

XMulator: A listener-Based Integrated Simulation Platform for Interconnection Networks

A. Nayebi, S. Meraji, A. Shamaei, H. Sarbazi-Azad

IPM School of Computer Science & Sharif University of Technology

Tehran, Iran

{nayebi, meraji, a_shamaei}@ce.sharif.edu, azad@ipm.ir

ABSTRACT

Simulation is perhaps the most cost-effective tool to evaluate the operation of a system under design. A flexible, easy to extend, fully object-oriented, and multilayered simulator for interconnection networks can be a very useful tool for multicomputer designers and researchers. It is so desirable to attach newly designed components to the existing models and to exploit detailed results. This paper presents XMulator, an object-oriented listener-based simulation environment for evaluating multicomputer interconnection networks. The simulator involves a toolbox of various network topologies, routing algorithms, switching techniques, and flexible router models. This work introduces a simulator, using listener-based integration methodology, which has a great impact on extensibility of the system. Mixed-mode event processing improves the performance of the simulator. By decoupling individual parts of the code, XMulator enables independent code development and creates a flexible and extensible environment for different aspects of network design. This simulator uses XML format to define network topologies, input parameters, and outputs reports providing a high level of flexibility. To the best of the authors' knowledge, it is the first simulator enabled to simulate any arbitrary interconnection topology under different working conditions including in the presence of faults.

1. Introduction

Message-passing multicomputers have emerged as a cost-effective platform for exploiting parallelism in applications. Microprocessors are becoming increasingly interconnected. Clusters of computers and server blades connected by interconnection networks are widely deployed in server farms today. Emerging microprocessor systems will have to interface with an interconnection network fabric.

There are three main approaches for performance evaluation of systems including interconnection networks. The first one is to monitoring the actual system, that can capture the effects of low-level design choices but this

restricts experimentation with different router policies since it can be prohibitively expensive and time-consuming to change these features. Mathematical modeling is the second approach for performance evaluation of interconnection networks. It provides a cost-effective way to explore design issues but such models often impose simplifying assumptions that degrade the accuracy of the evaluation. The last and perhaps the best way is a simulation environment, which can provide an extensible framework for evaluating multicomputer networks. Simulation can provide more features than other two methods. Using a simulator, one can implement different routing and switching algorithms, different flow control policies, and interconnection topologies and obtain performance metrics with high accuracy. The only problem with this method is the time consumption of simulation experiments.

Much effort has been done to develop interconnection networks simulators [1, 2, 6, 11, 14, 19]. Several recent simulation tools evaluate various aspects of multicomputer applications and interconnection networks. Execution-driven simulators [7, 10, 18] typically capture the instruction-level operation of applications on particular multicomputer architectures. Other simulation tools emphasize multicomputer network architectures, allowing users to vary the router's buffer architecture, switching scheme, and routing algorithm, under different synthetic traffic patterns [9, 13, 15]. *Flexsim* [20] is a special purpose simulator for some topologies of interconnection networks which can operate with many predefined switching and routing schemes but it is not easy to extend its model and reportable results are not very detailed and on demand. *CinSim* [19] is another interconnection simulator enabled to model and simulate all kinds of interconnection networks that are composed of switches and buffers. Steady-state simulation can be applied as well as terminating simulation to observe any transient behavior of the network. Bedichek in [1] introduces *Talisman* a simulator that models the execution semantics and timing of a multicomputer. *Talisman* is unique in combining high semantic accuracy, high timing accuracy, portability, and good performance but it is not flexible to model different interconnection networks and evaluate their performance. *PP-MESS SIM* is an object-oriented discrete-event simulator which is introduced in [16] by Rexford, Dolter, and Feng. This simulator provides a toolbox of various network topologies,

communication workloads, routing-switching algorithms, and router models but it has low extensibility.

In this paper, we introduce a new simulator with many desirable properties named as *XMulator* (XML + Simulator). *XMulator*¹ is a flexible simulation framework for performance evaluation of multicomputer networks implemented in C# language. It uses XML format for defining topologies, parameters, and outputs that is very flexible and user friendly and provides high level of interoperability. By enforcing strict boundaries between different components of a network (using object-oriented design), *XMulator* facilitates multi-factor experiments that independently explore network design issues. Using listener-based integration in company with multi-layered architecture enables *XMulator* to be used for various systems from interconnection networks to logic circuits. Developing new packages or new tools for existing packages as well as defining routing algorithms is accomplished easily due to its multi-level architecture and object-oriented design. Moreover, using *log4net* [21] gives a great logging capability to *XMulator* which enables the user to trace in step by step fashion and monitor the movement of each flit in the network and to log all the parameters of each component such as channels, routers, message generators, etc.

The rest of the paper is organized as follows. In section 2, we introduce the basic structure of *XMulator*. Section 3 discusses the basic component layer of *XMulator*. The components, which are essential to model interconnection networks, are introduced in section 4, and in section 5 introduces some sample networks and routing algorithms implemented by *XMulator*. Finally, section 6 concludes the paper and mentions some possible future works.

2. Basic Structure

XMulator is developed using modern paradigms in software architecture. Some of the most important features are listed below.

2.1. Listener-Based Integration

The method applied for integration of components in a component-based simulator has a great impact on extensibility and maintainability of a simulator [4, 19]. Listener-based integration solves the problem of two-way dependency, which leads to a non-extensible design². For example, consider a Queue component developed by developer *A* and a Dispatcher dequeues the objects from the queue and developed by developer *B*. Each one must know the design of the other one because Queue must provide a link to dispatcher and awake it when a new object is ready to be dispatched, and on the other hand, Dispatcher must know architecture of Queue to dispatch

the objects from it. In listener-based integration, Queue provides an event, say *onObjectReceived*, generally to everyone who may be interested in this event. Designer of the Queue does not know about the architecture of the other components that would potentially use it later. On the other hand, Dispatcher knows the Queue and simply registers a method on the list of listeners of the event that is called every time a new object is received by Queue.

As mentioned above, listener-based integration can change a two-way dependency to a one-way dependency. Defining useful events, which might potentially be used later by other components, increase the ease of development of future applications.

Using listener-based integration makes *XMulator* a mixed event type simulator. There are two types of events: XEvents and programming language events. The earlier is used as an essential part of event-based simulation. These events are used to advance the virtual time of simulation process. The latter ones are used to connect adjacent components and build a framework to handle inter-component communications and dependencies. The latter events are supported by the development platform and are of higher performance. Hence, using a mixed event simulation paradigm increases the performance of simulation.

2.2. Multi-Layered Architecture

XMulator uses a multi-layered architecture [4] to be coherent with the modern software architecture paradigms. Each layer provides services to the upper layer(s) and only depends on the lower layer(s). A two-way dependency between layers is not a good practice. Using listener-based integration enables the designer to build a multi-layered architecture freely. The lowest layer consists of general tools, the second layer consists of base components, and third layer is the main system layer. Fourth layer may consist of graphical classes and user interfaces or program execution controllers.

3. Base Components Layer

Generally speaking, we have some events, basic components, and a simulation engine in this layer. In other words, the basic components that are essential for a simple simulation are gathered in this layer. The engine holds and manages a queue of XEvents. To follow the best practices of simulation architectures, the engine is designed so lightly.

The main role of the engine is processing of XEvents sorted by the fire time. Execution of a XEvent may generate some further new XEvents. The lighter the engine structure is, the more modular and more extensible the architecture could be developed. The engine does not need to know the XEvent types and their internal processes. It is sufficient to know which event belongs to which component and when it must be fired. This layer

¹ Simulator program is accessible at www.xumulator.org

² Indeed, listener-based integration is based on the observer pattern [24].

consists of some important components such as XObject, XEvent, and XEngine.

The queue of events is sorted according to the firing time. The data structure to save this queue is so important and has a great impact on the performance of simulator. Arrays and linked lists have lower performance than AVL trees. In this simulator, we use the red-black tree [22], which is a special case of an AVL tree. Each data node of the tree contains a queue of events that have the same firing time.

Another component which is the parent of all simulated components, is XObject. XObject class has an unique ID field that helps identification of each individual component during the debugging, logging, and generating the results. Another important class of this layer that is used to fire events and advance simulation virtual time is XEvent. Each XObject instance can fire many types of XEvents that are unknown to the engine. Buffer class is another basic component that holds a FCFS queue of objects. Buffers could be accommodated with probes, which can provide detailed statistical information about buffer states. In other words, a buffer is modeled as a Markov chain [23].

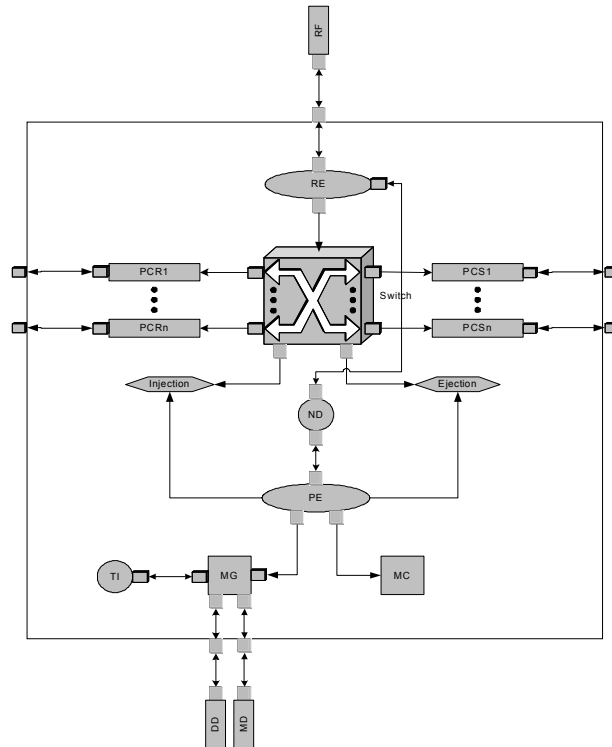
MetaComponent class is a XObject that acts as a container for other XObjects and other MetaComponents. Hierarchical designs can be implemented using MetaComponent in arbitrary levels. An efficient linkage mechanism is an essential part of an implementation of a

hierarchical design. That is, two high level MetaComponents must be connected together easily and efficiently. *XMulator* harnesses a caching mechanism to enhance components linkage performance.

4. Interconnection Package

The components essential to model interconnection networks are defined in this package. Fig. 1 depicts a MetaComponent represents a typical architecture of a multicomputer node. This MetaComponent has several ports to connect to the other nodes and to the shard components. In this sample implementation, DestinationDistribution, MessageLengthDistribution, and RoutingFunction are shared between the nodes. Adjacent components are connected through PhysicalChannelReceiverPort and PhysicalChannelSenderPort that interconnect two adjacent nodes. A brief explanation of major components is presented below.

InterconnectionBuffer: This component is derived from the Buffer class from base components layer. Size of interconnection buffers could be set through input parameters to any arbitrary length. Setting the buffer length to 1 induces a wormhole switching scheme and setting a greater value induces virtual cut-through switching mechanism [8].



PCR: PhysicalChannelReceiver	PCS: PhysicalChannelSender	MG: MessageGenerator
TI: TimeIntervalGenerator	MD: MessageLengthDistribution	DD: DestinationDistribution
RF: RoutingFunction	MC: MessageConsumer	ND: Node

Fig. 1. Connection of different parts of a node in *Xmulator*

BaseMessage: A message is recognized with an ID. It keeps generation time, length in flits, and an array of contained flits. Each flit can hold the current component, which is currently stored in. This feature enables full trace of message paths.

PhysicalChannelSender & PhysicalChannelReceiver: Each physical channel splits into two parts: a sender and a receiver. Hence, a physical channel is constructed by connecting these two parts.

Switch: Each switch has some sender and receiver physical channels. Switch component provides detailed statistics about the average number of used virtual channels, average number of virtual channels in blocking queue, etc.

InjectionChannel & EjectionChannel: *XMulator* has different strategies for implementation of injection and ejection channels. These channels can be implemented as a normal physical channel with limited bandwidth with arbitrary virtual channels or as a buffer without bandwidth limit.

VCArbiter: This component defines the strategy of sharing the physical link between the virtual channels. Currently, round robin is implemented.

Processing Element (PE): This component performs the message processing operations, message generation, and message consumption in a node. MessageConsumerProbe is a component that provides detailed statistical information of message generation and consumption.

Routing Element (RE): RE performs routing operations. RoutingFunction base class has a method named as *Route()* that must be overridden for different routing algorithms. RE has a queue of blocked messages that is checked in a Round Robin fashion and every time that a message is eligible to route, exits from the blocked

messages queue.

NetworkGeneration package is a helper package for integration of components in a node or building a network.

5. Sample Implemented Networks

XMulator can be used to implement different types of regular and semi-regular interconnection network topologies such as meshes, hypercubes, k -ary n -cubes, and some complex networks such as product networks. An irregular network package is also implemented which uses an adjacency matrix to build the network based on any arbitrary topology with irregular structure. Different routing algorithms, e.g. X-Y deterministic routing [17], planar partially adaptive routing [5], f -cube fault-tolerant routing, BC fault-tolerant routing [8], and Duato's fully adaptive routing are also implemented and tested. Some comparison with other simulators has been made to get more confidence. Fig. 2 shows the comparison of six different fault-tolerant routing algorithms implemented in a 10×10 mesh [24].

We also have developed a Queuing Network package. Fig. 3 depicts a report relevant to modeling of virtual channel multiplexing by queuing theory [12]. A Markov model package is also implemented upon the base components layer.

We also have modeled Duato's fully adaptive routing algorithm [8] according to Boura's model [3]. Simulation and analytical results are depicted in Fig. 4. As can be seen in this figure, simulation results are highly accurate and match the analytical model.

6. Conclusions

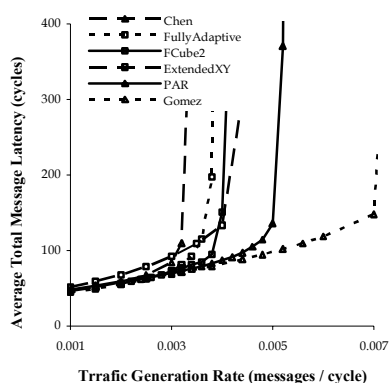


Fig. 2. The average message latency for six different fault-tolerant routing algorithms in a 10×10 mesh [24].

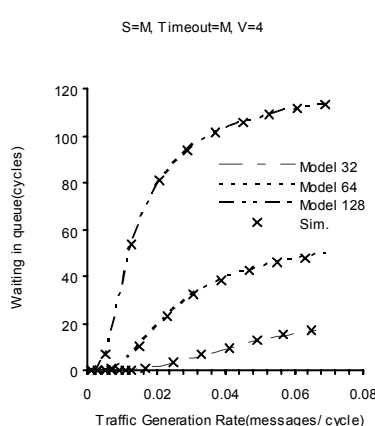


Fig. 3. Waiting time in queue versus generation rate, with timeout equal to message length, when four virtual channels are used, and message length is equal to 32, 64, and 128 flits [12].

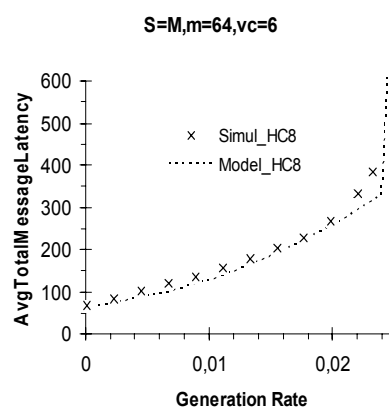


Fig. 4. Comparison of model and simulation results for an 8-dim hypercube with 6 virtual channels per physical channel and message length of 64 flits (generation rate is in message per cycle per node).

In this paper, we introduced *XMulator* as an object-oriented listener-based simulation environment for evaluating multicomputer networks. The simulator involves a complete toolbox of various network topologies, routing algorithms, switching methods, and flexible router models. *XMulator* is enabled to model and simulate all kinds of regular or irregular interconnection networks. Well-defined interfaces between the simulator components create an extensible environment that enables independent enhancements to the code. In this work, we focused on listener-based integration that has a great impact on extensibility of the system. Another feature of this simulator is mixed-mode event processing which improves the performance of the simulator. *XMulator* extensively uses XML format for defining topologies, parameters, and outputs that increases flexibility to deal with different types of inputs and different forms of results.

7. References

- [1] R.C. Bedichek, "Talisman: Fast and Accurate Multicomputer Simulation," Proc. ACM SIGMETRICS/Performance, pp. 14-24, May 1995.
- [2] K. Bolding, S.-E. Choi, and M. Fulgham, "The Chaos Router Simulator." Presentation at Parallel Computer Routing and Comm. Workshop, May 1994.
- [3] Y. Boura, C.R. Das, T.M. Jacob, "A performance model for adaptive routing in hypercubes," Proc. Int. Workshop Parallel Processing, pp. 11-16, 1994.
- [4] I. Crnkovic, S. Larsson, J. Stafford, "Component-Based Software Engineering: Building systems from Components," at 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems, Oxford, UK, August 2002.
- [5] A. Chien and J. H. Kim, "Planar-adaptive routing: Low-cost adaptive networks for multiprocessors," Proceedings of the 19th International Symposium on Computer Architecture, pp. 268-277, May 1992.
- [6] P.M. Dickens, P. Heidelberger, and D.M. Nicol, "Parallelized Network Simulators for Message-Passing Parallel Programs," Proc. Int'l Workshop Modeling, Analysis, Simulation of Computer and Telecommunication Systems, pp. 72-76, 1995.
- [7] J. Dolter, "A Programmable Routing Controller Supporting Multi-Mode Routing and Switching in Distributed Real-Time Systems," PhD thesis, Univ. of Michigan, Sept. 1993.
- [8] J. Duato, S. Yalamanchili & L. Ni, "Interconnection Networks: An Engineering Approach", IEEE Computer Society Press, 2003.
- [9] G. Ewing, Pawlikowski, K. McNickle, "Exploiting Network Computing by Distributing Stochastic Simulation," In Proceedings of the European Simulation Multiconference (ESM99); Warsaw, SCS, 175-181.
- [10] J.-M. Hsu and P. Banerjee, "Performance Measurement and Trace Driven Simulation of Parallel CAD and Numeric Applications on a Hypercube Multicomputer," IEEE Trans. Parallel and Distributed Systems, vol. 3, pp. 451-464, July 1992.
- [11] J.R. Jump and S. Lakshmanamurthy, "NETSIM: A General-Purpose Interconnection Network Simulator," Proc. Int'l Workshop Modeling, Analysis, Simulation of Computer and Telecommunication Systems, pp. 121-125, Jan. 1993.
- [12] A. Khonsari, M. Ould-Khaoua, A. Nayebi, H. Sarbazi-azad, "The Impacts of Timing Constraints on Virtual Channels Multiplexing in Interconnect Networks," Proc. of IPCC, 2006.
- [13] A.M. Law and M.G. McComas, "Simulation Software for Communications Networks: The State of the Art," IEEE Comm., pp. 44-50, Mar. 1994.
- [14] P.K. McKinley and C. Trefftz, "MultiSim: A Simulation Tool for the Study of Large-Scale Multiprocessors," Proc. Int'l Workshop Modeling, Analysis, Simulation of Computer and Telecommunications Systems, pp. 57-62, Jan. 1993.
- [15] E. Olk, "PARSE: Simulation of Message Passing Communication Networks," Proc. Simulation Symp., pp. 115-1245, Apr. 1994.
- [16] J. Rexford, J. Dolter, W. Feng, and K.G. Shin, "PP-MESS-SIM: A Simulator for Evaluating Multicomputer Interconnection Networks," Proc. Simulation Symp., pp. 84-93, Apr. 1995.
- [17] H. Sullivan and T. R. Bashkow, "A large scale, homogeneous, fully distributed parallel machine," Proceedings of the 4th International Symposium on Computer Architecture, March 1977.
- [18] D. Tutsch, M. Brenner, 2003, iMINSimulate "A Multistage Interconnection Network Simulator," In 17th European Simulation Multiconference: Foundations for Successful Modelling & Simulation (ESM'03); Nottingham, SCS, 211-216.
- [19] D. Tutsch,, D. Ludtke, A. Walter, and M. Kuhm, CINSim, "A Component-based Interconnection network Simulator for Modeling Dynamic Reconfiguration," 19th European conference on modeling and simulation, pp. 32-39, June 2005.
- [20] The Superior Multiprocessor ARchiTecture (SMART) Interconnects Group, Electrical engineering - Systems Department, University of Southern California. FlexSim at <http://www.usc.edu/dept/ceng/pinkston/tools.html>.
- [21] "log4net: A Tool to Help the Programmer Output Log Statements to a Variety of Output Targets," <http://logging.apache.org/log4net/>
- [22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, "Introduction to Algorithms," Second ed., McGraw-Hill, 2001.
- [23] S. M. Ross, "Introduction to Probability Models," 6th ed., Academic press, 1997.
- [24] A. Shamaei, "Performance Evaluation of Fault-Tolerant Routing Algorithms in Mesh Interconnection Networks," M.Sc. thesis, Sharif Uni. of Tech., 2006.
- [25] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns, Elements of Reusable Object-Oriented Software," Springer-Verlag, July 1993.