Discrete Event Simulation of Optical Switch Matrix Performance in Computer Networks

N. Imam and S. W. Poole

Abstract— **In this paper, we present application of a Discrete Event Simulator (DES) for performance modeling of optical switching devices in computer networks. Network simulators are valuable tools in situations where one cannot investigate the system directly. This situation may arise if the system under study does not exist yet or the cost of studying the system directly is prohibitive. Most available network simulators are based on the paradigm of discrete-event-based simulation. As computer networks become increasingly larger and more complex, sophisticated DES tool chains have become available for both commercial and academic research. Some well-known simulators are NS2, NS3, OPNET, and OMNEST. For this research, we have applied OMNEST for the purpose of simulating multi-wavelength performance of optical switch matrices in computer interconnection networks. Our results suggest that the application of DES to computer interconnection networks provides valuable insight in device performance and aids in topology and system optimization.**

Index Terms—**discrete event simulator (DES), distributed computing, high performance computing (HPC), optical switches, performance modeling.**

I. Introduction

THE use of simulation is becoming increasingly prevalent in the networking research community [1]. Network simulators allow one to model/analyze/evaluate an arbitrary computer network by specifying both the behavior of the computer hosts/nodes and the communication channels. A computer simulation model for a system is a representation of the system in terms of a set of states, events, and behavior functions. A discrete event-driven simulation is defined if the system's state transitions are based on discrete event activities. The DES maintains an event queue organized by the event execution schedule and successively processes the events in the queue [2]. The major network simulators available today all follow the discrete event paradigm. Examples of well known DE simulators are Network Simulator 2 NS2, Network Simulator 3 NS3, OPNET, and OMNEST [3]-[5]. These simulators are widely employed in network research and are considered to be important tools complementary to the analytical and experimental studies for investigating and understanding the behavior of complex computer networks.

The next generation of high performance computing systems will be distributed systems with hundreds of thousands of processing nodes interconnected via large and complex interconnection networks. In the design and development of such new systems, performance modeling of interconnection networks is as important as the modeling of processor performance. In this paper, we present the modeling of computer interconnection networks based on the Benes design. The Benes network has been studied for decades as a typical example of a rearrangeably non-blocking network [6]-[7]. In the next section, we present the reader with some background material related to our network simulator of choice, OMNEST, and optical switches. In Section 3, we discuss our simulation framework and implementation details. Section 4 presents simulation results and discussion. Our approach and results should be of considerable interest to the computer network community in general and the network simulation/modeling community in particular.

Manuscript received June 17, 2013; revised August 13, 2013. This work was supported by the United States Department of Defense and used resources of the Extreme Scale Systems Center at Oak Ridge National Laboratory. Oak Ridge National Laboratory is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725 for the U.S. Department of Energy.

N. Imam is with the Computer Science and Mathematics Division of Oak Ridge National Laboratory, Oak Ridge, TN 37831 USA (phone: 865-574-8701; email: imamn@ornl.gov).

S. W. Poole is with the Computer Science and Mathematics Division of Oak Ridge National Laboratory, Oak Ridge, TN 37831 USA (email: spoole@ornl.gov)

II. Technical Background

A. Overview of the Network Simulator OMNEST

OMNEST is the commercial version of OMNeT++, a C++ based discrete event simulator developed at the Technical University of Budapest by Andras Varga [8]. Although OMNEST can be used for a wide variety of applications, its primary application area is the simulation of computer/communication networks and other distributed systems. OMNEST has a modular structure and allows the creation of models composed of hierarchically nested modules. In OMNEST, the user works with simple and compound modules. The simple modules are at the bottom of the hierarchy level and implement the activity/behavior of the system. Several simple modules are combined to form compound modules. Inter-module communication happens by two way message passing that may contain timestamps and complex data about the sender/receiver modules. The user defines the higher level configuration and module interconnection of the discrete event model by using the topology description language (NED) of OMNEST. The atomic elements of the model are the simple modules implemented in C++. These simple modules are constructed of algorithms written as C++ functions, using the simulation class library. OMNEST has a powerful simulation class library to represent a wide variety of simulation objects as C++ classes. These classes include messages, data

ISBN: 978-988-19253-1-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

collection, statistics and probability distributions, and routing protocols. OMNEST also supports parallel distributed DES. OMNEST's object-oriented modular programming approach is suitable for flexible extension and scalability of the software for simulating very large scale systems such as HPC networks and interconnects.

B. Optical Switches and Benes Networks

The prediction of network performance and reliability is becoming progressively more difficult due to the fusion of different technologies, the exploding growth of the internet, and the hybrid nature of HPC platforms. The role of the interconnection networks in computer systems has become increasingly more important since performance in multiprocessor systems is highly dependent on communication processes between distributed processors, I/O devices, and processors and memory. Therefore, choosing the right interconnection network is important for efficiency and optimum throughput. Interconnection networks can be categorized according to topology with two main topologies being fixed and reconfigurable. Reconfigurable connections allow for dynamic configurations to match individual tasks thereby optimizing overall system performance. Several reconfigurable

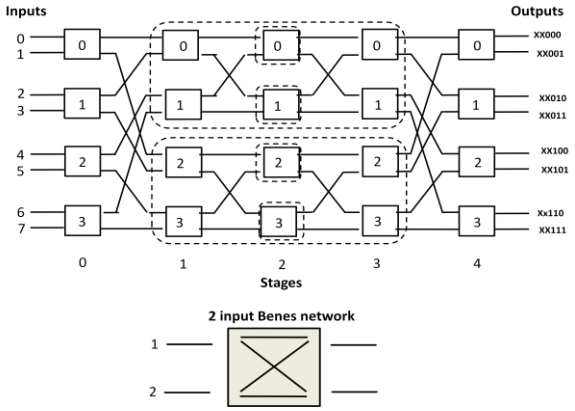

Fig. 1. An 8X8 Benes network constructed from five stages.

connection schemes exist such as crossbar switch connections and multistage interconnection networks (MINs). The Benes network is an example of a widely used multistage interconnection network. As a multistage interconnection network, a Benes network can scale with additions of 2X2 switch elements. An NXN Benes network is constructed by $2\log_2 N - 1$ stages and N/2 switch elements per each stage, so a total of $N(\log_2 N - 1/2)$ switch elements are required. Under the recursive configuration, an NXN Benes network consists of two N/2XN/2 Benes networks and two outer stages. The Benes permutation network is broadly adopted for photonic space division switching. In Figure 1, we show an N = 8 Benes network consisting of 2X2 basic switching elements. The basic switch is shown at the bottom of the figure and illustrates that each input port can be connected to both output ports via either the bar or the cross state connection.

A switching fabric constructed as a Benes network can be utilized in a switching node of optical networks. In optical packet switching networks, packet contentions and communication latency may occur when packets at different input ports need to reach the same destination output ports within the same timeslot. Processors will suffer frequent idle time due to inefficient data movement if the interconnection network cannot minimize its message latency. The use of multiple wavelengths may be useful in resolving packet contentions and increasing the capacity and connectivity of the network.

III. SIMULATION FRAMEWORK AND IMPLEMENTATION

This section describes our framework for implementing Benes 4X4 and Benes 8X8 switching fabrics in OMNEST. The OMNEST model for the Benes network consists of both compound and simple modules that communicate with each other via message passing. From a hierarchical perspective, the simple modules are nested within the top level Benes network modules and are grouped together to form compound modules. The active modules are also simple modules that are written in C++ and use the object classes contained in the powerful simulation class library within OMNEST.

A. Simulation Model

Our Benes network discrete event simulation model consists of the following parts:

- Network Description (NED) language topology description files: The NED language produces *.ned* files that describe the various module structures with parameters and gates and create the network topology.
- Simple modules sources: These are *C++* files that define the behavior of the modules. The typical functions in these simple modules are used to specify the initialization behavior, the message processing (such as message forwarding and deletion) procedures, simulation termination criteria, and statistics collection.
- Message definitions: These are *.msg* files which describe the message structures and may contain time stamp and node identity.

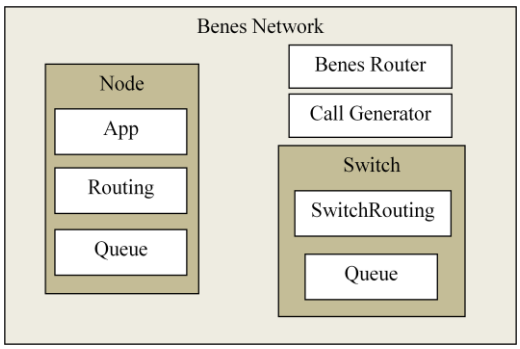

Fig. 2. Benes network model module relationship.

Table 1 shows the files that comprise our Benes network project and the relationship of the simple NED files, compound NED files, and simple module sources. Figure 2 shows the relationships within the model between simple

ISBN: 978-988-19253-1-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE 1
BENES NETWORK MODEL FILES AND THEIR RELATIONSHIPS

NED File	Compound File	Simple Module Source File	Message File
BenesNetwork.ned	BenesNetwork.ned	---	---
BenesRouter.ned	BenesNetwork.ned	BenesRouter.cc	---
CallGenerator.ned	BenesNetwork.ned	CallGenerator.cc	---
Node.ned	BenesNetwork.ned		
App.ned	Node.ned	App.cc	Packet.msg
Routing.ned	Node.ned	Routing.cc	---
Queue.ned	Node.ned	Queue.cc	---
Switch.ned	BenesNetwork.ned		---
SwitchRouting.ned	Switch.ned	SwitchRouting.cc	---

and compound modules, module interfaces, networks, channels, and channel interfaces. The functionality of each of the modules is described in further detail below.

B. Model Subcomponents

1) Benes Network Compound Module

Benes Network is the compound module that encompasses the sub-modules Node, Switch, Benes Router and Call Generator. While Benes Router and Call Generator are simple modules, Node and Switch are themselves compound modules. Figures 3 and 4 are screen-shots of the OMNEST Eclipse GUI showing the 4X4 and 8X8 Benes network simulations.

2) Node Compound Module

The nodes within the network are defined by the Node.ned network description file. The node definition within the model is a compound module. The node is comprised of the following simple modules: App, Queue, and Routing. The hierarchical structure of the Node compound module is shown in Figure 2.

App Simple Module: Our App simple module is defined by the App.ned network description file and the App.cc simple module source file. The App module is responsible for traffic generation in the network. The App simple module is responsible for generating the packet that is defined by the Packet.msg message definition file. Defined within the App simple module are the local node address, the destination addresses of the packet, the packet length, and the packet generation interval parameters. When a call has to be instantiated by a specific node, the ScheduleCall() function is invoked by the global Call Generator module. The scheduleCall() function in turn calls handleMessage() function of the App module with a "generatePacket" command to generate the packets with the packet parameters and activities defined in App simple module.

Routing Simple Module: The Routing simple module, which is defined by the Routing.ned network description file and the Routing.cc simple module source file, queries the Benes Router to check availability of a route to the destination of the call. If a route is available, the Benes Router returns the route bits (route_bits) to be used for the call and the wavelength (λ) value. This information is embedded in the packet to be transmitted by the router. If the "no route available" message is returned by the Benes Router, the call is blocked.

Queue Simple Module: The Queue simple module, which is defined by the Queue.ned network description file and the Queue.cc simple module source file, is responsible for queuing up packets to be sent out on the network as well as for their transmission. Incoming packets are held in the queue while the previous packet transmission is completed. Queue parameters such as the queue capacity and the queue service time are defined within the Queue simple module. These parameters are important in determining the overall throughput of the system.

3) Switch Compound Module

The crossbar switches in the network are defined by the Switch.ned network description file. The switch definition within the model is a compound module. The switch is comprised of the following simple modules: Queue, and Switch Routing. The hierarchical structure of the switch module is shown in Figure 2.

Switch Routing Simple Module: The switch routing module is contained in the switch and is different from the Routing module described earlier (contained in the Node). The Switch Routing simple module, which is defined by the SwitchRouting.ned network description file and the SwitchRouting.cc simple module source file, checks the route_bits contained in the packet that was received. Based on the bit for the current level, it chooses between upper and lower connections out to the next level. For example, for route_bits 011, a second level switch will route the packet in the lower connection out to the next level (discussed in detail in subsection III.5).

4) Benes Router and Call Generator Simple Modules:

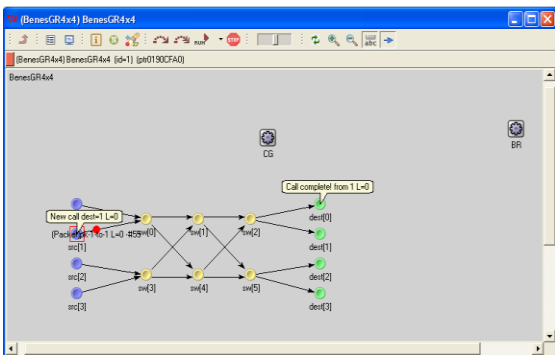

Fig. 3. Screen shot of a 4X4 Benes network in OMNEST graphical user interface.

These two simple modules implement incoming call generation functions and the global routing algorithm of the network. The Benes Router is responsible for checking the availability of a route from a given source to a given destination and allocating the route and the wavelength channel. The route discoverer checks the route_bit of every switch in the route to check if it has already been allocated. If the switch's particular channel has already been allocated the route cannot be used. Thus it has to check for an alternate route. If no alternate route is available in the current wavelength, then an alternate wavelength is

ISBN: 978-988-19253-1-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

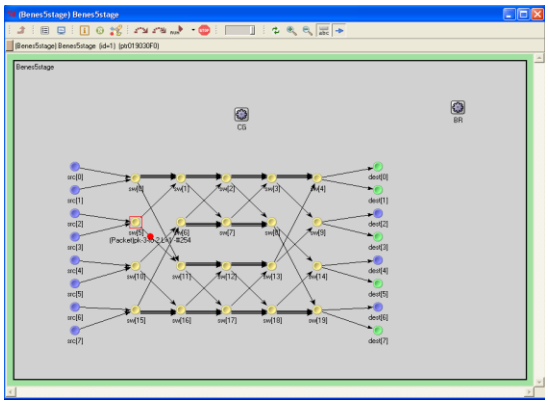

Fig. 4. Screen shot of an 8X8 Benes network in OMNEST graphical user interface.

assigned.

5) Routing Algorithm

The routing algorithm for a 4X4 Benes network is illustrated in Figure 5 below. The black dots represent the "nodes" (sources and destinations) and the white dots represent the switches. The nodes on the left are the sender nodes and the nodes on the right side of the switch matrix are the destination nodes. The destination node IDs are numbered 0 to 3. As discussed previously, these nodes are connected to the input and output ports of the switches. The switches have two input ports and two output ports. The connection that takes to the upper port is always numbered 0 and the connection that takes to the lower port is always numbered 1. In the diagram of Figure 5, the dashed connections represent 0 (upper) and the solid connections represent 1(lower).

The binary representation of the destination node ID is used in the routing algorithm. Destination node 0 is represented as x00. Where "x" denotes "don't care" bit i.e., it can be either 0 or 1. Each bit represents the route that needs to be taken at each level. Benes 4X4 network consists of 3 stages of switches. The bits of the destination ID is used to make the routing decision at each level. For example, to reach destination 0 (x00) from source 3, the routes are 100 and 000 (since the most significant bit is a don't care bit). At level 0, either the upper connection (0) or the lower connection (1) could be used. At level 2 and level 3 switches, the packet can only be routed via the upper connections (0).

6) Using Multiple Wavelengths

To improve throughput, we implement a multiple wavelength switch network. In our multiple wavelength network, we have four available wavelengths (λ_1, λ_2, λ_3, λ_4). The algorithm looks for a route at the lowest wavelength λ_1 from source to destination. If a route is not available in λ_1, a route is checked in the next wavelength λ_2 and so on until all available wavelengths are exhausted. It should be noted that for our implementation, it is not necessary to assign specific values to the wavelengths in the array. Suppose a network had only wavelength λ_1 available to it in order to make all the requested connections. The first two connections were established using λ_1. A third connection is requested but is blocked due to switch contention. In this case, the wavelength used for the third connection is changed to λ_2 and the device is allowed to be shared by the two signals of wavelengths λ_1 and λ_2. The call is allowed to go through and the probability of call blockage is reduced.

The routing algorithm is the same for the 8X8 network with differences in the number of rows, number of levels, and the numbers of source and destination nodes. An 8X8, 5-stage Benes network has 4 rows, 8 source and 8 destination nodes. The number of route_bits is equal to the number of levels in the Benes network. The number of don't care bits in the route_bits = number of route_bits – $\log_2$(number of destination nodes). Thus for the 8X8, 5-stage Benes network, the number of don't care bits in the route_bits = 5 –$\log_2$(8) i.e., the first 2 bits of the 5 route_bits are don't care bits. Therefore, during the first two levels, either the upper or the lower route can be taken. This is an important consideration in the routing algorithm to find alternate routes when a route is blocked. For example, in a five-stage (8X8) Benes network, the route to destination node 1 is represented as xx001. In a three-stage (4X4) Benes network, the route will be represented as x01.

7) Call Generation

The Call Generator simple module is responsible for generating calls at user specified intervals between a source and a destination node. The source and destination nodes for the calls are chosen from the parameters *sources* and *destinations* listed in the OMNEST initialization file (omnet.ini). If the *random address* flag is set in the omnet.ini file, the Call Generator chooses random source and destination nodes and invokes the *scheduleCall* function on the source node's "app" (application) object. If the *random address* flag is set to "false" the sources and destinations will be chosen in the order listed in the "omnet.ini" file. For example, sources = "0 0 1 2" and destinations = "0 1 2 3" will result in calls (s=0, d=0), (s=0, d=1), (s=1, d=2), (s=2, d=3), etc. For the results presented in this paper, the sources and destinations were chosen at random. The call inter-arrival time was exponentially distributed.

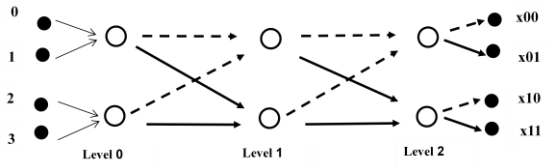

Fig. 5. The routing algorithm in a 4X4 network.

IV. RESULTS AND DISCUSSION

In Figure 6, we present the discrete event simulation results for a 4X4 Benes network with four wavelengths available to reduce call blocking probability. The calls were generated at random. The call generator chose a source and destination pair at random. The arrival time between the calls were exponentially distributed. Each call routing was initially attempted using λ_1. If call blockage occurred then the next available wavelength was used. However, wavelength switching between different

ISBN: 978-988-19253-1-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

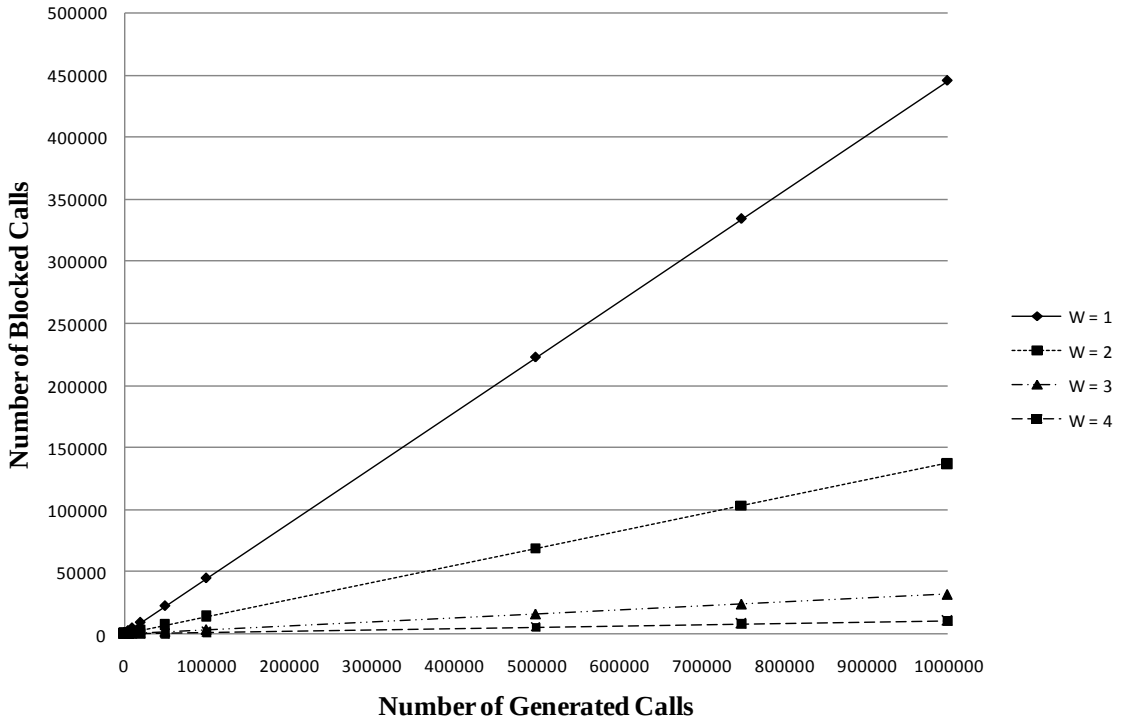

Fig. 6. Reduced call blocking for 4X4 Benes network as number of wavelength (W) increases.

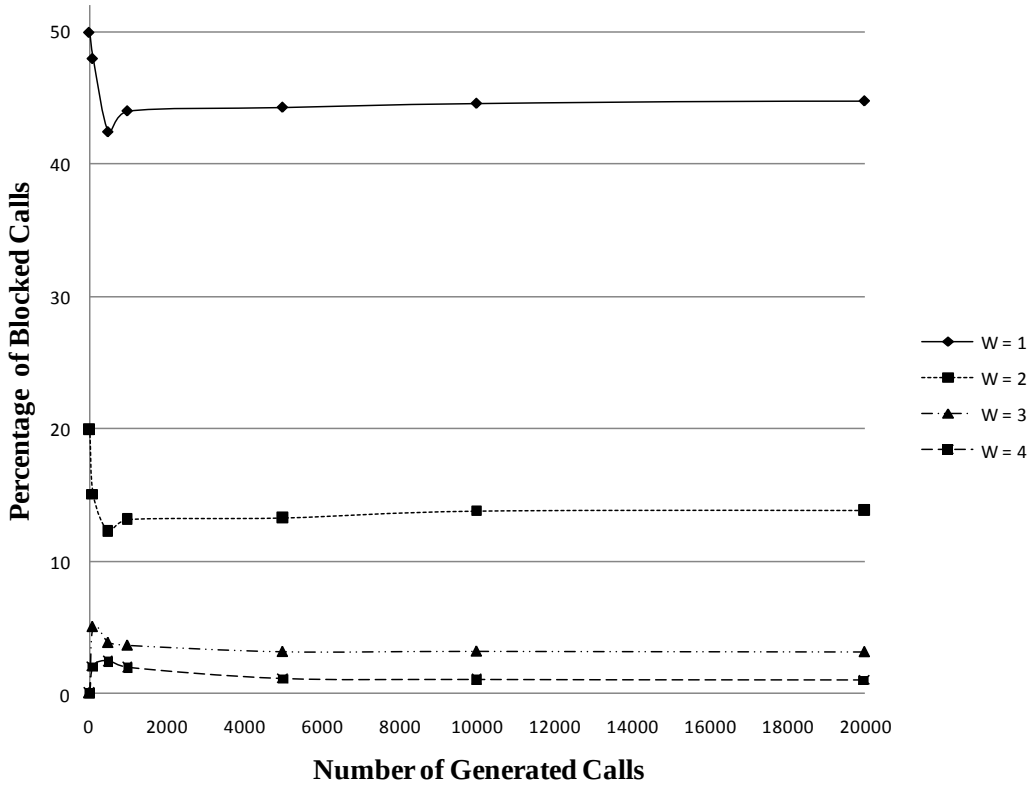

Fig. 7. Asymptotic convergence of the percentage of blocked calls for a 4X4 Benes network.

stages of the network was not allowed. A call was completed in its entirety using a single wavelength. As can be expected, the number of blocked calls dropped as the number of available wavelengths (W=1, 2, 3, and 4) was increased. Figure 7 shows that the percentage of the blocked calls converges asymptotically as the simulation reaches steady state. For W = 1, this percentage is 44.5 and for w = 4, this percentage drops to about 1.

ISBN: 978-988-19253-1-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

V. CONCLUSION AND FUTURE RESEARCH

Due to the rapidly growing complexity of computer networks, simulation tools play a crucial role in characterizing the behavior of the current system, predicting resilience and fault tolerance capabilities of new technologies, and evaluating new data transfer protocols. In this paper, we demonstrate that discrete event simulation approach is a valuable tool for evaluating interconnection networks that play a central role in determining the overall performance of a multiprocessor system. In addition, DES tools such as OMNEST can be applied to choose from a wide variety of HPC system design options, to optimize the processor and interconnection network performance, and to aid in the design of intelligent software [9]-[10]. For future research, we propose to examine the scalability of the switch matrices in an interconnection network for very large scale systems such as the exascale machines that are currently being investigated. We also aim to include details of the device physics in our DES simulation model.

REFERENCES

[1] J. Banks, J. S. Carson, B. L. Nelson, and D. M. Nicol, *Discrete-Event System Simulation*. 5th Ed. New Jersey: Prentice-Hall, 2009.
[2] M. Guizani, A. Rayes, B. Khan, and A. Al-Fuqaha, *Network Modeling and Simulation: A Practical Perspective.* West Sussex, United Kingdom: John Wiley and Sons Ltd., 2010.
[3] The Network Simulator 2 project. http://nsnam.isi.edu/nsnam/.
[4] The OPNET Modeler suite. https://www.opnet.com/.
[5] The OMNeT++ discrete event simulator. http://www.omnetpp.org/.
[6] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*. New York: Academic Press, 1965.
[7] H. S. Hinton, J. R. Erickson, T. J. Cloonan, F. A. P. Tooley, F. B. McCormick, and A. L. Lentine, *An Introduction to Photonic Switching Fabrics.* New York: Plenum Press. 1993.
[8] A. Varga, "The OMNeT++ discrete event simulation system," in *Proc. European Simulation Multiconference*, Prague, Czech Republic, June 2001, pp. 319-325.
[9] P. Huang, D. Estrin, and J. Heidemann, "Enabling large-scale simulations: selective abstraction approach to the study of multicast protocols," in *Proc. Sixth International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, Montreal, Canada, July 1998, pp. 241-248.
[10] C. Minkenberg and G. R. Herrera, "Trace-driven co-simulation of high-performance computing systems using OMNeT++," *Second International Conference on Simulation Tools and Techniques*, Rome, Italy, March 2009, article 65.

ISBN: 978-988-19253-1-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)