Modeling InfiniBand with OPNET

Aurelio Bermúdez, *Student Member, IEEE*, Rafael Casado, *Member, IEEE*, Francisco J. Quiles, *Member, IEEE*, Timothy M. Pinkston, *Senior Member, IEEE*, and José Duato, *Senior Member, IEEE*

***Abstract*— The InfiniBand architecture is a new standard for high-speed I/O and interprocessor communication, suitable for clusters and high-end servers executing distributed applications. Its specification is open to further enhancements by manufacturers and researchers. This paper presents an OPNET model of IBA. It can serve as a starting point for helping SAN designers evaluate the performance trade-offs of this new interconnection standard and locate its potential bottlenecks. Moreover, our model is a suitable platform on which to develop new proposals that may extend the functionality and performance of IBA.**

***Index Terms*—InfiniBand architecture, network modeling and simulation, OPNET.**

I. Introduction

THE InfiniBand architecture (IBA) [5][3] is a new standard for high-speed I/O and interprocessor communication. It as been developed by the InfiniBand Trade Association (IBTA) [5]. Founded in 1999, this association includes over 200 leading companies, including IBM, Intel, Microsoft, Compaq, HP, Sun, Dell, and Cisco. The result is a fast, highly scalable interconnect technology suitable for clusters and high-end servers executing distributed applications. The IBA specification—though an emerging standard—is open to further enhancements by manufacturers and researchers. Recently, the first commercial IBA-compliant products have started to appear in the marketplace [8][4].

This paper presents a model which embodies key physical and link layer features of IBA, which allows simulation of various IBA-compliant network designs. We have used the OPNET Modeler [9] simulation software. OPNET is a useful engineering and research tool for streamlining the design and performance analysis of communication systems and protocols. Our model can serve as a starting point (basis) for helping SAN designers evaluate the performance trade-offs of this new interconnection standard and locate its potential bottlenecks. Moreover, our model is a suitable platform on which to develop new proposals that may extend the functionality and performance of IBA. In this regard, the main objectives of this research are to develop and evaluate more efficient subnet management protocols—specifically, network discovery and reconfiguration techniques.

In Section II, we give an overview of relevant IBA features. Section III introduces the OPNET Modeler software. Section IV describes how we have modeled IBA using this tool. In Section V, we show how we can use this simulation model to evaluate the performance of any IBA topology and to analyze subnet management mechanisms. Finally, in Section VI, we conclude and outline future work.

Manuscript received December 5, 2002. This work was supported in part by the Spanish CICYT under Grant TIC2000-1151-C07-02, and the National Science Foundation under Grant CCR-0209234.

A. Bermúdez, R. Casado, and F. J. Quiles are with the University of Castilla–La Mancha, Department of Computer Science, 02071 Albacete, Spain (phone: 34-967-599200-2484; fax: 34-967-599224; e-mail: {aurelio.bermudez, rafael.casado, francisco.quiles}@uclm.es).

T. M. Pinkston is with the Department of Electrical Engineering/Systems, University of Southern California, Los Angeles, CA 90089 USA. (e-mail: tpink@charity.usc.edu).

J. Duato is with the Department of Systems Engineering, Computers and Automation, Technical University of Valencia, 46071 Valencia, Spain (e-mail: jduato@gap.upv.es).

II. InfiniBand Architecture

A. Overview

IBA defines a system area network (SAN) environment, where multiple processor nodes and I/O devices are interconnected using an arbitrary (possibly irregular) switched point-to-point network, instead of using a shared bus. Processor nodes can include several CPUs and memory modules, and they use one or several host channel adapters (HCAs) to connect to the switch fabric. I/O devices can have any structure, from a simple console to a RAID subsystem. These devices use one or several target channel adapters (TCAs) to connect to the fabric. The fabric is structured in subnets connected by means of routers, as illustrated in Fig. 1. The specification allows three different topologies for the

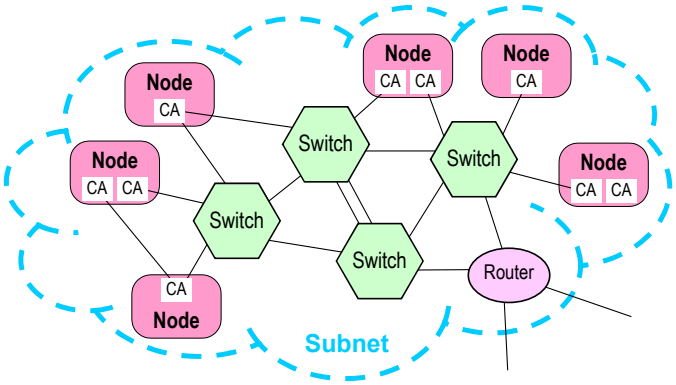

Fig. 1. IBA subnet. Each subnet includes a set of switches and point-to-point links. Several links can be used in parallel to increase the bandwidth, or to provide redundant paths to increase fault tolerance.

fabric: a direct connection between two end nodes, a single subnet, and a set of subnets interconnected by routers.

IBA uses copper or optical links. The raw bandwidth of an IBA 1X link is 2.5 Gbps. Data bandwidth is reduced by 8b/10b encoding to 2.0 Gbps. Therefore, for a full duplex connection, the data rate is 4 Gbps. Other specified link bandwidths are 10 and 30 Gbps (named 4X and 12X links, respectively).

B. Link Layer

The architecture is divided into multiple layers where each one operates independently of another. IBA is decomposed into the following layers: physical, link, network, transport, and upper layers. The link layer (along with the transport layer) is the heart of the architecture. This layer encompasses packet layout, point-to-point link operations, and switching within a local subnet. Next, we briefly describe several link layer issues. Additional details can be found in the description of the IBA model.

IBA uses a virtual lane (VL) based mechanism to create multiple virtual links within a single physical link. This improves link utilization and provides support to prevent deadlock. Each port could implement 1 (VL0), 2 (VL0-1), 4 (VL0-3), 8 (VL0-7), or 15 (VL0-14) data virtual lanes. For subnet management purposes, an additional virtual lane (VL15) must be implemented. Each VL has a dedicated set of transmit/receive packet buffers to store at least one packet. Virtual cut-through switching [6] is considered. A credit-based flow control mechanism acts separately over each data VL. Output channels in switches and channel adapters use a VL arbitration table defined by the subnet manager to determine the next packet to inject into the physical link. This table defines the amount of packets to send from each VL, allowing traffic prioritization.

As a packet traverses the subnet, each link along the path can support a different number of VLs. Therefore, it is not possible to transmit a packet only using a VL specified in its header. Instead, IBA uses a more abstract criterion, based on the concept of service level (SL). Each switch has a SL to VL mapping table (set by the subnet manager) to establish a correspondence between the service level of the packet (a number from 0 to 15) and the VLs supported by the output port assigned to the packet. In this way, IBA can ensure end-to-end QoS.

All devices within a subnet have a 16 bit local identifier (LID) assigned by the subnet manager. Packets within a subnet use the LID for addressing. This value is included in the packet header, with the SL. Switches perform the intra-subnet routing, using the packet's destination LID. To decide the output port for an incoming packet, switches use a forwarding table configured by the subnet manager. Each switch must support one of two types of forwarding tables, the random forwarding table (RFT) and the linear forwarding table (LFT). The main difference between them is that each entry in the former must include an explicit LID value, whereas in the later, the LID value is implicit in the position of the entry in the table.

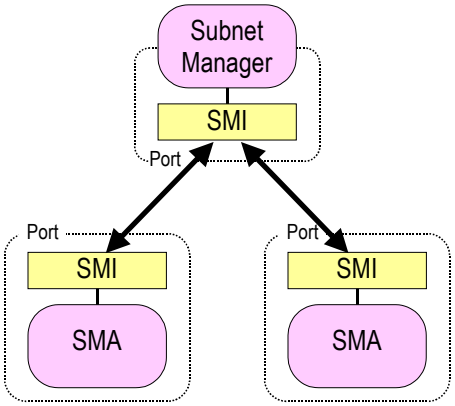

Fig. 2. Subnet management model.

C. Subnet Management

IBA subnets are managed in an autonomous way using several entities shown in Fig. 2. There is a subnet manager (SM) in charge of discovering, configuring, activating, and maintaining the subnet. Through the subnet management interface (SMI), this entity exchanges control packets with the subnet management agents (SMAs) present in every subnet device.

To guarantee compatibility between different vendor implementations, the specification defines the subnet management entities describing their functions and the structure of the control packets used to exchange information among them. However, the exact behavior of these management entities has not been detailed. Examples of commercial management products can be found in [7][14]. Our main goal is the design of efficient subnet management protocols such as those in [1][10] that can be applied to IBA.

III. OPNET MODELER

OPNET Modeler [9] provides support to model communication networks and distributed systems. Network engineers use this tool to design and analyze communication equipment and protocols. The use of OPNET accelerates the design, development and deployment cycle, and improves product performance and reliability.

This tool defines three work domains: network, node, and process domain. The node level specifies objects belonging to the network level, and the process level specifies objects belonging to the node level.

Network models consist of nodes, links, and subnets. Nodes represent network devices. Links represent point-to-point and bus connections. Subnets organize network components into a single object. Network models are developed using the project editor (Fig. 3). This editor allows rapid construction and test of various possible network configurations.

Node models are built using basic OPNET modules and specifying the connections between them. In Fig. 4 we can see (from left to right) processors, queues, packet streams, statistic

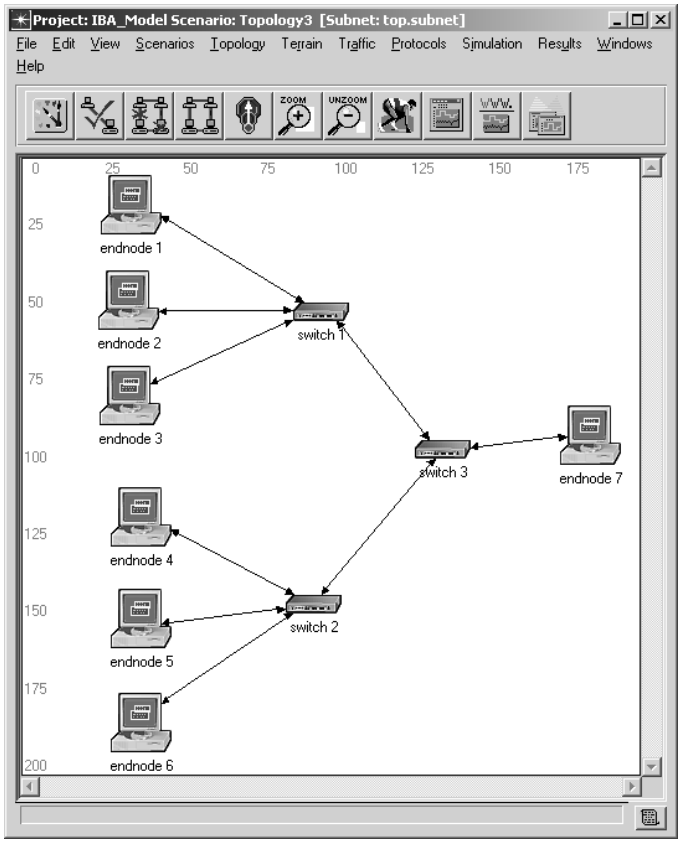

Fig. 3. Project editor. The window shows an example of a topology composed of three switches and seven hosts.

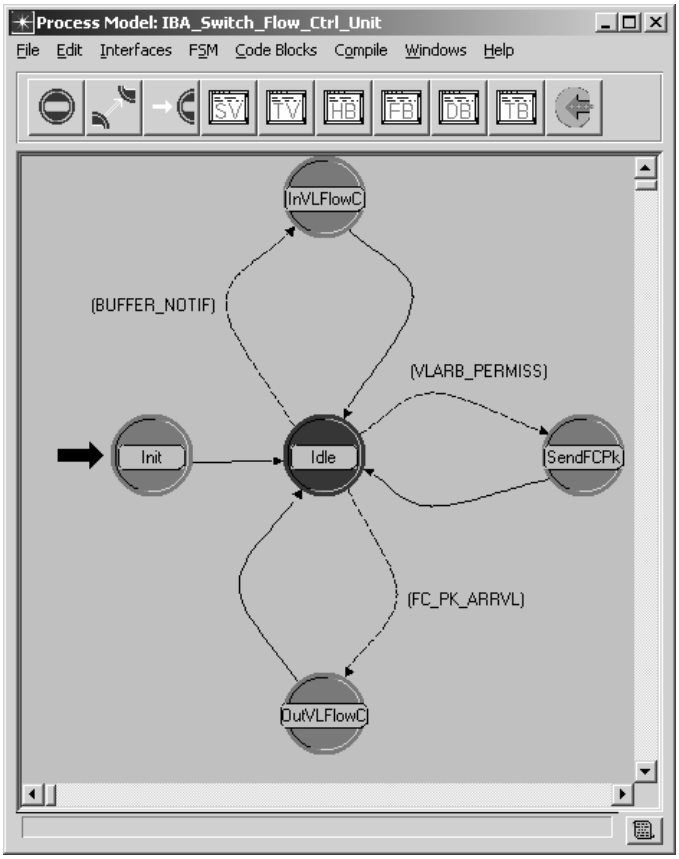

Fig. 5. Process editor. The window shows an example of a finite state machine. The large arrow indicates the initial state.

wires, logical associations, transceivers (point-to-point, bus, and radio links), and antennas. Each module can generate, send, and receive packets from other modules to perform its function. Modules typically represent applications, protocol layers, and physical resources, such as buffers, ports, and buses. Packet streams allow the transmission of packets among the node modules. Statistic wires are used to exchange control information. Node models are developed using the node editor.

The behavior of processors and queues are programmable via their process models. They consist of finite state machines (FSM) containing blocks of C/C++ code and kernel procedures (KP). State machines respond to interrupts generated by the OPNET simulation kernel. The user code is executed when the machine enters or leaves a state, and it can also be associated to a transition. Kernel procedures provide the way to perform common tasks, such as to manipulate packets, collect statistics, operate with queues, or program future interrupts. Fig. 5 shows an example of a process model.

Additional OPNET editors allow the design of links, packets, probability density functions, etc. Using the link editor, a user can specify bandwidth, bit error rate, propagation delay, packet types supported by the link, as well as other attributes of the link. The packet editor allows the system designer to graphically build packets of the format desired, naming the packet fields, specifying their size, and choosing the type for the value stored. It is also possible to encapsulate in a field a packet from an upper network layer.

OPNET simulations are event-driven. The simulation kernel handles a single global event list and a shared simulation time clock. Events are removed from the list and delivered to the appropriate module in time order. The process associated to the destination module performs the actions programmed for the event. These actions may (directly or indirectly) involve the generation of new events. Then, the simulation kernel receives requests from processes and inserts new events in the event list.

Finally, the OPNET Modeler provides several tools to run sequences of simulations, and choose and analyze the results. Some examples are presented in Section V.

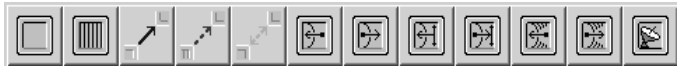

Fig. 4. Node domain elements.

IV. IBA Model Components

Our current IBA model is composed of links, switches, and end nodes containing a HCA, as described below.

A. Links

We have modeled IBA links starting from the basic OPNET

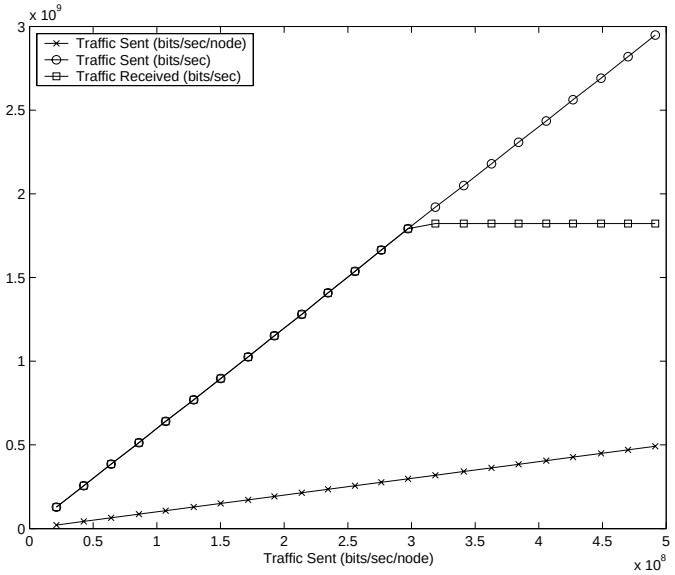

Fig. 6. Example of validation.

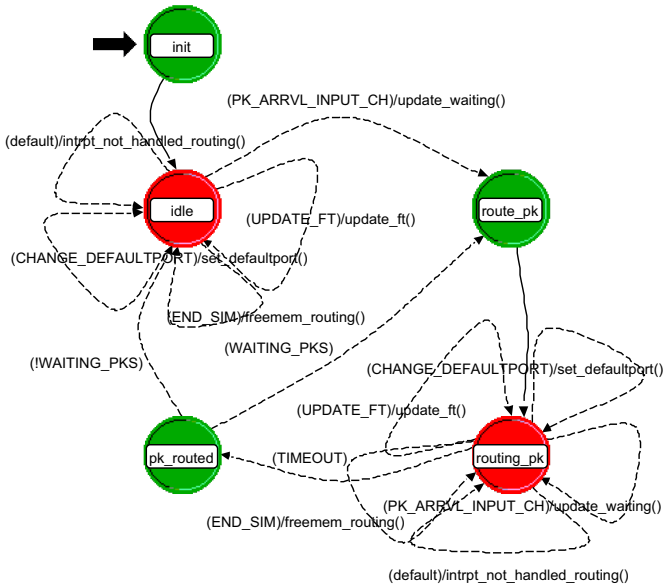

Fig. 7. Routing unit state machine. The *idle* state waits for requests from input buffers. The *routing_pk* state models the time to access the routing table. This process begins and finishes in the *route_pk* and *pk_routed* states, respectively.

point-to-point bidirectional link. The specified bandwidth for 1X links is 2.5 Gbps. Fig. 6 shows some results related to the topology previously shown in Fig. 3. The six end nodes on the left send packets to the one located on the right. The plot shows on the vertical axis the amount of traffic injected by each source, the aggregated traffic injected by the six sources, and the traffic received at the destination. We can see that the maximum link bandwidth (2 Gbps) is not exceeded.

As in [11] we consider 20 meter copper cables with a propagation delay of 5 ns/m; therefore, the flight time has been set to 100 ns. We have not considered transmission errors.

Instead of store-and-forward packet switching, the receiver begins to process the packet when the complete header is received. In the case of data packets, it requires 20 bytes [5]. In the case of flow control packets, the header matches up with the entire packet (6 bytes).

B. *Switches*

Fig. 9 shows the internal structure of a 4-port IBA switch model. Next, we describe the different blocks that compose this model.

1) *Input Channels*

Each input channel contains a point-to-point receiver connected to the link. A demultiplexer analyzes the header of incoming packets and delivers them to the corresponding units. Flow control packets are sent to the flow control unit. Data packets are sent to the input buffers associated with virtual lanes. The IBA specification allows up to 15 data VLs. To this point, we have incorporated only two of them (VL0-1) in our IBA model. The management virtual lane, VL15, over which control packets flow is also incorporated in our model. A notification is sent to the flow control unit each time the available resources in input buffers vary. Additionally, when a packet reaches the head of a VL, a notification is issued to the routing unit. Incoming packets could also be dropped according to the current port state.

2) *Routing Logic*

The packet routing logic includes the routing unit and the service level to virtual lane (SL to VL) mapping unit. The

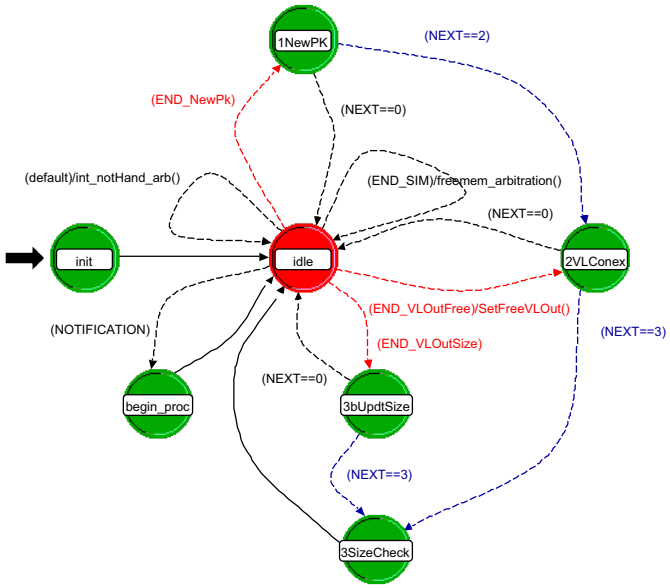

Fig. 8. Arbitration unit state machine. The *begin_proc* state models the time to process an incoming notification. When a notification is received from the mapping unit, the *1NewPK* state is reached. If the output channel is busy, the machine returns to the *idle* state; otherwise, the state *2VLConex* is reached. This state is also reached when the channel is released. When the *3SizeCheck* state is reached, the unit checks for resources to allow the entire packet in the output buffer. If there is enough space, the connection is established. When buffer space is released, due to the transmission of a packet through the link, the *3bUpdtSize* state is reached, and size is checked again.

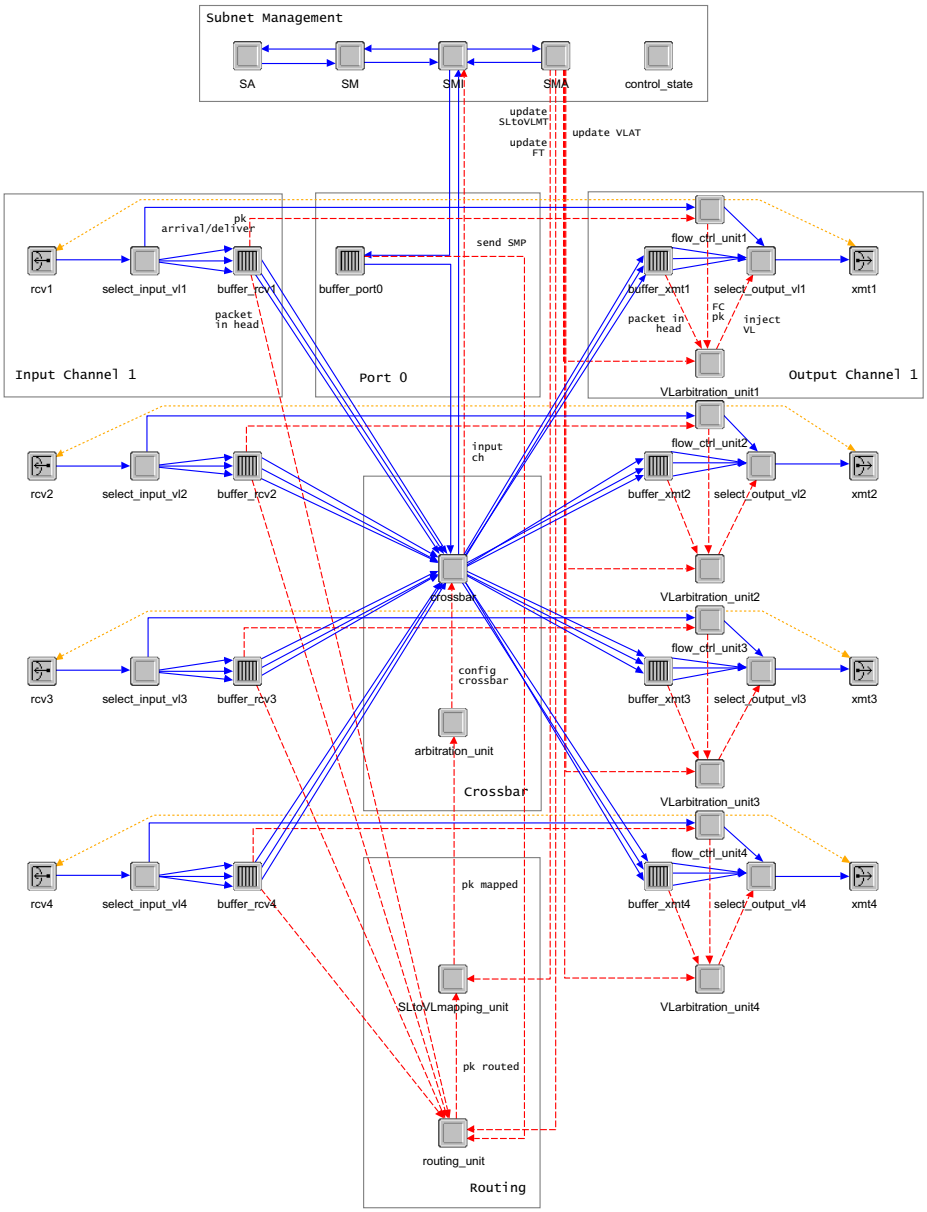

Fig. 9. Internal structure of a four port IBA switch model.

routing unit assigns an output port to each incoming packet. The model shown in Fig. 7 processes simultaneous requests in a sequential and prioritized way. First, it considers packets in VL15. Then, for data VLs, it applies a round-robin policy. The routing unit uses a (unicast) forwarding table, configured by the management protocol. When the forwarding table can not provide an output port for the packet's destination, the provided port is not supported by the switch, or if it is the same as the input port used by the packet, the routing unit notifies the input buffer that the packet must be discarded. For management packets using directed (source) routing, the routing unit employs the information stored in the packet instead of looking up routing options in the forwarding table.

After a packet has been successfully routed, the result is transferred to the mapping unit, in order to determine an output VL for the packet. This unit includes a mapping table defined by the management protocol. The mapping function is not applicable to management packets, and it could also lead to discarding the packet (when the management VL is provided by the mapping table). Finally, this unit provides all this information to the crossbar arbitration unit.

3) Switching Logic

The switching logic (see center of Fig. 9) is composed of a fully demultiplexed crossbar and the arbitration unit. We have also modeled a multiplexed crossbar commonly used in commercial products. The crossbar unit allows several packets to cross simultaneously from input buffers to the corresponding output buffers. The arbitration unit arbitrates among concurrent requests and configures the crossbar. The policy used to establish crossbar connections satisfies the following conditions: VL15 has priority over data VLs and all data VLs have the same priority. The state machine associated to this unit is shown in Fig. 8.

4) Output Channels

Crossbar outputs are connected to the output buffers associated with VLs. These output buffers are connected to a multiplexer. This unit receives management and data packets from the output buffers, and flow control packets

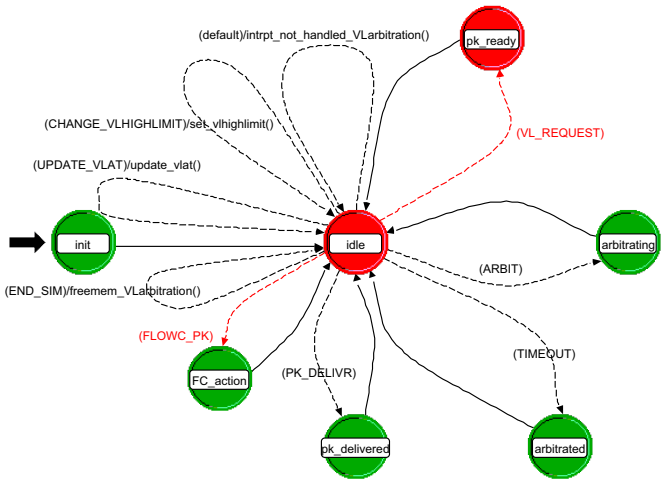

Fig. 10. Channel arbitration unit state machine. The *pk_ready* state manages requests from the output buffers. The *arbitrating* and *arbitrated* states model the time to select the next packet to inject into the link. The *pk_delivered* state indicates that an entire packet has left the output buffer, and the channel has been released. The *FC_action* state is reached when the flow control unit activates or deactivates the transmission through a VL.

from the flow control unit. According to the current port state, the multiplexer may drop packets.

The IBA specification defines a credit-based flow control scheme. Each port incorporates a flow control unit that enables/disables the transmission from data output buffers, according to the flow control packets received from the port at the other end of the physical link. Additionally, the flow control unit sends flow control packets to stop/restart the injection of data packets at the other end of the link. The state machine related to this unit has been shown in Fig. 5.

The channel arbitration unit model is shown in Fig. 10. It receives requests from the output buffers and the flow control unit, and it determines the next packet to inject into the physical link. Management and flow control packets have the maximum priority. To determine the priority between data VLs, this unit uses the VL arbitration table provided by the management protocol.

5) Management Port

Finally, for administration purposes, the switch includes an internal port (numbered 0) connected to the management logic. This port implements only a packet buffer associated with VL15.

C. End Nodes

Fig. 11 shows the end node model implemented, including a basic processor node and an interface card (HCA).

The processor node we have modeled is very simple. It consists of a packet source and a packet consumer. The source generates link layer data packets according to a probability density function, and the consumer drops received packets and updates traffic statistics (packet delay, throughput, etc.). This model could be extended in the future, considering more realistic traffic approaches suitable for clusters.

The channel adapter includes a single port. As in the case of switch ports, it supports VL0-1 and VL15. Transmitter and receiver modules connect the card to the physical link. Since there is not a crossbar, the mapping unit is directly connected to the output buffers associated with the VLs. The behavior of most of the modules, including the management entities, is the same as the corresponding modules in the switch model. Upper level functions considered in the IBA specification (as queue pairs, service types, verbs, etc.) have not been modeled yet.

Both types of IBA subnet nodes (switches and end nodes) have a wide set of configurable attributes. Common node attributes are the device globally unique ID (GUID), the number of data VLs and packet maximum transfer unit (MTU) supported by the physical ports, the size of the buffers

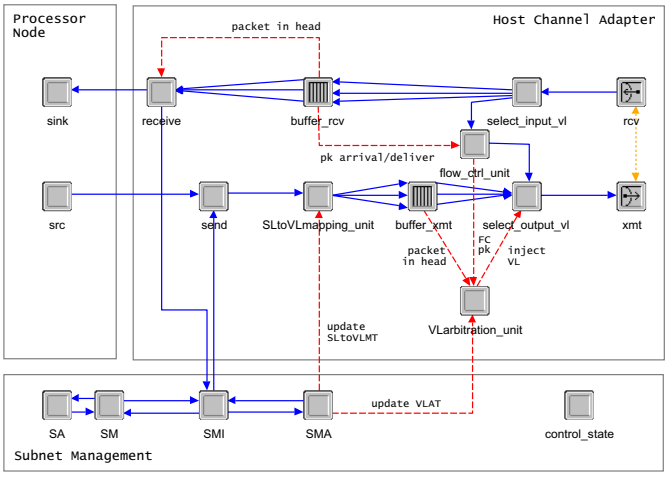

Fig. 11. Internal structure of the IBA end node model.

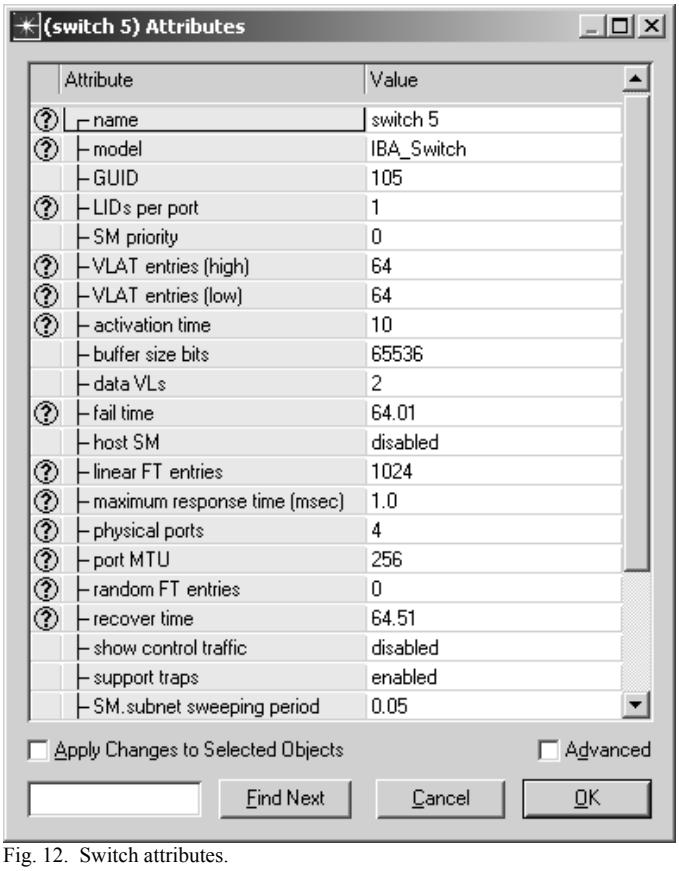

Fig. 12. Switch attributes.

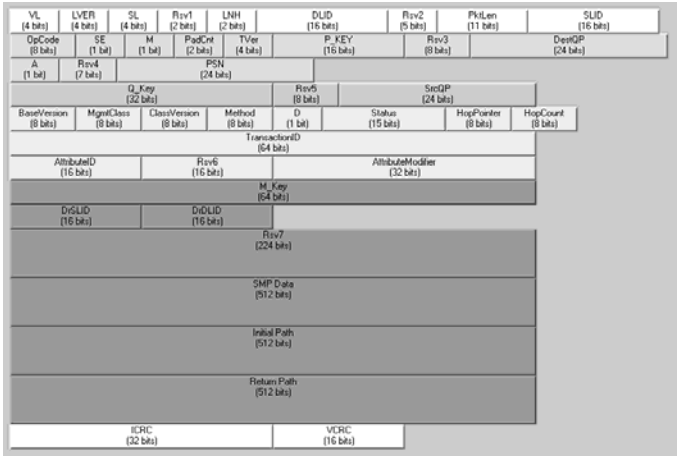

Fig. 13. Directed routed SMP.

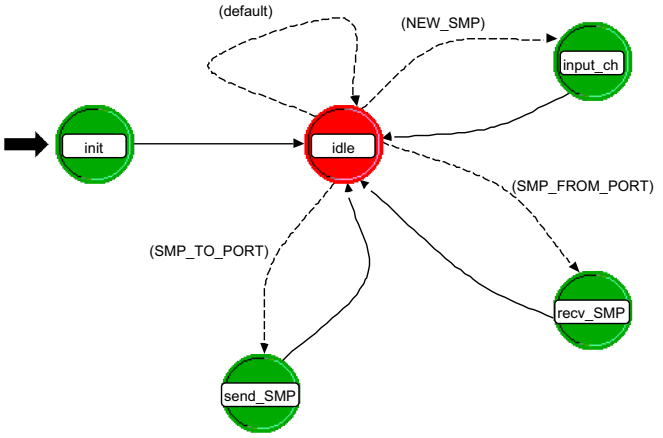

Fig. 14. SMI state machine. The *input_ch* and *recv_SMP* states manage incoming SMPs. The *send_SMP* state delivers outgoing SMPs.

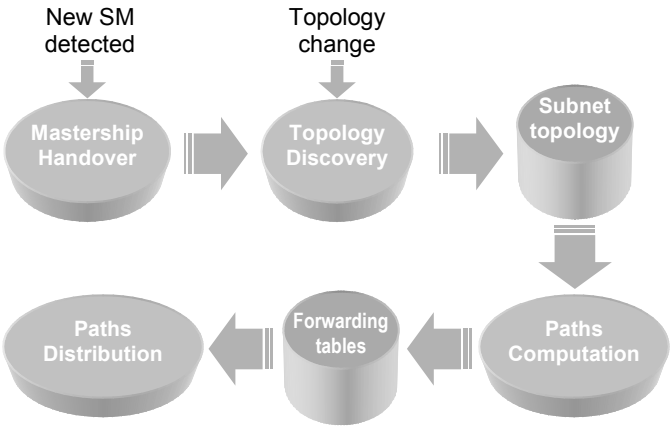

Fig. 16. Tasks performed by the SM.

associated with the VLs, and the size of the VL arbitration table used by the channel arbitration unit. Switches include the size of the forwarding table used by the routing unit (Fig. 12). Additionally, end nodes need several attributes related to packet generation (injection rate, start time, etc.).

D. Subnet Management Model

We have modeled the subnet management packets (SMPs) and the acknowledgment and reinjection protocol defined in the IBA specification. As an example, Fig. 13 shows the fields of a directed routed SMP.

Next, we describe the subnet management modules included in switches (Fig. 9) and end nodes (Fig. 11): SMI, SMA, and SM.

1) Subnet Manager Interface: Fig. 14 shows the behavior of the SMI module. It injects SMPs generated by the SM and SMA into the network. Also, it validates and delivers incoming SMPs. The destination entity for an arriving SMP depends on the packet type (request, response, trap, etc.). In switches, the SMI implements directed routing, updating in each hop the *Return Path* and *HopPointer* SMP fields.

2) Subnet Manager Agent: Fig. 15 shows the state machine associated to the SMA module. The tasks performed by this entity include processing received SMPs, responding to the SM, and configuring local components according to the management information received. The received SMPs could contain information related to physical ports, as the assigned LID, the port state, or the number of operational data VLs. Other SMPs are used to update the local forwarding table, the SL to VL mapping table, and the VL arbitration tables. The SMA is also in charge of updating the state of a port when the condition of a neighbor device changes.

3) Subnet Manager: The tasks implemented in the SM model are represented in Fig. 16. Each management task is modeled with a different state machine. Incoming SMPs are analyzed and delivered to the corresponding state machine by a dispatcher process (shown in Fig. 17) associated to the SM module. The handover task guarantees that only one SM (the master) manages the subnet at any given time. This master

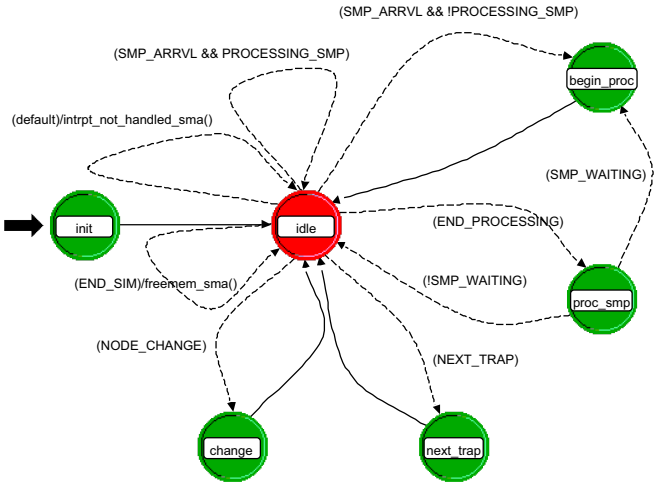

Fig. 15. SMA state machine. The *begin_proc* and *proc_smp* states model the time consumed in processing a received SMP sent by the SM. The *change* state is reached if a change is detected at any port. The *next_trap* state manages trap notifications.

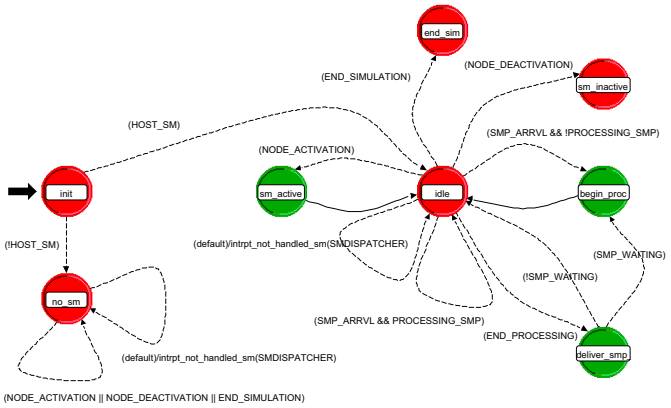

Fig. 17. Dispatcher process state machine. The *begin_proc* and *deliver_smp* states model the time to process an incoming SMP sent by another management entity in the subnet. The *deliver_smp* state provides the SMP to the state machine that must manage it. SMPs are sequentially processed.

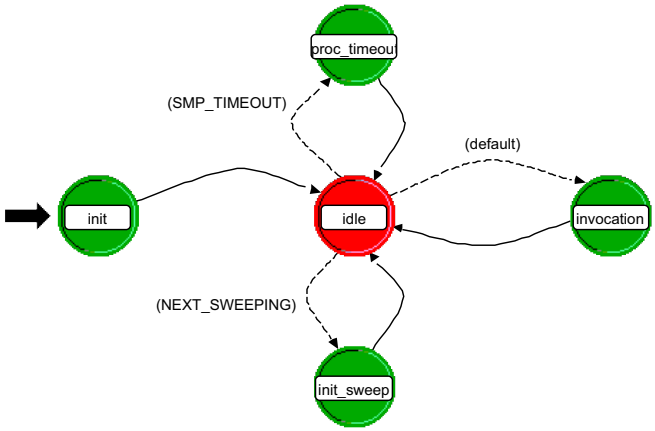

Fig. 18. Discovery process state machine. The *invocation* state processes a SMP received from the dispatcher process. The *proc_timeout* state is reached when the timeout associated with a request SMP expires, and the corresponding response has not been received. The *init_sweep* state starts the next subnet sweeping process.

periodically sweeps the subnet searching for topology changes. It is also possible that SMAs located in the switches notify port changes to the master. In either case, the master performs the topology discovery task shown in Fig. 18. It includes the assignment of LIDs and the configuration of other port attributes. After that, routes are computed and distributed to the forwarding tables. To determine the paths through the subnet, the master uses the collected topological information.

In the management mechanism currently implemented, the above tasks are sequentially executed. After a topology change is detected, the subnet topology is collected starting from scratch. The discovery process is centralized in the master, which performs a propagation-order exploration. To compute the routes, the master uses the up*/down* routing algorithm [12]. Finally, in order to prevent deadlock situations, all subnet ports are deactivated before sending the new switch forwarding tables. Once the tables are completely distributed, the subnet is activated again. Hence, static reconfiguration is assumed.

The SM must receive a response for each request SMP. If the response is not received after a period of time, the SMP is injected again. After several injections for the same SMP, the SM concludes that the destination is either disabled or unreachable.

Each component (switches and end nodes) in the subnet contains several configurable management attributes. They specify if the component hosts a SM, if it supports traps (only for switches), or the maximum response time for the node SMA.

Additionally, using profiling techniques, we have measured the time required to process a SMP by the management entities, for a wide set of subnet configurations. In this way, the processing times are dynamically computed according to the subnet size.

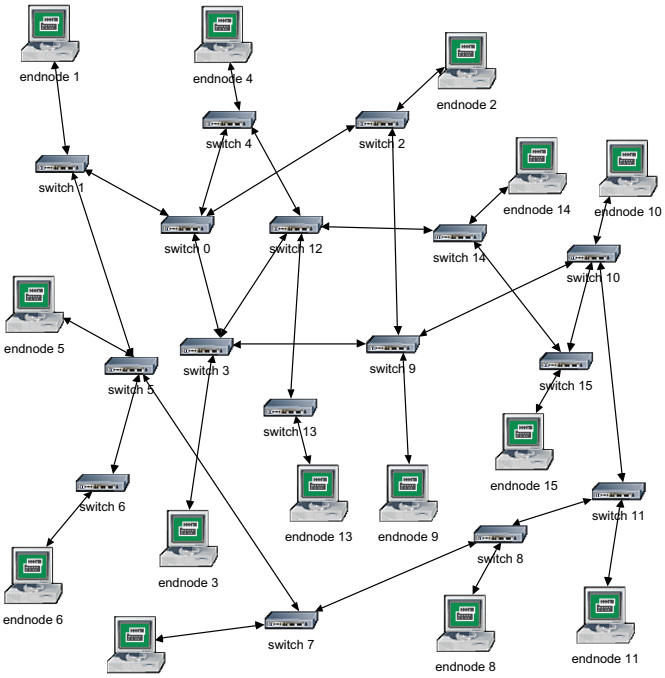

Fig. 19. Example of a cluster composed of 16 IBA switches and 14 hosts in an irregular topology.

V. Evaluating IBA Performance

The purpose of this section is not to present a detailed performance evaluation of IBA. Instead, we use some examples to illustrate the way in which our OPNET IBA model can be used to analyze different aspects of the architecture.

A. Building Subnet Topologies

Using the IBA link, switch, and end node models described before, we may build and evaluate any subnet configuration.

For this work, we have selected a set of randomly generated irregular topologies. The network size ranges from 8 to 64 4-port switches. Each switch has connected an end node, if a

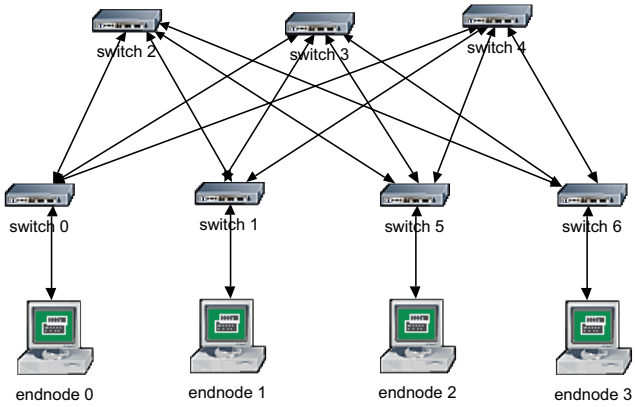

Fig. 20. Example of a cluster composed of 7 IBA switches and 4 hosts in a Clos topology (3,1,4).

port is available. Fig. 19 presents an example of a 16-switch topology.

Several studies focus on network topologies applied to NOW and SAN environments. An example of a more regular topology is the Clos network [2]. It has excellent properties such as full bisection, scalability, modularity, and multiple-path redundancy that make it ideal for cluster networks [13]. Fig. 20 shows an example of Clos topology. At the moment, we are increasing the amount of ports available in the current switch model. It would allow the generation of larger Clos topologies than that shown in the figure.

B. Preparing Simulations

Once we have defined the network topology and attribute values for each device in the subnet (as shown in Fig. 12), we must specify the statistics that will be collected during the simulation and the way they are collected. Fig. 21 shows the editor used for that purpose. OPNET provides a wide set of predefined statistics; additionally, the user can customize the model by adding new ones. Statistics can be collected in several ways: all values (recording all updates to the statistic), samples (collecting only certain statistic updates), buckets (recording a single value from a set of points, for example, the sample mean), etc. Additionally, for each statistic we can store an output vector ordered by time, and/or an individual value that summarizes the behavior of the statistic (the average value, the maximum value, etc.).

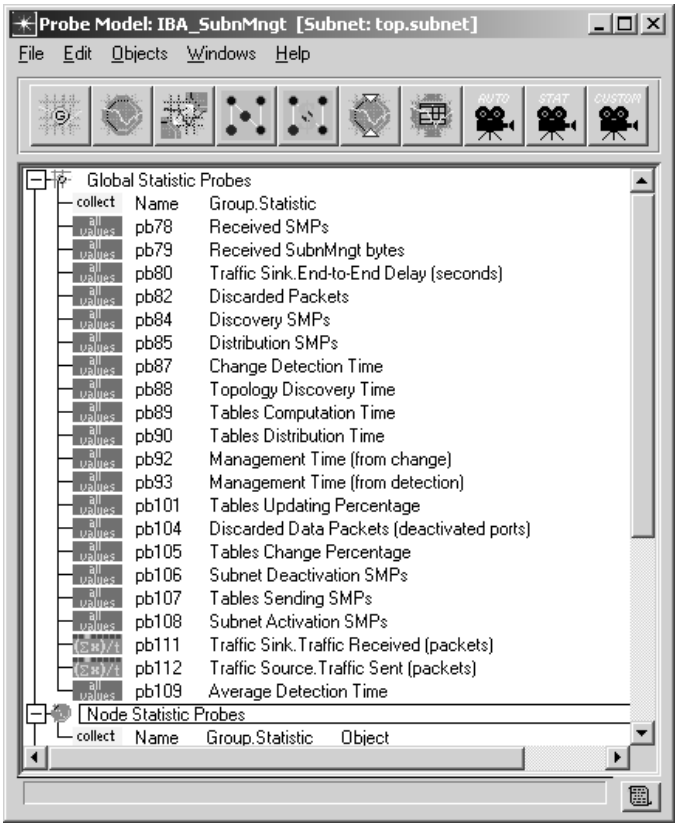

Fig. 21. Probe editor. The window shows the set of customized statistics we have defined to analyze subnet management protocols, as the amount of time that the user traffic is stopped, or the time and number of SMPs required by each management task. They may help us to locate the bottlenecks of a management mechanism and compare different management policies.

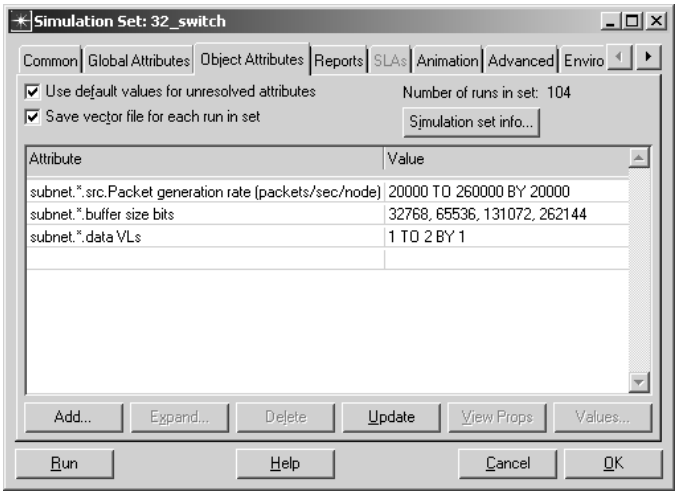

Fig. 22. Specifying simulation parameters. In the example of the window, the 104 simulations are obtained as combination of 13 traffic rate values (20E3, 40E3, 60E3, 80E3, 100E3, 120E3, 140E3, 160E3, 180E3, 200E3, 220E3, 240E3, and 260E3 packets/sec), 4 buffer sizes (32768, 65536, 131072, and 262144 bits), and 2 different numbers of operational data VLs (1 and 2).

The value of model attributes can be defined before running the simulations. For the results presented here, we have fixed some of them, as the duration of the simulation, the probability density function used by the packet sources, and the amount of upper-level information contained in these packets.

We have programmed for all simulation runs a complete subnet activation at time 10 sec. Sources begin to generate packets at time 60 sec. To evaluate network performance, we have considered a transient period of 0.1 sec and, after that, a useful period of 0.1 sec, obtaining a total duration of 60.2 sec. During the useful period, statistics are collected. To evaluate the subnet management mechanism, for each simulation we have programmed a topology change (activating or deactivating a switch) after time 60 sec. For these probes, the duration of the simulation varies according to the topology.

We have considered a uniform distribution for packet generation, assuming the same generation rate for all sources. Sources also use a uniform distribution to obtain the packet destination and service level. As packet length, we have considered in all cases a MTU of 256 bytes, the minimum value allowed by the IBA specification.

Other model attributes can be parameterized over a set of simulations. In this work, we have considered as simulation parameters the number of packets per second sent by each end node, the amount of operational VLs in the subnet ports, the buffer size, and the device internal bandwidth. Fig. 22 shows the way to specify some of these parameters to create a set of simulations.

For each topology we have considered packet generation rates from low load to saturation. For the number of operational VLs, we have used 1 or 2 data VLs. For buffer sizes, we have assumed values ranging from 32768 bits (4096

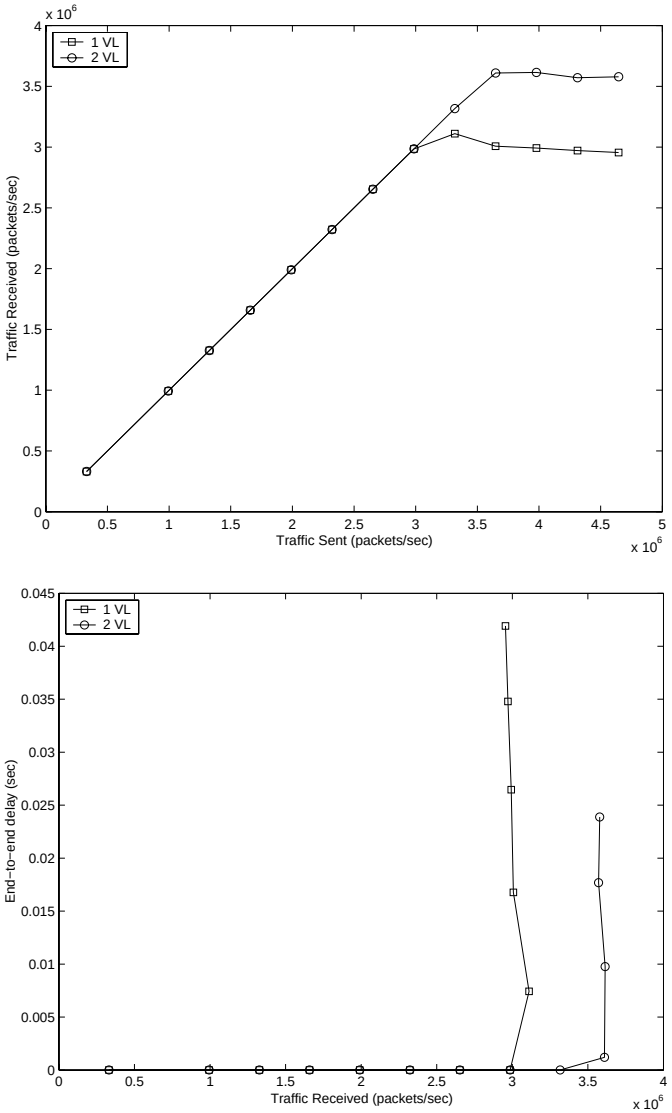

Fig. 23. Influence of the amount of VLs on performance (assuming a buffer size of 32768 bits).

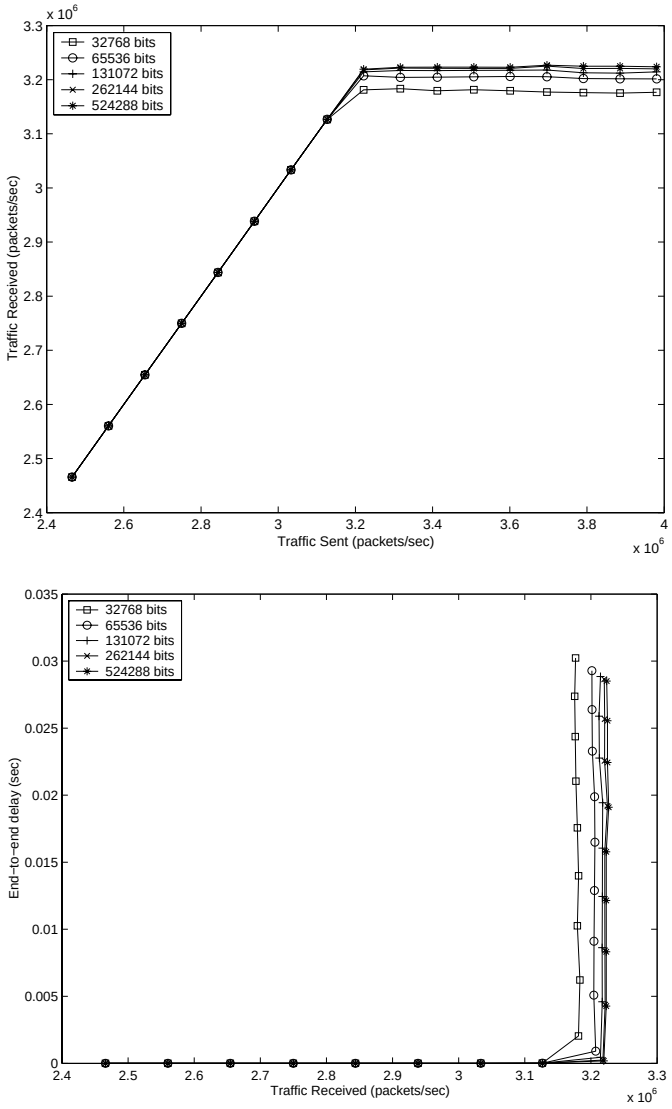

Fig. 24. Influence of VL buffer size on performance (assuming 2 data VLs per port).

bytes) to 524288 bits (65536 bytes). Finally, we have used a factor (ranging from 1 to 2) to specify different internal bandwidths for the subnet devices. This factor is applied to the bandwidth for the IBA 1X link.

C. Evaluating Design Parameters

In this section, we show several examples of utilization of our model to analyze the impact of some architecture parameters on network performance. The parameters we have varied are the number of VLs per channel, the size of their associated buffers, and the device internal bandwidth.

Fig. 23 shows the simulation results obtained for the subnet topology presented in Fig. 19. The top plot represents network throughput versus network load. Load is expressed using the number of packets per second send by all the subnet end nodes; throughput is expressed using the number of packets per second received by all the subnet end nodes. The bottom plot represents packet latency versus network throughput. Each point in this plot indicates the average end to end delay obtained in the corresponding simulation. We have obtained these results for 1 and 2 operational data VLs per port. As shown in [11], we can clearly appreciate the increment in performance using virtual lanes.

Fig. 24 shows the same statistics as Fig. 23, obtained for the Clos topology shown in Fig. 20. The parameter we have analyzed is the size of the packet buffers associated to the VLs in the subnet switch ports. We can see that in this case buffer size does not contribute as much to increased network throughput.

Finally, Fig. 25 analyses the impact of device internal bandwidth on network performance. Results correspond to an irregular topology with 8 switches and 7 end nodes. Again, there are not significant increments in performance for factor values of 1.5 and 2.

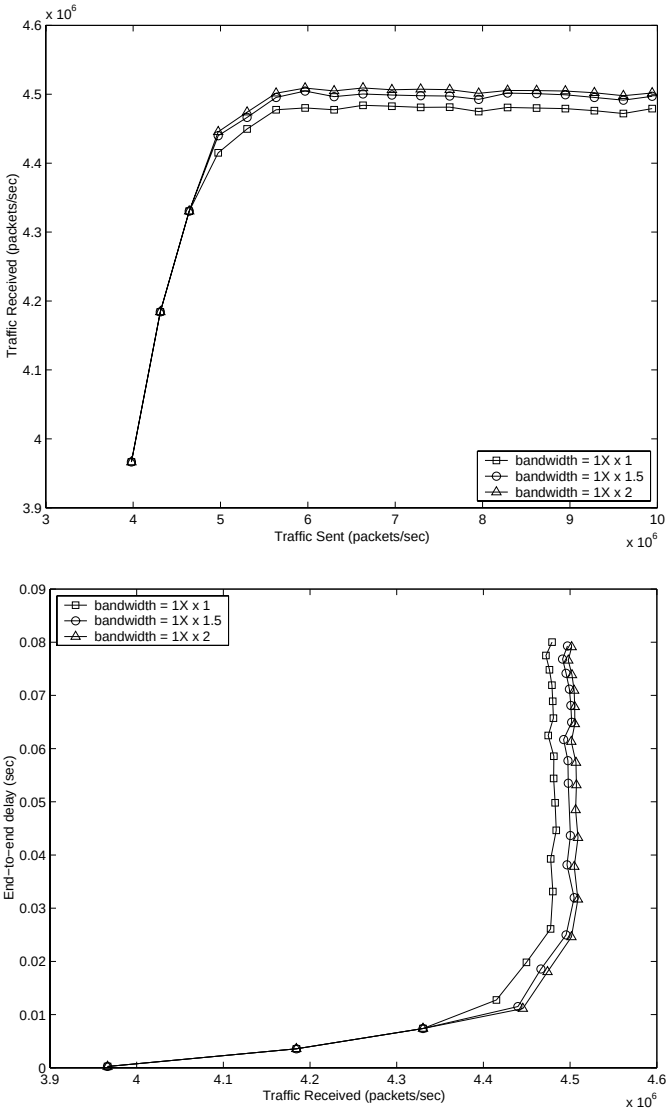

Fig. 25. Influence of internal bandwidth on performance (assuming 2 data VLs per port and a buffer size of 32768 bits).

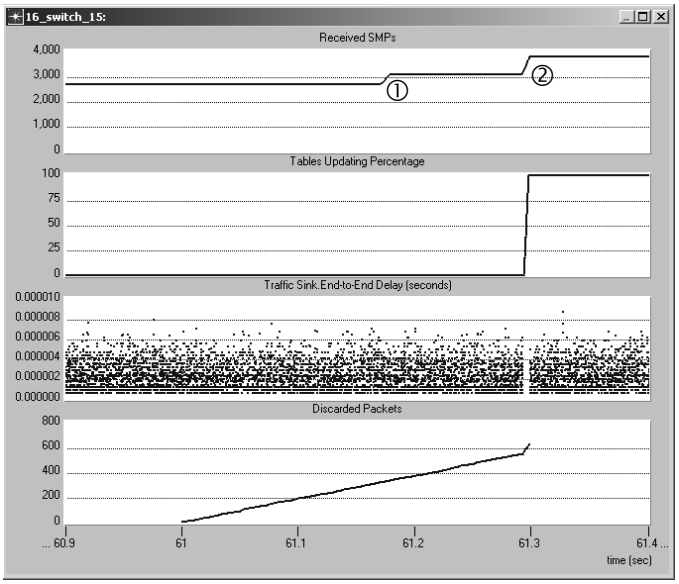

Fig. 26. Effect of SM tasks on user traffic.

D. Evaluating the Subnet Management Mechanism

Fig. 26 shows an example of how our OPNET model could be used in the analysis of new IBA subnet management protocols. Plots represent some statistics added to the IBA model in order to evaluate management mechanisms (see Fig. 21).

The results correspond to the topology shown in Fig. 19, in which switch 15 has been deactivated. The traffic load applied is very low, to avoid network saturation during the time that the change is being assimilated.

The upper plot shows the aggregate amount of SMPs exchanged by the management entities. Approximately 2700 SMPs (corresponding to the transient period) have been delivered before the period of time shown in the figure.

The second plot represents the percentage of updated forwarding tables during the process. It allows us to locate the distribution phase in time.

The last two plots show the effect of the topology change (and its assimilation) over the application traffic. The third plot shows transmission delay for application packets. There is a point in the plot for each packet received. The X-axis represents the time the packet is received, and the Y-axis represents its latency from generation.

The fourth plot represents the aggregate amount of discarded packets. At 61 seconds of simulation the topology change is produced. The deactivation implies that data packets begin to be discarded. Logically, the amount of discarded packets has a direct relationship with network load.

We can appreciate a long period of time (0.17 seconds approx.) between when the change is produced and when the next sweeping process detects it. To reduce this time, it could be useful to conduct a deeper study to tune the sweeping rate according to the topology characteristics. The benefits obtained using a detection mechanism based on traps could also be explored.

Once the change is detected, the topology discovery process involves 200 SMPs during a period of time of 0.01 seconds (see ① in the figure). After that, subnet routes are computed according to the new topology. This process takes more than 0.1 seconds. The graph clearly shows that one of the most important bottlenecks of this mechanism is the paths computation process. A possible optimization could be to reduce the complexity of the algorithm, taking advantage of previous information, or overlapping it with the other tasks.

Finally, the path distribution task (see ② in the figure) implies the distribution of 360 SMPs during a period of time of 0.0067 seconds (faster than the discovery process). During this time, subnet ports are deactivated to avoid deadlock. The consequence is that application traffic delivery is stopped. We can see a gap in the latency plot and an increment of discarded packets in the bottom plot. The final amount of discarded packets (applying a low traffic rate) is greater than 600

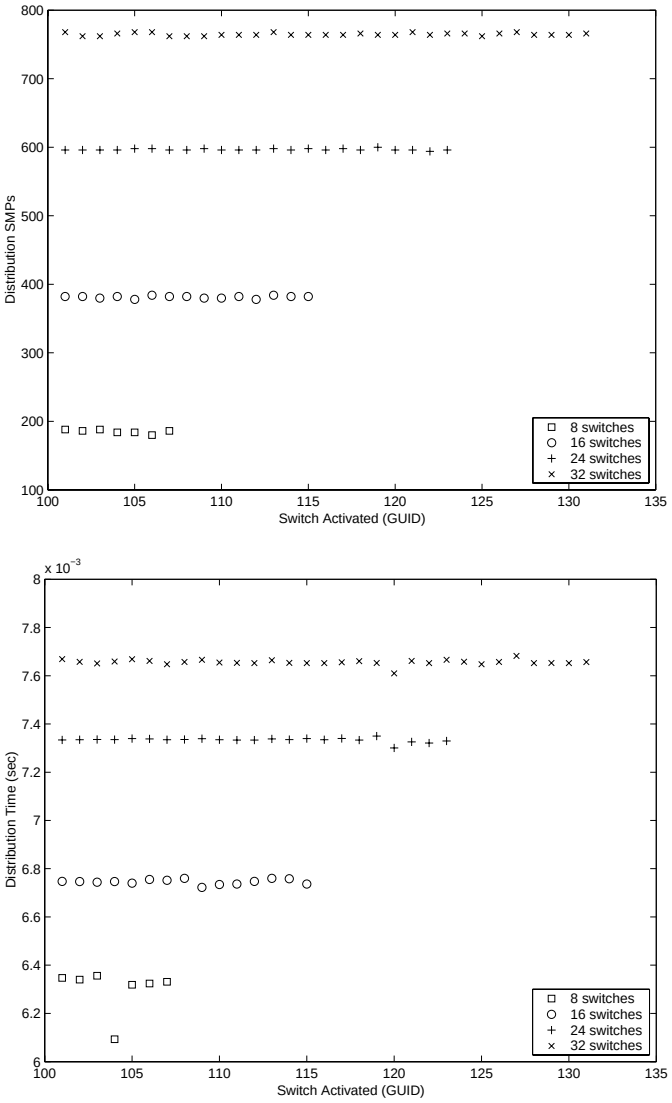

Fig. 27. Analysis of the paths distribution process.

packets. This negative effect could be reduced by the utilization of dynamic reconfiguration techniques as presented in [1][10].

Using additional management statistics, we could analyze in detail the behavior of each management task. As an example, Fig. 27 represents the number of SMPs (top plot) and the time (bottom plot) required by the paths distribution task. In the plots, the simulation parameter is the network size. In this case, for each topology (with 8, 16, 24, and 32 switches) and simulation run, we have programmed the activation of a different switch. The corresponding device identifier (GUID) is represented on the horizontal axis. Again, injection rates are very low, in order to prevent network saturation.

We can see that the amount of SMPs required to distribute the new forwarding tables increases with the network size. Similarly, the time spent by the distribution process has a direct relationship with the network size.

VI. Conclusion

In this paper, we present a way of modeling IBA that is amenable to simulation using OPNET. This provides a useful tool that enables designers of InfiniBand networks to evaluate at the physical and link levels various performance trade-offs of key design parameters. We have shown how this model can be used specifically as a means for designing and evaluating subnet management mechanisms which meet IBA specifications. Thus far, we have modeled a completely functional prototype of IBA's subnet management protocol. Preliminary results provide insight into the overheads resulting from non-optimized implementations and what techniques might be useful in reducing them.

Acknowledgment

The authors thank *Pablo E. García* for his collaboration in modeling many aspects of the architecture.

References

[1] R. Casado, A. Bermúdez, F. J. Quiles, J. L. Sánchez, and J. Duato, "A protocol for deadlock-free dynamic reconfiguration in high-speed local area networks," Special Issue on Dependable Network Computing, *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 2, February 2001.

[2] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal*, no. 32, pp. 406-424, March 1953.

[3] W. T. Futral, *InfiniBand Architecture. Development and Deployment*, Intel Press, August 2001.

[4] IBM InfiniBand products. http://www-3.ibm.com/chips/products/infiniband/

[5] *InfiniBand Architecture Specification* (1.0.a), June 2001, InfiniBand Trade Association. Available: http://www.infinibandta.com/

[6] P. Kermani, and L. Kleinrock, "Virtual cut-though: A new computer communication switching technique," *Computer Networks*, vol. 3, pp. 267-286. 1979.

[7] Lane15 Software, Inc. Austin, TX (USA). http://www.lane15.com/

[8] Mellanox Technologies. Santa Clara, CA (USA). http://www.mellanox.com/

[9] *OPNET Modeler documentation*, OPNET Technologies, Inc. http://www.opnet.com/

[10] R. Pang, T. M. Pinkston, and J. Duato, "The Double Scheme: deadlock-free dynamic reconfiguration of cut-through networks," in Proc. International Conference on Parallel Processing, pp 439-448, Aug. 2000.

[11] J. C. Sancho, J. Flich, A. Robles, P. López, and J. Duato, "Analyzing the influence of virtual lanes on the performance of InfiniBand networks," in Proc. workshop on Communication Architecture for Clusters (held in conjunction with IPDPS '02), April 2002.

[12] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwate, C. P. Thacker, "Autonet: a high-speed, self-configuring local area net-work using point-to-point links," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, October 1991.

[13] C. L. Seitz, "Recent advances in cluster networks," Cluster2001, Newport Beach CA, October 2001.

[14] VIEO, Inc. Austin, TX (USA). http://www.vieo.com/