it to its appropriate output port (if that port is able to accept flits).

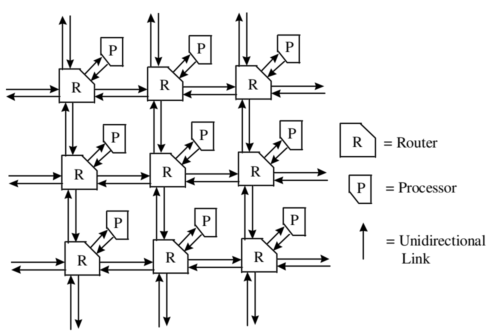

Figure 1: Part of a two-dimensional mesh network

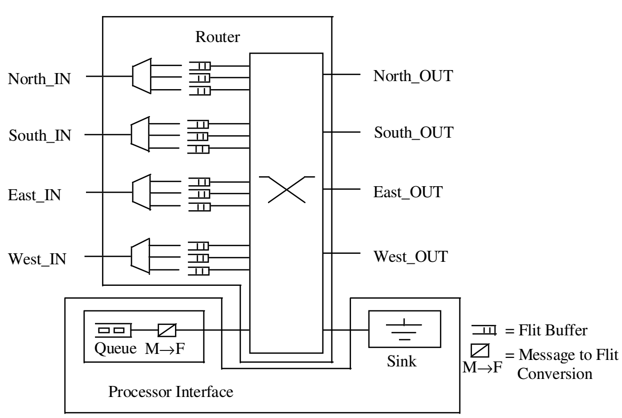

Figure 2: Router architecture

2.2 Network Operation Modes

In order to reflect various hardware implementation alternatives, three different network operation modes for the simulated network are considered: asynchronous, synchronous send - asynchronous receive (SSAR), and synchronous operation.

Asynchronous operation: In this mode, no clock – neither global for the whole system nor local on a node – controls flit transmission. Instead, flits are sent and received under a simple handshake protocol between neighboring nodes [7]. To simulate this mode, it is assumed that a router needs 0.9 simulator time units to route a flit, while the transmission time over a link is equal to 0.1 simulator time units [7].

Synchronous send - asynchronous receive (SSAR): This mixed-mode operation assumes an independent clock on each node of the system. Although the nominal values of the clocks are equal, they deviate slightly from each other, building a plesiochronous system (this mode was for example used in the design of an ATM switch [9]). The purpose of this mode is to provide a model for large MIMD machines, where distribution of a single clock to all nodes without phase differences is difficult or almost impossible. Flits are sent synchronously to the clock of the sending node, which is generated locally; they can be received by a neighboring node at arbitrary times. As routing and switching requires a certain amount of time, this model requires that each flit has to wait a minimum time in the receiving buffer until it can be forwarded. In the simulation results presented in Section 5, this time has been chosen as 0.8 simulator time-units [9]. The clock periods of all nodes are fixed (do not change with time) and follow a gaussian probability distribution function with a mean value of 1.0 and a relatively small standard deviation σ.

Synchronous operation: A global clock distributed with no phase difference to all nodes controls all routers in the network. Flits are transmitted and received only synchronously to this clock. Although the sources in each node produce messages at arbitrary times, they have to wait at least for the next clock tick to be sent to the next node. This operation is a special case of the SSAR mode with σ=0.

In order to allow a comparison of those operation modes, the peak bandwidth of the physical links interconnecting two neighboring node and the simulator time units are assumed to be the same in all modes. The time unit chosen is the clock period in the synchronous, the mean clock period in the SSAR, and the (minimal) time period to route a flit through a router and over a link (assuming empty flit buffers within a router) in the asynchronous case.

3 PARALLEL SIMULATION ASPECTS

3.1 Simulation Method

Parallel discrete event simulation methods usually are separated into two categories, termed *conservative* and *optimistic* methods. For an introduction into these techniques please refer to [10]. A simulated system consists of *physical communicating processes* (*PPs*), which exchange messages at discrete points in time. These PPs are mapped onto *logical communicating processes* (*LPs*) running on the PEs of the parallel machine, which exchange information with each other by the means of event messages containing a timestamp value determining the time of occurrence of this event. The simulation paradigm chosen in this work is the conservative approach presented in [11] using null messages to advance the local time of logical processes.

3.2 Implementation Aspects

Each of the LPs of the simulator represents one node of the simulated network. One or more of these LPs are simulated on one PE of the Intel Paragon. All messages between LPs are processed by a message passing function. If the destination LP is executed on the same PE as the source LP of a message, this message is sent using a local FIFO queue, otherwise it is sent using remote buffers. Each node of the interconnection network to be simulated is represented by its own LP. As there usually are more LPs than physical PEs in the machine executing the simulation program, several LPs have to be mapped onto one PE. The data of the LPs on one PE are organized as a linear array, with a flag indicating the state of each LP as "ready," "waiting," or "terminated." If an LP is ready, the next event is committed and the new logical process state is calculated; if an LP is in the waiting state, it is determined whether the awaited message has been arrived, and, in this case, the status is changed from waiting to ready.

4 SIMULATOR SPEEDUP

The simulator was implemented on the Intel Paragon XP/S MIMD machine with 72 PEs [4]. In Figure 3, the achievable speedup for the simulation of a 32×32 asynchronous mesh is shown. Speedup is defined as the quotient of the simulator execution time of a fixed-size network on $i > 1$ Paragon PEs and the execution time of the same network on two Paragon PEs (due to the restricted memory size on a single Paragon PE, the simulation of a 32×32 mesh on a single PE was impossible). As can be seen, the simulator exhibits a good speedup characteristic (e.g., 6.7 for 8 PEs). However, there is a significant speedup decrease (as compared to the ideal case) when more than eight PEs are used (i.e., when less than 32 nodes are simulated on an individual Paragon PE). Similar speedup results were obtained when simulating different network configurations and operation modes.

Another important performance measure of a parallel simulator is its problem-size scalability. Problem-size scalability is defined here as the performance behavior in the case of increasing the problem size and the parallel machine size proportionally (i.e., the number of nodes simulated on an individual PE stays constant). If, for example, the execution time of the simulation of an 8×8 mesh on one PE is known, this measure indicates the simulation time of a 32×32 mesh on 16 PEs or that of a 128×128 mesh on 256 PEs. In the ideal case, scaling the problem and machine size proportionally should result in the same simulation execution time. Figure 4 shows the normalized execution time (normalized to the execution time on one PE) vs. the number of PEs for four different scenarios: simulations with 16, 32, 64, and 128 simulated synchronous nodes per PE (simulations with different network configurations exhibited similar behavior). It can be seen that for simulations with at least 32 simulated nodes per PE, the parallel simulator shows a good scalability (up to 35% higher execution time as compared to a simulation on a single PE), where as for simulations with 16 nodes per PE and less, a factor of two and higher in execution time is observed. The fact that simulating at least 32 nodes per PE results in a good simulator performance was also seen in the speedup behavior in Figure 3.

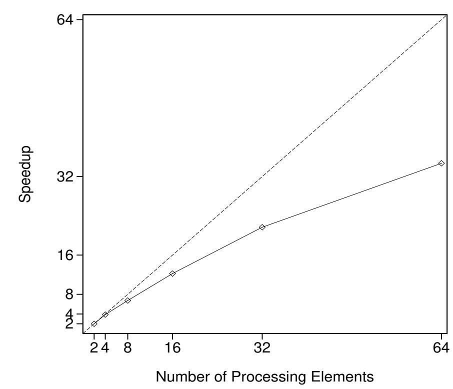

Figure 3: Simulator speedup for the simulation of an asynchronous 32×32 mesh

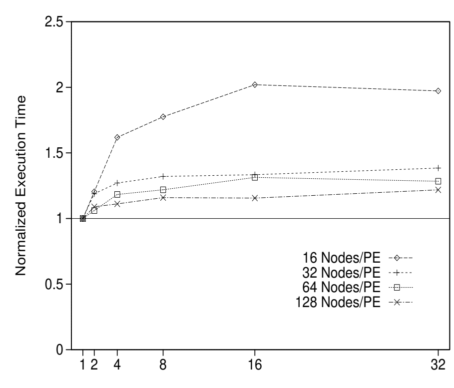

Figure 4: Execution time for simulations with a constant number of simulated nodes per Paragon PE

In an event-driven network simulation, the absolute simulator execution time depends on the data traffic to be simulated and on the network operation mode (assuming a fixed-size network). To study these dependencies, Table 1 depicts the absolute number of flits per hour and the absolute time-units per hour that could be achieved when simulating a 32×32 mesh under uniform traffic with varying traffic loads (20%, 40%, and 70%) on 16 Paragon PEs (in the SSAR case, a clock period deviation of 1% was assumed). In the synchronous and SSAR cases, because of the synchronous send, the simulator will step form time-unit to time-unit, which is fixed and independent of the data traffic. Therefore, the amount of time-units per hour stays roughly the same, while the number of flits per hour will increase with increasing traffic load. In the asynchronous case, the simulator will step from flit to flit. Thus, the time-unit per step depends on the inter-arrival time of flits and thus on the traffic load. So, the number of simulated flits per hour stays roughly the same, while the amount of simulated time-units decreases with increasing traffic load.

In most cases, the number of flits simulated is more important than the amount of time-units in a network simulation (e.g., to obtain flit latency or flit acceptance rate measures). It can be concluded from Table 1 that the synchronous operation mode results in a much higher number of simulated flits than the asynchronous mode (by a factor of 2.5 to 9), while the SSAR mode is in between. This is due to the fact that in the asynchronous mode, many more null messages have to be send to advance the simulation time as compared to the synchronous and SSAR modes (these null messages slow down the simulation and are not a part of the actual simulated data traffic). Thus, to obtain the fastest and most efficient network simulator, networks should be simulated synchronously. However, it is not obvious whether a synchronous simulator can be used to accurately simulate an asynchronous or an SSAR network. This will be discussed in the next section.

network operation	data traffic	flits/h simulated	time-units/h simulated
synchronous	20 % uniform	$5.0 * 10^6$	$202 * 10^3$
	40 % uniform	$9.2 * 10^6$	$200 * 10^3$
	70 % uniform	$13.9 * 10^6$	$182 * 10^3$
SSAR	20 % uniform	$2.8 * 10^6$	$111 * 10^3$
	40 % uniform	$5.2 * 10^6$	$109 * 10^3$
	70 % uniform	$8.0 * 10^6$	$104 * 10^3$
asynchronous	20 % uniform	$2.0 * 10^6$	$52 * 10^3$
	40 % uniform	$2.0 * 10^6$	$34 * 10^3$
	70 % uniform	$1.5 * 10^6$	$21 * 10^3$

Table 1: Performance of simulators for different network operation modes for a 32×32 mesh simulated on 16 PEs

5 NETWORK PERFORMANCE RESULTS

Two important performance measures for wormhole-routing networks are the message latency and the acceptance rate. The *message latency* is defined as the time between the generation of a message in the source node and the arrival of the tail flit of that message at the destination node. The *message acceptance rate* is the ratio of the number of messages accepted into the source queues to the number of generated messages (messages can only be lost, if they cannot be buffered in the source queue because that queue is full). In this section, the following simulation scenario is assumed: a 32×32 mesh network with XY-routing, one or four virtual channels per router port, message size of 20 flits, flit buffer length of four flits, source queue length of one message, and uniformly distributed message destinations. A synchronous, an asynchronous, and SSAR networks with a clock period deviation of 0.1%, 1.0%, 10%, and 20% are considered. These values have been chosen larger than in real systems, where they usually are in the PPM range, in order to achieve a significant drift of the clock phases in less simulation time, without having the effect of reduced throughput due to nodes with significantly smaller clock frequencies (the 10% and 20% cases are unrealistic but serve as trend indicators).

Figure 5 depicts the mean message latency vs. traffic load for the network with one virtual channel per router port (Figure 5a) and four virtual channels per router port (Figure 5b). The message acceptance rate vs. the traffic load is depicted for the network with one virtual channel per router port in Figure 6a) and with four virtual channels per router port in Figure 6b). It can be seen that the synchronous and asynchronous networks exhibit roughly the same behavior, especially in the network with four virtual channels. The message latency and acceptance rate in SSAR networks differ from the ones in the synchronous and asynchronous cases even for very low clock period variations. Simulations with different network and traffic parameters showed similar results. Thus, an asynchronous network can be realistically simulated using a (fast) synchronous simulator that is preferable over the slower asynchronous simulator (see Section 4). However, SSAR networks cannot be realistically simulated with a synchronous simulator. In this case, a slower SSAR simulator has to be used to obtain most accurate network performance results. The implementation of a parallel discrete event driven simulator is in most cases much more complicated than the implementation of a parallel time-driven simulator [12]. This study shows that an asynchronous network can be realistically simulated using a synchronous simulator that could even be implemented in a time-driven fashion.

6 CONCLUSION

A study of the parallel discrete-event simulation of k-ary 2-cube networks was performed in this paper. Simulator speedup behavior and the influence of the network operation modes (synchronous, asynchronous, and mixed mode) on simulator and network performance

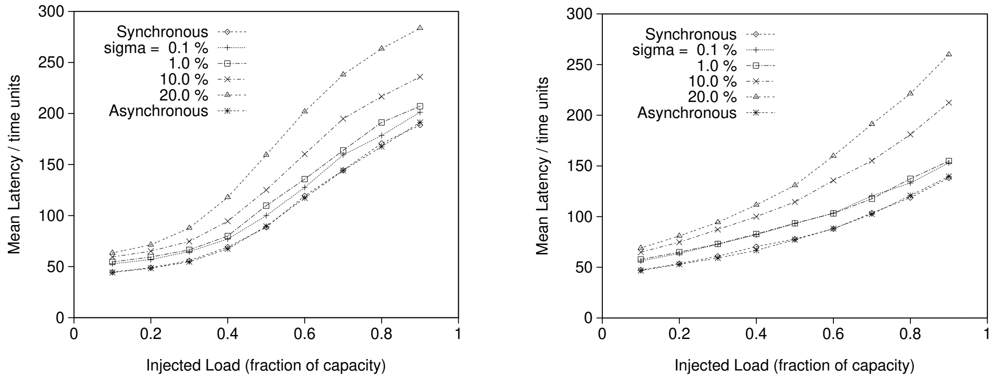

Figure 5: Mean message latency for a 32×32 mesh with (a) one virtual channel and (b) four virtual channels per router port

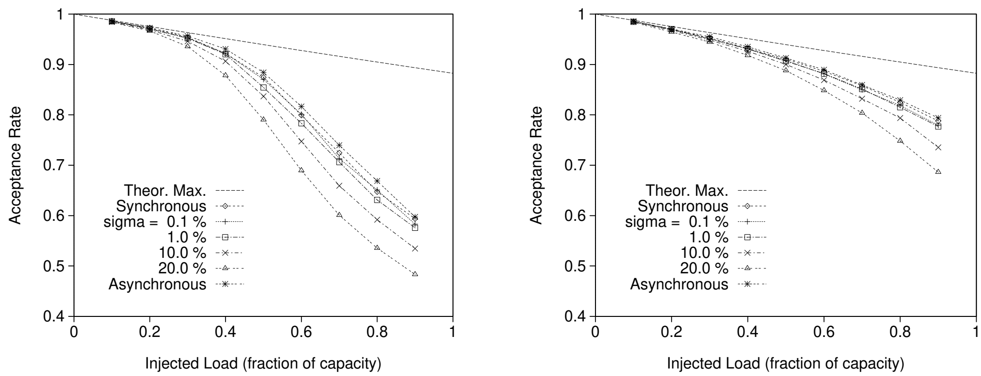

Figure 6: Message acceptance rate for a 32×32 mesh with (a) one virtual channel and (b) four virtual channels per router port

were studied. With our simulator, similar speedups of up to 6.7 for 8 PEs of the Intel Paragon have been achieved for the different network operation modes. The simulator shows only a slight increase in execution time if problem (network) size and number of PEs are scaled up proportionally, as long as 32 or more nodes are simulated on each Paragon PE. Furthermore, it is shown that the synchronous operation mode results in a much faster simulator than the asynchronous mode (by a factor of 2.5 to 9 in the cases shown), while the mixed-mode operation is in between. The network performance study in this paper showed that an asynchronous network can be realistically simulated using a (fast) synchronous simulator that is preferable over the slower asynchronous simulator. However, mixed-mode networks cannot be realistically simulated with a synchronous simulator. In this case, a slower mixed-mode simulator (that is still faster than the asynchronous simulator) has to be used to obtain most accurate network performance results.

REFERENCES

[1] M. Jurczyk and T. Schwederski, SIMD processing: Concepts and systems in *Handbook of Parallel and Distributed Computing*, A. Y. Zomaya, ed., (New York: McGraw-Hill, 1996).

[2] M. Jurczyk, H. J. Siegel, and C. Stunkel, Interconnection Networks for Parallel Computers in *Encyclopedia of Electrical and Electronics Engineering* , J. G. Webster, ed., (New York: John Wiley and Sons, to be published, 1999).

[3] M. Jurczyk and T. Schwederski, Phenomenon of higher order head-of-line blocking in multistage interconnection networks under nonuniform traffic patterns, *IEICE Trans. on Information and Systems.*, E79-D(8), 1996, 1124-1129.

[4] R. Esser and R. Knecht, Intel Paragon XP/S - Architecture and software environment, *Supercomputer*, 1993, 121-141.

[5] B. Gaujal, A. G. Greenberg, and D. M. Nicol, A sweep algorithm for massively parallel simulation of circuit-switched networks, *J. of Parallel and Distr. Computing*, 18(4), 1993, 484-500.

[6] W. J. Dally, Performance analysis of k-ary n-cube interconnection networks, *IEEE Trans. on Computers*, C-39(6), 1990, 775-785.

[7] L. M. Ni and P. K. McKinley, A survey of wormhole routing techniques in direct networks, *IEEE Computer*, 26(2), 1993, 62-76.

[8] W. J. Dally, Virtual-channel flow control, *IEEE Trans. on Parallel and Distr. Systems*, 3(2), 1992, 194-205.

[9] K. Sakaue et al., A 0.8-um BiCMOS ATM switch on an 800-Mb/s asynchronous buffered banyan network, *IEEE J. of Solid-State Circuits*, 26(8), 1991, 1133-1144.

[10] R. M. Fujimoto, Parallel discrete event simulation, *Communications of the ACM*, 33(10), 1990, 31-53.

[11] K. M. Chandy and J. Misra, Distributed simulation: A case study in design and verification of distributed programs, *IEEE Trans. Software Eng.*, SE-5(5), 1979, 440-452.

[12] M. Jurczyk, T. Schwederski, H. J. Siegel, S. Abraham, and R. Born., Strategies for the implementation of interconnection network simulators on parallel computers, *Int. J. of Comp. System Science & Eng.*, 13(1), 1998, 5-16.