University of Castilla-La Mancha

A publication of the
Department of Computer Science

Evaluation of a Subnet Management Mechanism for InfiniBand Networks

by

Aurelio Bermúdez, Rafael Casado, Francisco J. Quiles,
Timothy M. Pinkston, and José Duato

Technical Report **#DIAB-03-02-36** March 2003

Submitted to the 2003 International Conference on Parallel Processing (ICPP-03).
Kaohsiung, Taiwan, ROC.

DEPARTAMENTO DE INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete – 02071 – Spain
Phone +34.967.599200, Fax +34.967.599224

Evaluation of a Subnet Management Mechanism for InfiniBand Networks

Aurelio Bermúdez[1], Rafael Casado[1], Francisco J. Quiles[1]
Timothy M. Pinkston[2], and José Duato[3]

[1] Department of Computer Science, Universidad de Castilla-La Mancha,
02071 - Albacete, Spain
{aurelio.bermudez, rafael.casado, francisco.quiles}@uclm.es
[2] SMART Interconnects Group, University of Southern California,
Los Angeles, CA 90089-2562, USA
tpink@charity.usc.edu
[3] Department of Computer Engineering (DISCA),
Universidad Politécnica de Valencia, 40022 - Valencia, Spain
jduato@gap.upv.es

Abstract. The InfiniBand Architecture is a high-performance network technology for the interconnection of processor nodes and I/O devices using a point-to-point switch-based fabric. InfiniBand specification defines a basic management infrastructure that is responsible for subnet configuration, activation, and fault tolerance. Subnet management entities and functions are described, but the specifications do not impose any particular implementation. This paper presents and analyzes a complete subnet management mechanism for this architecture. This work allows us to anticipate future directions to obtain efficient management protocols.

Keywords. System area network, InfiniBand architecture, subnet management, network reconfiguration, performance evaluation.

1. Introduction

The InfiniBand Architecture (IBA) [5, 6] is a new standard for high-speed I/O and interprocessor communication, developed by the InfiniBand Trade Association (IBTA). IBA defines a switch-based network with point-to-point links that supports any topology, including irregular ones, in order to provide flexibility and incremental expansion capability. Recently, the first commercial IBA-compliant products have started to appear in the marketplace [10, 13].

An IBA network is composed of several subnets interconnected by routers, each subnet consisting of one or more switches, processing nodes and I/O devices. Subnets are managed in an autonomous way. There is a subnet management mechanism capable of assimilating any topology change without external intervention, guaranteeing service availability. However, several important questions related to this mechanism have not been addressed in the initial IBA specifications. Instead, they define a general behavior, leaving implementation details to manufacturers and researchers. In particular, the specification does not detail the way in which the subnet topology must be gathered, or the exact procedures to compute and distribute the subnet routes. As we will see, a non-optimized implementation for these key aspects could lead to a degradation in network performance.

Our work focuses on the design of efficient management protocols [4, 12]. In this direction, this paper presents and evaluates a completely functional prototype of a subnet management protocol which meets IBA specifications. This approach covers the detection of topology changes, device discovery and configuration, and computation and distribution of subnet routes. Also, we analyze in detail the behavior and performance of the proposed mechanism, identifying its potential bottlenecks, and exploring various ways in which IBA subnet management can be done more efficiently.

The remainder of this paper is organized as follows. This section presents an overview about the IBA technology and describes the subnet management infrastructure and duties. Section 2 presents our subnet management protocol, detailing the way we have implemented those open aspects. After that, in Section 3, the mechanism is evaluated through several simulation results. Finally, some conclusions are given and future work is proposed in Section 4.

1.1. InfiniBand Architecture Overview

IBA defines a technology for interconnecting processor nodes (hosts) and I/O nodes (I/O units) to form a system area network. The fabric supports a heterogeneous mix of systems with multiple hosts

and I/O units. Each I/O unit can be dedicated to a particular host or shared between multiple hosts. The architecture is independent of the host operating system and processor platform.

Hosts and I/O units 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 nodes 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. Each subnet port has a 16-bit local identifier (LID) assigned by a subnet manager. Switches perform intra-subnet routing using the packet's destination LID. A forwarding table in each switch specifies which port forwards the packet. On the other hand, inter-subnet routing is performed by routers, using global identifiers (GID).

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).

The architecture defines a layered hardware protocol (Physical, Link, Network, and Transport layers) as well as a software layer to manage initialization and communication between devices. Each link can support multiple transport services for reliability and multiple prioritized virtual communication channels.

1.2. InfiniBand Subnet Management

IBA defines a small number of management classes. In particular, the subnet management class specifies methods that enable a *subnet manager* to discover, configure, and manage the subnet. Other

management classes include, for example, subnet administration, device management, and communication management. In this work, we focus on the subnet management class only.

To guarantee compatibility between different vendor implementations, the specification defines different subnet management entities, describing their functions and the structure of the control packets used to exchange information among them. However, as mentioned before, the exact behavior of these management entities has not been detailed.

Subnet management entities are shown in Fig. 1. 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.

Control packets used by the subnet management class are called subnet management packets (SMPs). Each SMP includes exactly 256 bytes of management information. SMPs are sent using the unreliable datagram (UD) class of service, contain a key to authenticate the sender, and use exclusively the management virtual lane (VL15). This VL has priority over data VLs and it is not subject to flow control.

According to the routing mechanism, there are two types of SMPs: *destination (or LID) routed*

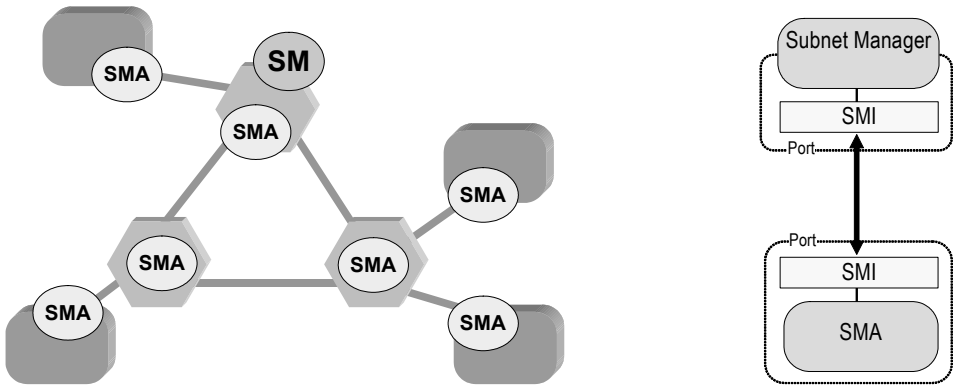


Fig. 1. Subnet management physical (*left*) and logical (*right*) models. The SM can reside in any subnet device (switch, router, or channel adapter). Each node in the subnet contains a SMA. The SMI is associated to an internal management port in switches, or a physical port in the rest of devices.

SMPs and *directed route* SMPs. The former are routed by switches in the same way as data packets. The latter include the sequence of switch output ports to reach the destination. They are primarily used for discovering the physical connectivity of a subnet before it has been initialized.

The SMP header specifies a *method* which indicates the operation being performed by the packet sender (SM/SMA). There are five subnet management methods, whose function will be described in the next section, namely *SubnGet, SubnSet, SubnGetResp, SubnTrap,* and *SubnTrapRepress*. The management information included in a SMP is called the *attribute*. Management attributes are composite structures consisting of components typically representing hardware registers in channel adapters, switches, or routers. Examples of these attributes are *NodeInfo, PortInfo, LinearForwardingTable,* and *Notice*.

The SMI 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 its method. In switches, the SMI implements directed routing, updating packet fields and determining if the current switch is the destination of the SMP. Note that this processing is applied to the directed route SMP in each intermediate SMI. As a consequence, the SMP progresses slower than a destination routed SMP.

SMAs are passive management entities. The tasks performed by the SMA 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, such 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 service level (SL) to VL mapping table, and the VL arbitration tables. In switches, the SMA can (optionally) send traps to the SM to notify that the state of a local port has just changed.

Finally, the SM is the management entity that configures and maintains the subnet operation. Using SMPs, the SM is able to discover the subnet topology, configure subnet ports and switches, and

receive traps from SMAs. There can be multiple SMs, but only one of them can be active. The mastership handover protocol guarantees that only one SM manages the subnet at any given time. This SM is the *master* SM, and the rest of SMs are the *standby* SMs. Standby SMs monitor the health of the master SM and, if it goes down, they negotiate among themselves on who will become the successor. From now on, we denote 'SM' to refer to the master SM.

2. Subnet Management Mechanism

Examples of commercial management products can be found in [7, 17]. Also, IBTA technical working groups are currently defining management aspects not detailed in the IBA specification. Unfortunately, neither a comprehensive description nor a performance evaluation comparison with these mechanisms is available at this time.

In the management mechanism presented and analyzed here, SM tasks are sequentially executed (see Fig. 2). Once the SM becomes the master, it periodically sweeps the subnet searching for topology changes. After a change is detected, it must obtain the new subnet topology. The topology discovery process is centralized in the SM, which collects the subnet topology starting from scratch and performing a propagation-order exploration. Once the exploration finishes, the SM uses the topological information previously collected to determine the routes through the subnet, and applies the *up*/down** routing algorithm [16]. Finally, in order to prevent deadlock situations during the distribution of

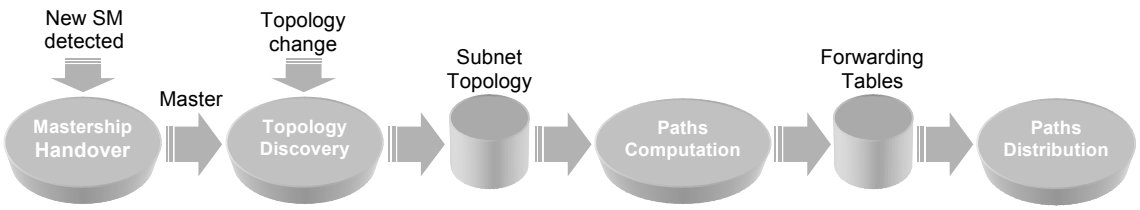


Fig. 2. Sequence of tasks performed by the SM. The topology discovery task obtains the subnet topology used by the paths computation task. The paths computation task computes the forwarding tables that the distribution task must deliver to the switches.

forwarding tables, static reconfiguration is assumed. This means that user traffic is stopped while forwarding tables are being updated. Next, these management tasks are detailed.

2.1. Change Detection

When a topology change occurs, the state of at least one subnet port changes. The SM is in charge of detecting this change. We have considered both detection mechanisms defined by the IBA specification: subnet sweeping and traps.

First, the SM is responsible for periodically polling the subnet to gather information about topology changes. The sweeping rate must be tuned according to parameters such as the subnet size or the response time required. In particular, the SM establishes a communication with each subnet switch, examining all its ports and searching for possible changes of state. To this end, it sends a *SubnGet(PortInfo)* SMP for each switch port. To speed up this process, the IBA specification allows the use of only one SMP to detect global changes in the switch. In this case, the SM sends only a *SubnGet(SwitchInfo)* SMP for each subnet switch. We have implemented this sweeping method.

In addition to sweeping, a switch SMA may optionally inform to the SM about the change of state in a local port by sending a *SubnTrap(Notice)* SMP. Moreover, the SMA could periodically repeat the trap message until it receives a notification from the SM to stop the trap sending with a *SubnTrapRepress(Notice)* SMP.

2.2. Topology Discovery

The first step in assimilating a topology change consists of determining the current subnet topology. Note that subnet activation is a particular case of topology change. For the shake of simplicity, our mechanism will always obtain the complete subnet topology, ignoring all the previously collected information.

The IBA specification does not detail any implementation for the subnet discovery algorithm. It only states that the SM shall send repetitive SMPs to identify all active nodes (and SMs) in the subnet. In particular, the SM uses *SubnGet(PortInfo)* SMPs to obtain information about each port in a subnet node, and *SubnGet(NodeInfo)* SMPs to determine the nature (switch, router, or channel adapter) of the device at the other end of an active port. All these SMPs must use directed routing, because switch forwarding tables have not been distributed yet.

During the discovery process, the SM assigns LIDs to the discovered devices, and configures other port attributes, sending *SubnSet* SMPs to the nodes. For example, the SM must notify its own LID to a new node, in order to allow the traps mechanism to function correctly.

We have implemented a propagation-order exploration [14] to perform a search over the graph used for modeling the subnet. In this case, exploration SMPs spread throughout the subnet in an uncontrolled way. The SM sends new SMPs as it receives responses to previous SMPs from the subnet SMAs. Alternative exploration methods could perform a controlled breadth-first or depth-first search.

To start the discovery process, the SM sends a first *SubnGet(NodeInfo)* SMP to the local node (using an empty directed path), and waits for a response. Each time the SM receives a response (*SubnGetResp* SMP) from a previous request, it executes a block of code similar to the following:

```
If attribute_id = NodeInfo then
  If sender_node not visited then
    Assign a LID to sender_node
    For each port in sender_node do
      Send SubnGet(PortInfo)     ; (use the same path)
    EndFor
  EndIf
ElseIf attribute_id = PortInfo then
  If management port then        ; send the LID to the SMA
    Send SubnSet(PortInfo)       ; (use the same path)
  EndIf
  If port_state <> DOWN then     ; discover a new device
    Send SubnGet(NodeInfo)       ; (add a port to the path)
  EndIf
EndIf
```

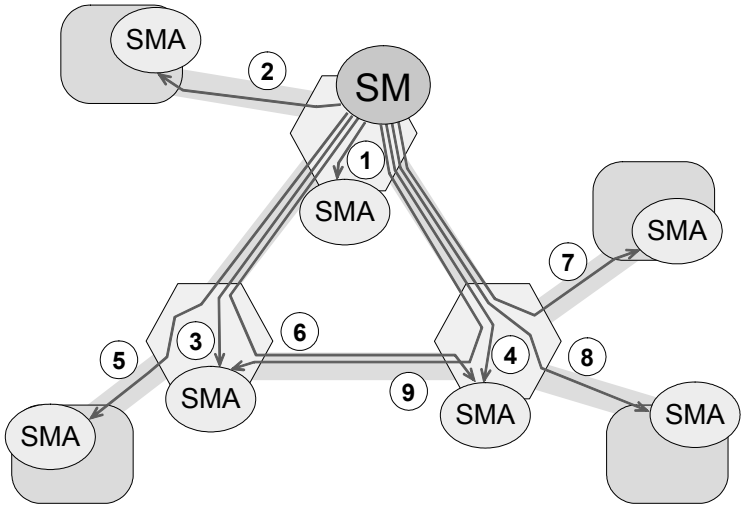


Fig. 3. Example of subnet discovery process. Each arrow represents a *SubnGet(NodeInfo)* SMP sent by the subnet SM, and the associated number indicates the order in which it is generated. After receiving the response to *1*, the SM sends new SMPs (labeled as *2*, *3*, and *4*) through all its active ports. Similarly, SMPs *5*-*6* and *7*-*8*-*9* are generated after processing the responses to *3* and *4*, respectively. Note that it is not necessary to send new SMPs after receiving the responses to *6* and *9*, because the source devices have been previously visited.

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

To illustrate this process, Fig. 3 shows a possible sequence of SMPs used by the SM to discover the topology for the subnet in Fig. 1(*left*), assuming that all devices are active. Note that the concrete discovery order is not deterministic, instead, it depends on the ordering of the responses received from the SMAs. For the shake of clarity, the figure only represents the *SubnGet(NodeInfo)* SMPs sent by the SM. Neither the *SubnGet(PortInfo)/SubnSet(PortInfo)* SMPs nor the *SubnGetResp* SMPs have been included.

2.3. Paths Computation

Using the current topological information, the SM must establish the subnet paths that data packets will use to reach their destination. In other words, it is necessary to compute the set of subnet

forwarding tables. The IBA specification does not impose any specific routing algorithm for the computation of routing tables.

The up*/down* routing algorithm is a popular deadlock-free algorithm valid for any topology. This algorithm is based on a cycle-free assignment of direction to the operational links in the network. For each link, a direction is named *up* and the opposite one is named *down*. To avoid deadlocks, legal routes never use a link in the up direction after having used one in the down direction.

Unfortunately, the up*/down* routing algorithm cannot be used in IBA subnets, because it may lead to deadlock [9]. It is because the up*/down* routing function takes into account the packet input port and destination node, and IBA switches only consider the packet destination LID for routing packets. The reason is that this drastically reduces the forwarding table size.

There are several proposals that allow using the up*/down* routing algorithm to compute the IBA routing tables [9, 15]. Our mechanism uses a simple (non-optimal) approach. First, using the above rule we compute all the valid up*/down* routes. After that, we remove those routing alternatives that could lead to a deadlock situation. The criterion is that if for a given destination there is any output port in the down direction, we ignore all the routing options that imply the use of a link in the up direction. Finally, we must choose only one output port for each destination, because IBA routing is deterministic. The criterion we have used is to select the shortest route.

It is important to note that the SM is also responsible for computing (and configuring) other switch and channel adapter internal tables, such as the VL arbitration table and the SL-to-VL mapping table. The computation of these tables is out of the scope of our work, and can be intended, for example, for providing QoS in IBA networks [1].

2.4. Paths Distribution

The SMPs for updating switch forwarding tables are completely defined in the IBA specification. However, the update order is not detailed. Updating the switch tables in an uncontrolled way could generate deadlock situations [4]. The reason is that although the new and the previous sets of subnet routes are deadlock-free, the coexistence of both routing schemes during the distribution process is not necessarily deadlock-free.

Traditional reconfiguration mechanisms [3, 16] solve this problem by preventing the existence of data packets in the network during the process. This approach is called static reconfiguration. Alternatively, in [4, 12] we have proposed two deadlock-free reconfiguration schemes that allow traffic through the network while the routing tables are being updated. Nevertheless, the subnet management protocol we have currently developed uses the static approach to distribute the set of subnet forwarding tables.

The reconfiguration process is controlled by the SM, which must deactivate all subnet ports before starting the distribution of tables. Once the new forwarding tables have been completely distributed, the subnet is activated again. SMPs used to perform the two first steps (subnet deactivation and distribution of tables) must use directed routing. On the other hand, SMPs for the subnet reactivation phase can use either directed or destination routing.

In particular, the SM sends a *SubnSet(PortInfo)* SMP to change the state of each subnet port to *INITIALIZE*. In this state, the port can only receive and transmit SMPs, discarding all other packets received or presented to it for transmission [5]. In the same way, to reactivate the service when the tables have been distributed, the SM sets the state of the subnet ports to *ACTIVE*. In this state, a port can transmit and receive all packet types. The delivery of routing tables is performed using either *SubnSet(LinearForwardingTable)* SMPs or *SubnSet(RandomForwardingTable)* SMPs.

3. Performance Evaluation

All the performance results presented in this work have been obtained by using simulation techniques. Before showing and analyzing simulation results, we describe the simulation methodology.

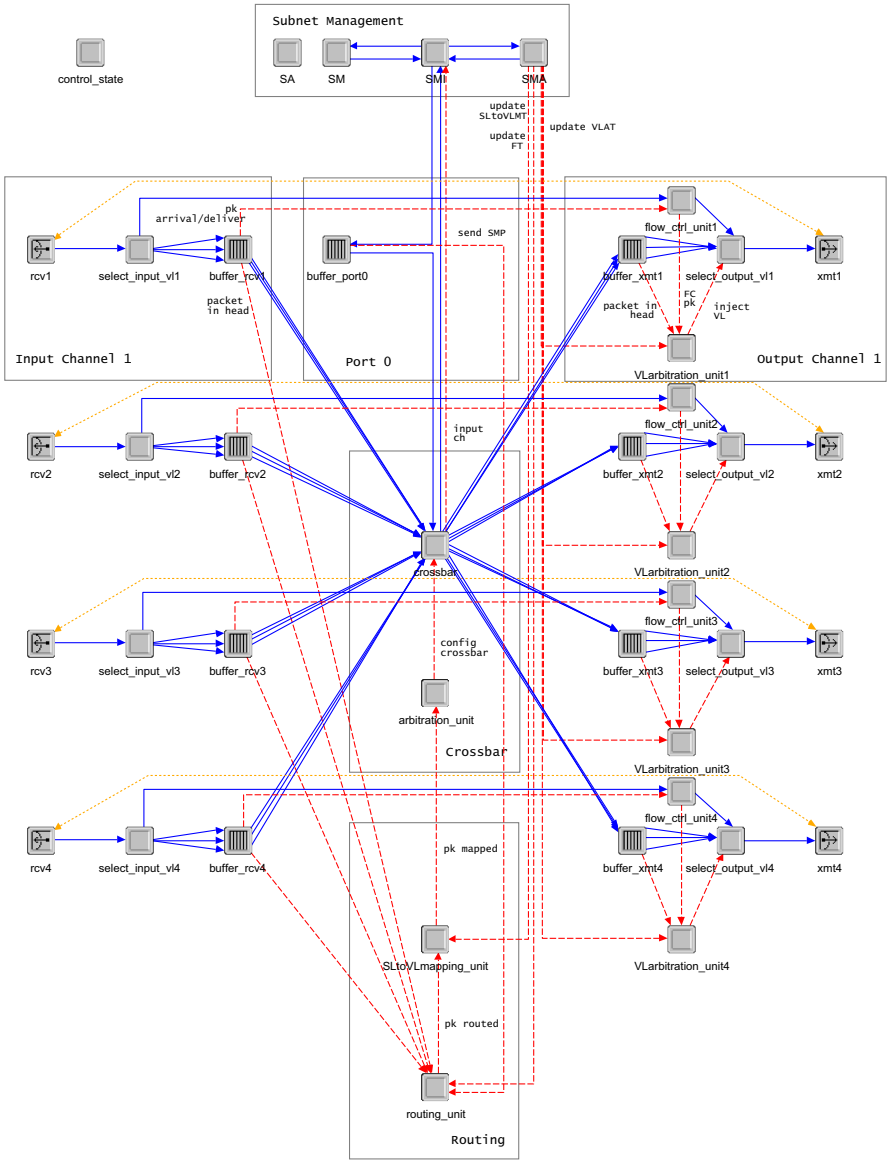


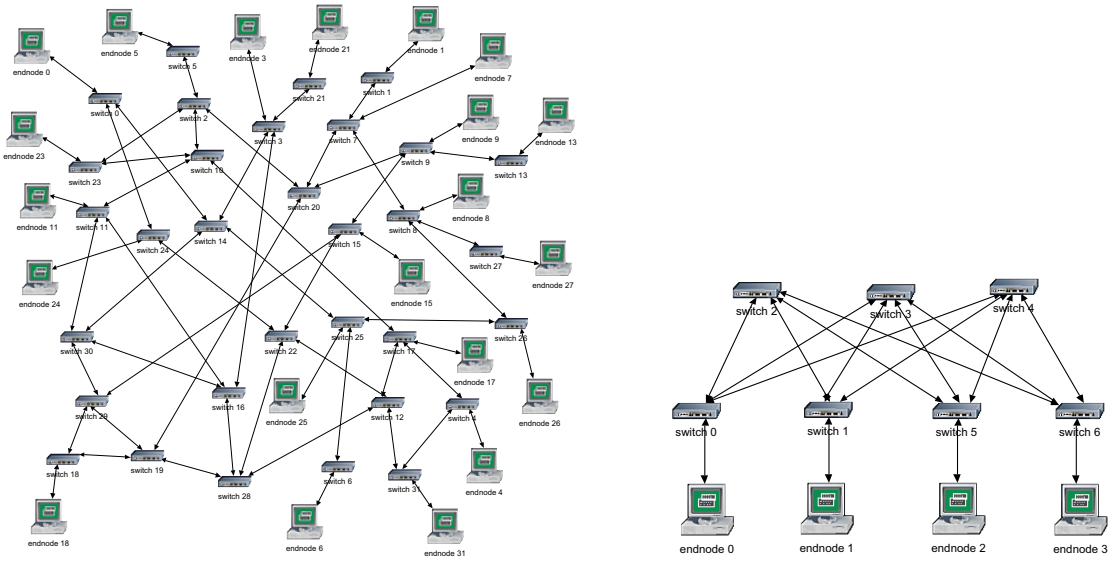
Fig. 4. Internal structure of a 4-port switch. Each input channel contains a point-to-point receiver, a demultiplexer and a set of input buffers associated with VLs. The packet routing logic includes the routing unit and the SL to VL mapping unit. The switching logic (in the center) is composed of a fully demultiplexed crossbar and the arbitration unit. Each output channel incorporates a set of output buffers, a multiplexer, the flow control unit, and the channel arbitration unit. Finally, at the top of the figure there are several modules modeling the subnet management entities. These entities are connected to the crossbar through an internal port.

3.1. Simulation Methodology

Our InfiniBand model [2] embodies key physical and link layer features of IBA, allowing the simulation of various IBA-compliant network designs. To develop it, we have used the OPNET Modeler [11] simulation software. OPNET is a powerful engineering and research tool for streamlining the design and performance analysis of communication systems and protocols.

The current IBA model is composed of 1X links, 4-port fully demultiplexed switches, and end nodes containing a HCA (hosts). As an example, Fig. 4 shows the switch model we have implemented. See [2] for more details.

For this work, we have used a set of randomly generated irregular topologies, such as the one shown in Fig. 5(*a*). We have evaluated subnets with 8, 16, 24, and 32 switches. We assume that there is a host connected to each switch, if a port is available. Also, not all switch ports are connected. Apart of irregular topologies, we have analyzed non-arbitrarily generated topologies, as the one shown in Fig. 5(*b*). This in an example of clos or fat-tree network, widely used in SAN and IPC networks.



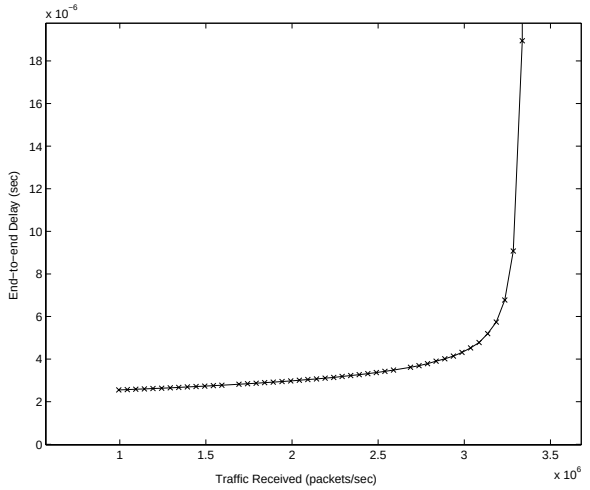
(*a*) (*b*)

Fig. 5. (*a*) Irregular subnet topology composed of 32 switches and 21 hosts. (*b)* Clos topology composed of 7 switches and 4 hosts.

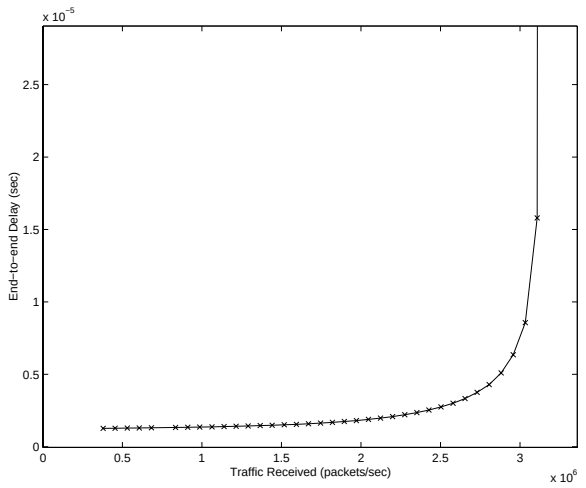
In all cases, the amount of operational data VLs per subnet port is 2 (VL0-1). Physical links are assigned to data VLs using a round robin strategy. The size of the input and output buffers associated to each VL is 4,096 bytes. Each subnet switch supports a linear forwarding table with 1,024 entries. For SL mapping, a cyclic assignment of VLs is considered.

The application traffic pattern is very simple. The traffic load is defined by the packet length and generation rate. The model is completed with the destination and SL distributions. As packet length, we have considered a maximum transfer unit (MTU) of 256 bytes (the minimum MTU value allowed by the IBA specification). The generation rate is uniform, and it is expressed in packets/sec/node. Traffic sources also use a uniform distribution to obtain the packet destination (among all the active hosts) and SL value (from 0 to 15). The traffic load applied is different for each subnet topology.

After obtaining subnet performance, we have selected a low load value for the analysis of switch addition and removal, in order to prevent network saturation during this analysis. For example, the plot in Fig. 6(*a*) represents the packet delay versus network throughput obtained for the topology in



(*a*) Irregular topology in Fig. 5(*a*)



(*b*) Clos topology in Fig. 5(*b*)

Fig. 6. Subnet performance, assuming 2 operational data VLs per port, a packet buffer size of 4,096 bytes, and a packet MTU of 256 bytes.

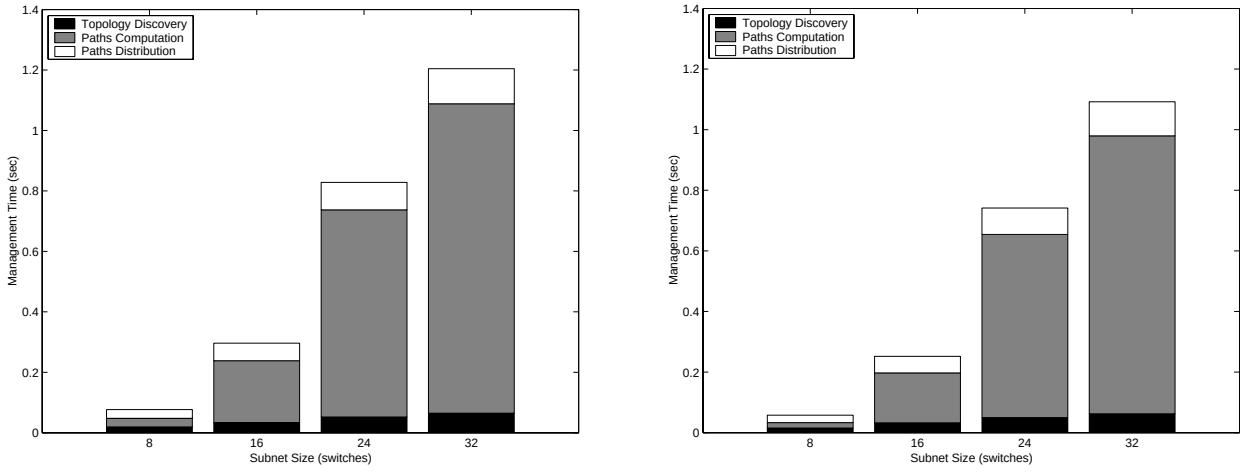
Fig. 5(*a*). In this case, we have considered a packet generation rate of 41,250 packets/sec/node, representing the 25% of the saturation rate observed (165,000 packets/sec/node). Similarly, we have used the results in Fig. 6(*b*) to determine the traffic load for the subnet in Fig. 5(*b*).

For each simulation run, we have programmed the subnet activation at time 10 sec. Traffic sources in hosts begin to generate packets at time 60 sec. After a transient period, a topology change, consisting of the addition or removal of an individual switch, is simulated. In all cases, subnet sweeping rate is set to 0.1 sec, and traps support is disabled in switch SMAs. This experiment has been repeated for each switch in the subnet. Average values are shown in the figures, except for the plots related to instantaneous values.

3.2. Simulation Results

In this section, we present several plots to analyze the behavior of the subnet management mechanism proposed in this paper. We study the time and the amount of control packets required to assimilate a change. Moreover, we are especially interested in evaluating the effects of the protocol over application traffic.

Fig. 7 shows the time required by the mechanism to completely assimilate a topology change,



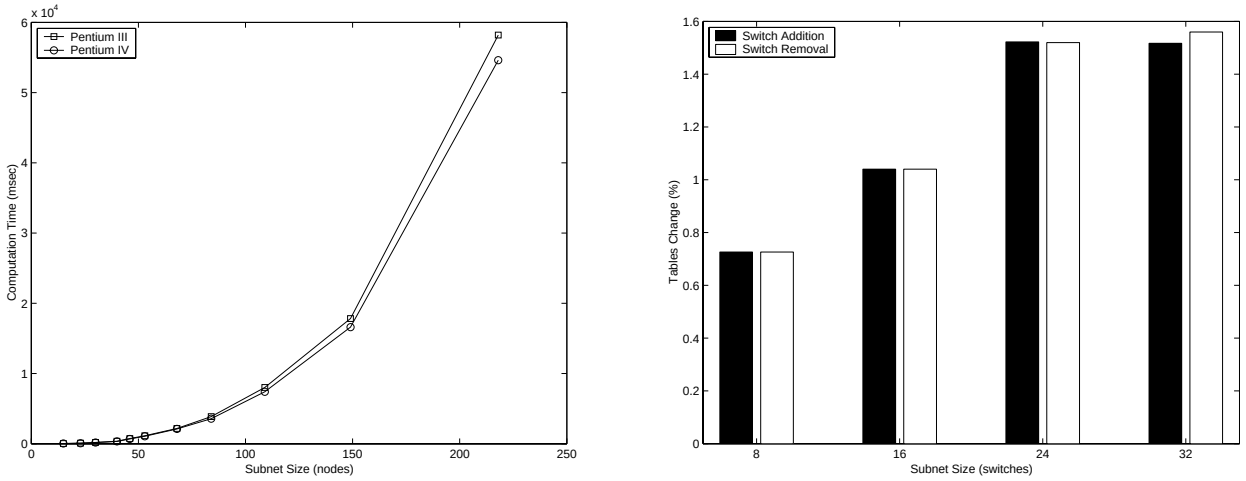
(*a*) Switch addition (*b*) Switch removal

Fig. 7. Time required by the subnet management mechanism to assimilate a change as a function of subnet size.

once it has been detected. Fig. 7(*a*) shows the case for switch addition and Fig. 7(*b*) presents the results for switch removal or failure. Note that the represented value is the sum of the time spent by the successive management tasks (discovery, computation, and distribution).

In general, switch addition requires more time than switch removal. The reason is that subnet size is larger in case of activation (two additional subnet nodes). Also, the plots clearly show that most of the time spent by this mechanism is in the path computation process. Results for non-arbitrary tologies are similar.

Moreover, the fraction of time required to compute the paths increases with subnet size. Fig. 8(*a*) shows the time required to build the set of subnet forwarding tables, considering two different PC architectures. This figure clearly shows that the time required to compute the forwarding tables increases quadratically with subnet size. This result is expected since the size of each table increases linearly with subnet size. Results also show that the management processor performance does not have a significant influence on computation time, mostly because path computation involves integer



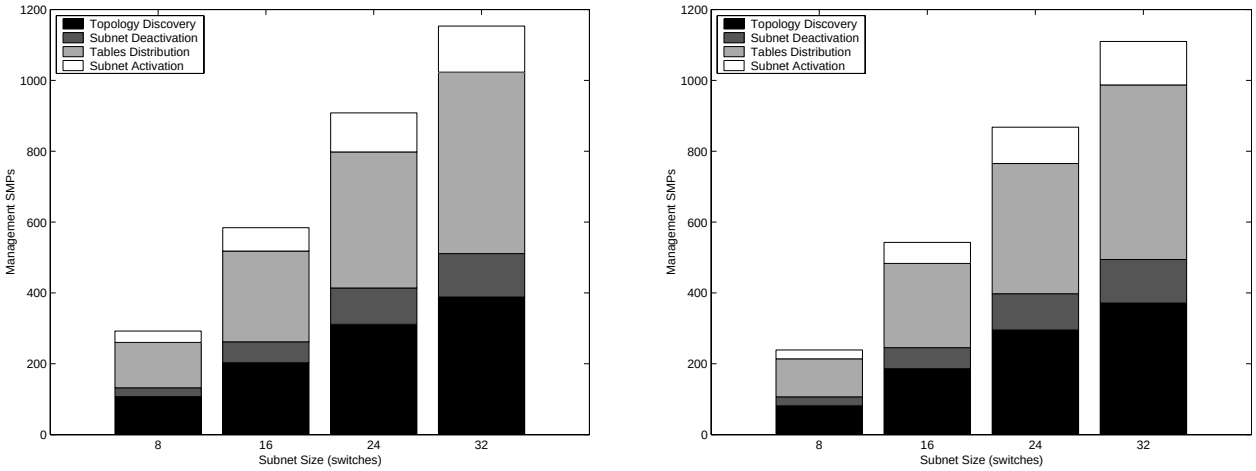
(*a*) (*b*)

Fig. 8. (*a*) Time required to compute the subnet forwarding tables for different subnet sizes. The values have been empirically obtained by executing our IBA model on both an Intel Pentium III (1.06 GHz) and an Intel Pentium IV (1.5 GHz) microprocessors. (*b*) Average percentage of table entries that must be updated after a topology change.

arithmetic. A possible optimization could be to reduce the complexity of the algorithm used to compute the forwarding tables, taking advantage of the topological information available before switch addition/removal. In particular, Fig. 8(*b*) shows that a very small percentage of forwarding table entries is affected by a topology change (less than 2%). Therefore, it would be possible to derive an algorithm that only computes the forwarding tables that suffered changes. We plan to analyze this issue in future work.

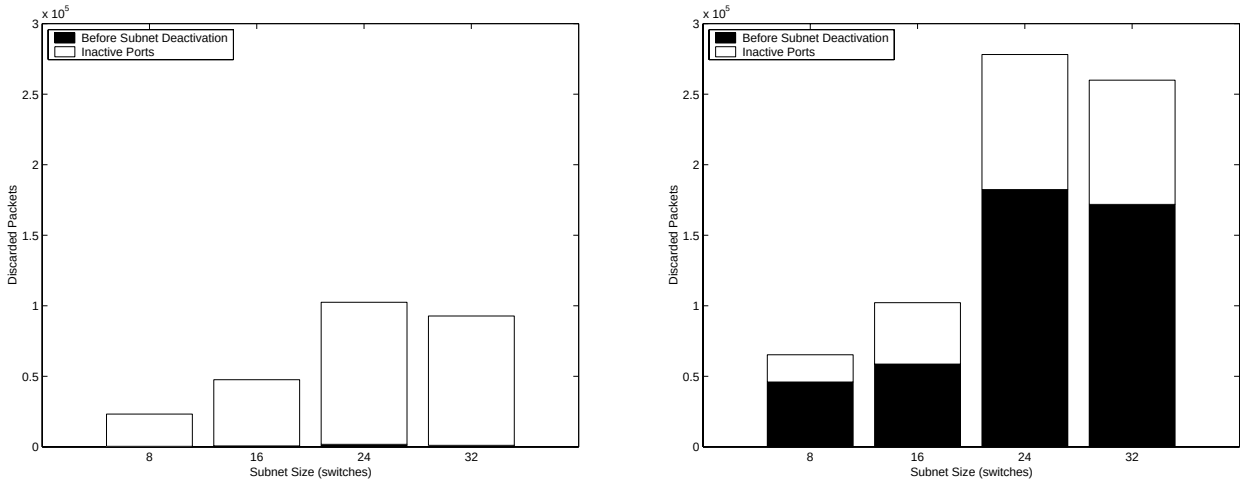
Fig. 9 represents the total number of SMPs required by the subnet mechanism, considering separately the contribution of each management process. Most of the SMPs correspond to the discovery task and the distribution of forwarding tables (during the paths distribution process). We have obtained the same results for non-arbitrary tologies. An implementation that takes advantage of the previous configuration could considerably reduce the number of SMPs that the SM must send out.

Fig. 10 shows the amount of data packets that are discarded during the change assimilation, as a function of subnet size. There are two main causes of packet discarding. One is that, according to the static reconfiguration process, all the packets generated by the hosts during the distribution of



(*a*) Switch addition (*b*) Switch removal

Fig. 9. SMPs required by the subnet management mechanism as a function of subnet size. Note that the paths computation task does not involve the use of SMPs. Once again, switch addition implies more SMPs than removal.



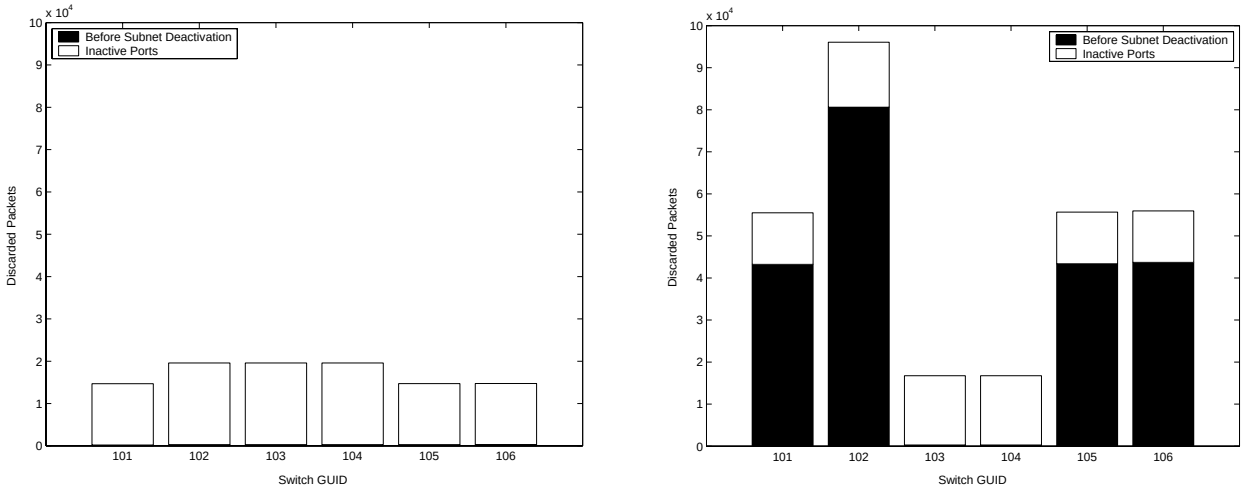
(*a*) Switch addition (*b*) Switch removal

Fig. 10. Number of packets discarded during the change assimilation as a function of subnet size. The amount of packets discarded by inactive ports is similar in both cases (switch addition and removal), although it is slightly larger in case of addition.

forwarding tables must be discarded; the reason is that subnet ports are inactive (they only allow SMPs). The second cause occurs for the case of switch removal: packets stored in the deactivated switch buffers and packets that use this switch to reach their destinations (according to the old routing tables) will be dropped. This source of packet discarding dominates the other, and increases, in general, with subnet size and traffic load. The limitation of IBA in restricting routes to be deterministic results in these packets having no other alternative but to be dropped. Discarded packets are not reinjected into the network. We assume that this function is performed by upper layer mechanisms.

The massive discarding of packets in case of switch removal is inevitable. However, we can reduce this effect by reducing the time required to discover the subnet topology and, especially, the time spent on computing the forwarding tables.

Fig. 11 shows the same statistics as Fig. 10, obtained in this case for the clos topology composed of 7 switches and 4 hosts shown in Fig. 5(*b*). Results are very similar. However, we can observe a particular behavior when we simulate the removal of switches *3* (GUID *103*) and *4* (GUID



(*a*) Switch addition (*b*) Switch removal

Fig. 11. Number of packets discarded during the change assimilation, considering the clos topology in Fig. 5(*b*). Each bar in the plots corresponds to a single simulation run. Horizontal axis represents the globally unique identifier (GUID) of the switch that is added/removed from the subnet. The SM is hosted at switch *0*. We assume that GUID *101* corresponds to switch *1*, GUID *102* corresponds to switch *2*, and so on.

104). In both cases, the amount of packets that are discarded before the subnet deactivation is very small. The reason is that subnet routes do not use these switches. Instead, they employ switch *2*.

Finally, to analyze the instantaneous behavior of the evaluated management mechanism, Fig. 12 shows some results obtained from an irregular subnet composed of 8 switches and 7 hosts. The topology change consists of a switch removal (at time 60.1 sec). For all plots, the X-axis represents the simulation time.

The top plot shows the aggregate amount of SMPs exchanged by the management entities. This plot allows us to identify the different tasks in the management process, because the two *steps* in the plot correspond with the discovery and distribution phases, respectively. Before that, we can appreciate a long period of time (0.08 seconds approx.) between when the change is produced and when the sweeping process detects it (and the discovery process begins). Obviously, the detection period could be reduced using a different sweeping rate, or enabling the use of traps. The second plot shows the

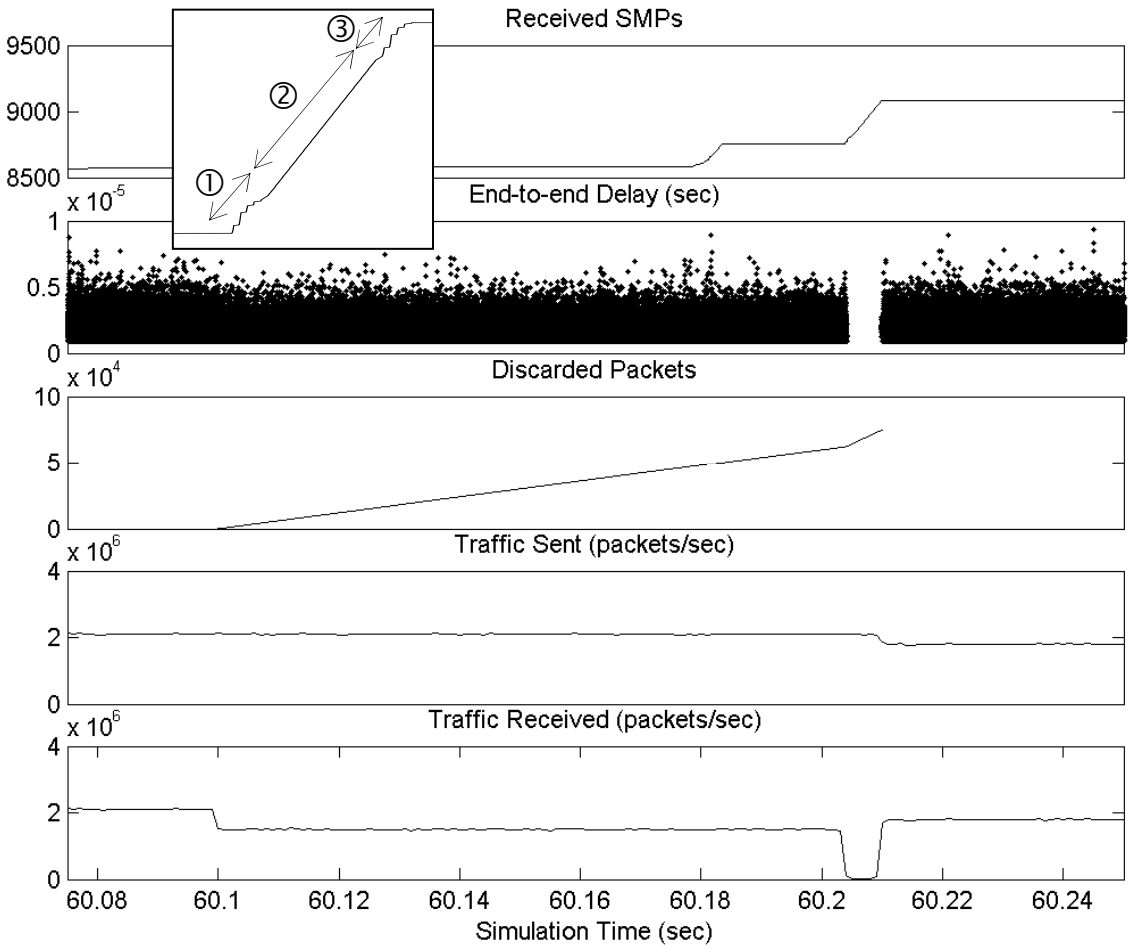


Fig. 12. Impact of a switch removal (or failure) on application traffic, for a subnet with 8 switches and 7 hosts. Packet generation rate is 300,000 packets/sec/node. The box in upper left-hand corner shows a magnification of the SMPs corresponding to the distribution process, allowing us to identify the three steps of the process (subnet deactivation ①, tables distribution ②, and subnet reactivation ③).

latency (from generation) for each received data packet. The third plot represents the aggregate amount of discarded packets during the simulation. As the change considered is a switch removal, packet discarding begins exactly at time 60.1 sec. The two last plots show instantaneous network throughput, through the number of packets sent and received per second in the whole subnet.

During the distribution of forwarding tables, subnet ports are deactivated. As a result, application traffic delivery is stopped. We can see a gap in the latency and traffic received plots (0.005 seconds approx.) and an additional increment in the number of discarded packets. The final amount of

discarded packets exceeds 75,000 packets. These negative effects (lack of service and packet discarding due to inactive ports) could be reduced by the utilization of dynamic reconfiguration techniques as presented in [4, 12] and/or reducing the amount of information to distribute after the topology change.

4. Conclusions and Future Work

This paper describes a whole functional prototype of subnet management protocol. It has required the previous definition of many design issues not covered by the IBA specifications. The proposed subnet manager is able to detect topology changes and to configure subnet devices according to the new topology in an autonomous and deadlock-free way. We have modeled and analyzed our design using OPNET. Instead of implementation over real components, simulation provides researches a more flexible and controlled environment to develop novelty proposals.

The obtained results clearly state that the main bottleneck of this management mechanism is the forwarding tables computation process. In particular, a sequential computation of forwarding tables is too slow. There are several ways to speed up this process. One of them is the addition of a pre-processing step to compare old and new network graphs, extracting only those routes that are likely to change, and computing new forwarding tables for only those. Obviously, the route selection process must be faster than the forwarding tables computation process. Other possibility is performing a distributed computation of forwarding tables. The IBA specifications allows a distributed implementation of the SM, instead of a centralized one. In this case, several replications of this entity may provide forwarding tables to their neighboring nodes in parallel. The third idea is to overlap in time the discovery, FT computation and distribution processes, partially shadowing the overhead of the second one. Apart of FT computation, we are considering the improvement of the other tasks performed by the SM. Related to the discovery process, we plan to provide the directed routed SMPs

with a initial destination routed segment, reducing the overhead due to the SMI interface. Another possibility is to explore only the region that has changed (as in the *skyline* approach [8]) instead of the entire topology. On the other hand, IBA specifications states a mechanism to support automatic path migration. In this process, a channel adapter signals another CA to migrate the connection they establish to the predefined alternate path. The application of this mechanism can sensibly reduce packet discarding during the assimilation of a topology change. As future work, we plan to include and evaluate these ideas in our management mechanism.

References

1. Alfaro, F. J., Sánchez, J. L., Duato, J., Das, C. R.: A strategy to compute the InfiniBand arbitration tables. In Proc. 2002 International Parallel and Distributed Processing Symposium. Ft. Lauderdale, Florida (USA), April 2002
2. Bermúdez, A., Casado, R., Quiles, F. J., Pinkston, T. M., Duato, J.: Modeling InfiniBand with OPNET. In Proc. 2nd Annual Workshop on Novel Uses of System Area Networks. Anaheim, CA (USA), February 2003
3. Boden, N. J., Cohen, D., Felderman, R. E., Kuawik, A. E., Seitz, C. L., Seizovic, J. N., Su, W.: Myrinet: a gigabit per second LAN, IEEE Micro, vol. 15, no. 1, pp. 29-36, February 1995
4. Casado, R., Bermúdez, A., Quiles, F. J., Sánchez, J. L., Duato, J.: A protocol for deadlock-free dynamic reconfiguration in high-speed local area networks. IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 2, pp. 115-132, February 2001
5. InfiniBand Architecture Specification (1.0.a), June 2001, InfiniBand Trade Association. http://www.infinibandta.com/
6. Futral, W. T.: InfiniBand Architecture. Development and Deployment, Intel Press, August 2001
7. Lane15 Software, Inc. Austin, TX (USA). http://www.lane15.com/

8. Lysne, O., Duato, J.: Fast dynamic reconfiguration in irregular networks. In Proc. 2000 International Conference on Parallel Processing, August 2000
9. López, P., Flich, J., Duato, J.: Deadlock-free routing in InfiniBand™ through destination renaming. In Proc. 2001 International Conference on Parallel Processing, September 2001
10. Mellanox Technologies. Santa Clara, CA (USA). http://www.mellanox.com/
11. OPNET Technologies, Inc. http://www.opnet.com/
12. Pinkston, T. M., Zafar, B., Duato, J.: A method for applying Double Scheme dynamic reconfiguration over InfiniBand. USC Technical Report, March 2002
13. RedSwitch, Inc. Milpitas, CA (USA). http://www.redswitch.com/
14. Rodeheffer, T. L., Schroeder, M. D.: Automatic reconfiguration in Autonet, SRC Research Report 77 of the ACM Symposium on Operating Systems Principles, October 1991
15. Sancho, J. C., Robles, A., Duato, J.: Effective strategy to compute forwarding tables for InfiniBand networks. In Proc. 2001 International Conference on Parallel Processing, September 2001
16. Schroeder, M. D., Birrell, A. D., Burrows, M., Murray, H., Needham, R. M., Rodeheffer, T. L., Satterthwate, E. H., Thacker, C. P.: 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
17. VIEO, Inc. Austin, TX (USA). http://www.vieo.com/