Implementing the Advanced Switching Fabric Discovery Process*

Antonio Robles-Gómez[1], Aurelio Bermúdez[1], Rafael Casado[1],
and Francisco J. Quiles[1]

[1]Universidad de Castilla-La Mancha
Instituto de Investigación en Informática (I3A)
02071 – Albacete, Spain
{arobles, abermu, rcasado, paco}@dsi.uclm.es

Abstract

Advanced Switching is a new high-speed industrial standard serial interconnect. It is defined as a switching fabric architecture based on the PCI Express technology. The Advanced Switching specification establishes a management infrastructure which maintains the fabric operation. The topology discovery process is triggered after fabric initialization and every time a topological change is detected. The information gathered by this process is used to build a set of paths between fabric endpoints. This work analyzes the performance of several possible implementations for this management task.

1. Introduction

The Advanced Switching (ASI) technology has been recently proposed as a standard for future interconnects [4, 10]. The ASI specification [1] has been developed by the Advanced Switching Interconnect Special Interest Group (ASI-SIG). It is a chip-to-chip and backplane interconnect switched fabric architecture.

In order to support high availability, ASI includes important features, such as device hot addition and removal, redundant pathways, and fabric management failover. In particular, the specification provides a fabric management mechanism, which basically configures and monitors the status of the network. Every time a topological change is detected (for example, a failure in a network device), this mechanism must discover the resulting topology. After that, a new set of routes must be obtained and distributed to the fabric endpoints. All these tasks are performed by the fabric manager (FM), a software entity running on one or more ASI endpoints.

The internal behavior of the management mechanism is currently an open issue for vendors and researchers. The ASI specification only considers a set of configuration data structures into each device, and the management packets used to access those structures.

Obviously, reducing the time required to completely assimilate a change will minimize its negative impact on application traffic. Some examples of this impact are packet losses, network congestion, and increment of latency. Alleviating these effects is the final goal of our work.

In this paper the focus is on the first management task after the detection of the change; the discovery process. The ASI specification does not detail the way in which the FM must obtain the fabric topology. It only states that repetitive discovery packets must be sent in order to identify all active devices in the fabric. In this way, the FM builds a graph of the fabric topology and learns the configuration of each node.

Nevertheless, the ASI-SIG developers have recently proposed a serialized discovery algorithm [11]. In this work, we propose and comparatively analyze two alternative parallel implementations for this process. As we will see, one of them significantly improves the serialized proposal.

This paper is organized as follows. First, Section 2 briefly introduces the ASI architecture and the fabric management support provided by the specification. Then, Section 3 describes the three mentioned implementations for the fabric discovery process. After that, Section 4 presents a detailed performance evaluation of each implementation. Finally, Section 5 gives some conclusions and describes our future work.

*This work was partly supported by the following projects: CSD2006-46 and TIN2006-15516-C04-02 (Ministerio de Educación y Ciencia), and PBC05-007-1 (Junta de Comunidades de Castilla-La Mancha). It was also supported by an FPI grant (TIC2003-08154-C06-02 – Ministerio de Educación y Ciencia).

2. The Advanced Switching Architecture

ASI can be seen as the next step in the evolution of the traditional PCI bus. In particular, it uses the PCI Express [7] physical and link layers, differing at the transaction layer. ASI provides enhanced support for features such as flexible protocol encapsulation, peer-to-peer transfers, multicast transfers, and QoS.

An ASI network connects multiple endpoints by means of a switched serial fabric. Endpoints support up to 4 ports, and switches support up to 256 ports. The specified base link bandwidth is 2.5 Gbps. However, effective bandwidth is reduced to 2.0 Gbps by 8b/10b encoding.

The specification establishes three types of virtual channels: unicast bypassable (BVC), unicast ordered (OVC), and multicast (MVC). Each BVC implements an ordered queue and a bypass queue. Packets marked as "bypassable" are delivered to the bypass queue, and can be "bypassed" by other packets at the ordered queue. On the other hand, OVCs and MVCs only support ordered queues.

A traffic class (TC) mechanism allows to group flows of traffic for similar treatment. The traffic class of a packet is defined at the source endpoint, and included at the packet routing header. When a packet reaches a port, this value is used to obtain the corresponding VC, by using a set of fixed TC/VC mapping tables.

In order to simplify the hardware, ASI states that unicast packets use source routing. Endpoints include path information into the packets, by filling up the *Turn Pool*, *Turn Pointer*, and *D* (direction) fields in the routing header (see Figure 1). These fields are used at each intermediate switch to obtain the output port. On the other hand, multicast packets require looking up into a specific forwarding table.

ASI defines several mechanisms for congestion management. First, link layer uses the credit-based flow control defined by the PCI Express architecture. Additional optional congestion mechanisms are status-based flow control, minimum bandwidth scheduler, and endpoint source injection rate limiting.

ASI also establishes a mechanism to encapsulate packets of any upper-layer protocol. In particular, the *PI* (Protocol Interface) field in the packet routing header identifies the nature of the encapsulated information. This allows an ASI fabric to concurrently carry an indeterminate number of independent data protocols.

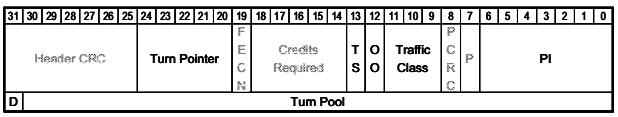


Figure 1. ASI packet routing header.

2.1. ASI Fabric Management

Fabric management [10] is a set of functions, activities, and tasks that may include any or all of the following operations among many others: fabric discovery, path determination between endpoints, local and distributed connection management, multicast group management, bandwidth management, dynamic device addition and removal, fabric supervision, and APIs and data-structure elements for upper level, operating-system support.

After the fabric is powered up, a distributed process is triggered in order to select primary and secondary fabric managers. Only these two endpoints can configure the fabric. If the primary FM fails, the secondary one takes over. The first task of the FM consists in discovering the fabric topology. This information is necessary to obtain a set of paths between endpoints. The fabric discovery process is also triggered every time that the FM detects the occurrence of a topological change in the network.

To perform its functions, the FM accesses the configuration space in each fabric device (endpoint or switch). It is a storage area that contains a set of fields to specify device characteristics as well as fields used to control the device. This information is presented in the form of structures called *capabilities*. Each capability structure defines a specific characteristic of the device. In particular, the *baseline* capability includes device control and status information. The first six 32-bit blocks in this capability contain general information for the device, such as its type and serial number, the number of ports supported, and the maximum packet size. Next, we can find up to 256 32-bit blocks that point to the information about each particular port in the device. This information includes link speed and width, and current port state.

A "node configuration and control" protocol, PI-4, defines the exchange of information between the FM and the devices. The PI-4 *read request* packets allow the FM to obtain information from any capability into a device. A PI-4 *read completion with data* packet is returned by the device, containing the requested information (up to eight 32-bit blocks). The path –in the opposite direction– and the traffic class used by the response are the same as the ones used by the request. If the read operation was not successful, a PI-4 *read completion with error* packet is returned.

Another management protocol considered in the ASI specification is PI-5. It is an event-reporting mechanism which may be used to detect topological changes. In particular, when a fabric device detects a change in the state of a local port, it can notify this event to the FM, by means of a PI-5 packet. After receiving this packet, the FM starts the change assimilation process.

3. Implementing the Discovery Process

In this work, we have assumed that the discovery process is centralized in the primary FM. In [10], alternative organizations are discussed. We also suppose that the FM obtains the complete fabric topology, discarding all the previously collected information.

In this section three possible ways to implement the discovery process are described. In all the cases, the FM begins the process discovering the endpoint which hosts it. After that, it uses a sequence of PI-4 *read request* packets to determine the nature (switch or endpoint) of each discovered device, and to obtain information about the activity of each port in those devices. The paths that these packets need to reach fabric devices are computed as the topology information grows.

3.1. Serial Discovery

A simple approach proposed by the ASI-SIG to implement the discovery process consists in performing a serialized discovery [11]. In this case, once the algorithm starts discovering a device in the fabric, it reads all the necessary information from its device configuration space, using a sequential and synchronized way, before it proceeds to discover additional devices. In other words, in this algorithm there is only a request packet in the fabric in every moment in time. In this paper, this algorithm will be called *Serial Packet*.

This implementation follows a breadth-first strategy to explore fabric devices. Figure 2 shows the flow chart describing the algorithm. An active port indicates that there is a live device attached to the other end of the port. The FM extracts the following device to explore from an exploration queue. Once it receives the device general information, it checks if the device has already been discovered through a different path. In that case, the FM updates its topology database and proceeds to discover the next device in the queue. In other case, the FM obtains additional attributes for each port and updates its topological information. The FM inserts a new element in the queue for each active port discovered. The discovery process concludes when the exploration queue is empty.

3.2. Improving the Serialized Algorithm

Our first proposal consists in improving the *Serial Packet* algorithm. In particular, we propose to add an internal parallel behavior to the algorithm when it obtains additional information about a specific device. Devices are discovered serially, but internal ports are checked in parallel. In this work, this algorithm will be called *Serial*

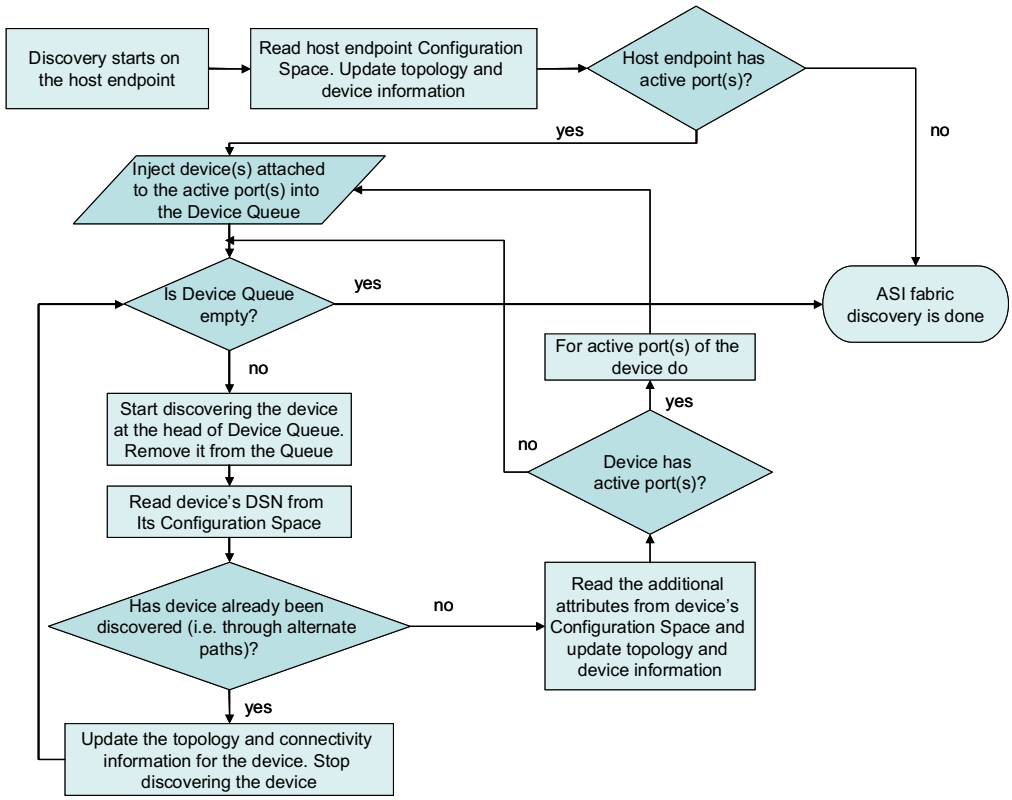


Figure 2. Serial discovery algorithm proposed in [11].

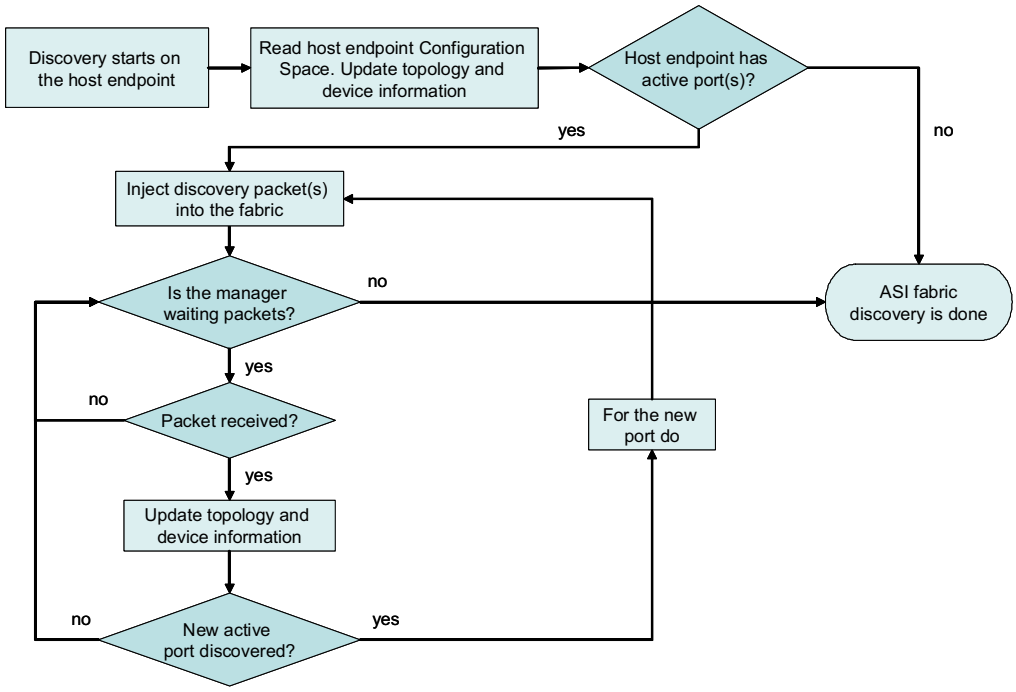


Figure 3. The proposed parallel discovery algorithm.

Device.

The flow chart in Figure 2 is also valid for the *Serial Device* algorithm. The difference is that the information about the ports in a device is obtained in a parallel way, by sending concurrently all the necessary PI-4 *read request* packets.

3.3. Parallel Discovery

In a completely parallel solution, multiple devices are discovered simultaneously. In our implementation, the FM performs the well-known propagation-order exploration algorithm [9] over the fabric. This means that discovery packets (PI-4) spread throughout the fabric in an "uncontrolled" way. The FM sends new PI-4 packets as soon as it receives responses to previous requests from devices. In this way, the order in which devices are discovered is not deterministic. In this paper, this algorithm will be called *Parallel Device*.

Figure 3 shows the behavior of the parallel discovery algorithm. In this case, the exploration queue has been replaced by a table of pending packets. Every time the FM receives a response packet, it updates its topology database. When the response packet includes general information about a device, the FM must inject new packets to obtain information about the ports in the discovered device. If a new active port has just been discovered, the FM sends a request packet, in order to discover the device at the other end of the link. The fabric topology has been completely discovered when the table of pending packets is empty.

4. Performance Evaluation

In this section, we present the simulation results that allow us to comparatively analyze the discovery alternatives described above. All the results presented in this work have been obtained using simulation techniques. Before showing and analyzing them, we describe the simulation methodology.

4.1. Simulation Methodology

Our simulation model [8] has been developed using the OPNET Modeler software [6]. The model embodies

Topology	Switches	Endpoints	Total
3×3 mesh, 3×3 torus	9	8	17
4×4 mesh, 4×4 torus	16	12	28
6×6 mesh, 6×6 torus	36	20	56
8×8 mesh, 8×8 torus	64	28	92
9×9 torus	81	32	113
4-port 2-tree	6	8	14
4-port 3-tree	20	16	36
4-port 4-tree	56	32	88
8-port 2-tree	12	32	44

Table 1. Topologies evaluated.

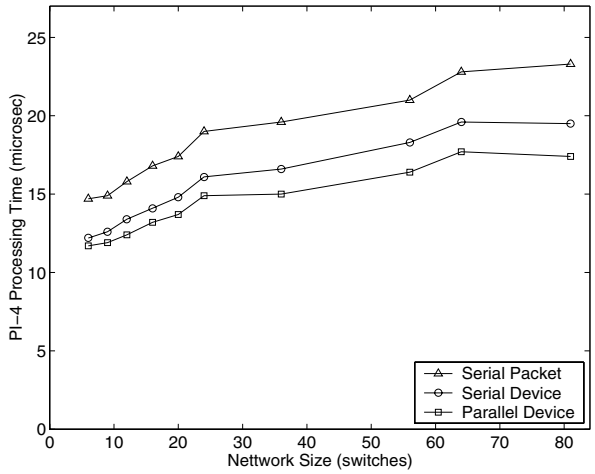


Figure 4. Average time to process a PI-4 packet at the FM for each discovery algorithm, as a function of the network size.

physical and link layers of ASI, allowing the simulation of several network designs. It is made up of ASI x1 links, 16-port multiplexed virtual cut-through switches [3], and 1-port fabric endpoints.

Additionally, the model provides the necessary support –management entities, device capabilities, and PI-4 and PI-5 packets– to develop fabric management mechanisms. It also allows accurate measuring of control overhead and the time spent by each task in the management process.

In order to obtain more realistic results, the model considers the time consumed by the FM and the device to process each PI-4 packet. In particular, we have measured this time by using profiling techniques, assuming a software implementation for the management entities, and using an Intel Pentium 4 (3.00 GHz) microprocessor.

We have checked that the packet processing time at the FM is slightly smaller for the *Parallel Device* discovery implementation (see Figure 4). The reason is that the implementation of the serial algorithms is more complex –they have to handle an exploration queue. The *Serial Device* algorithm is also faster than the *Serial Packet* one. The reason is that the former has to maintain less temporary information than the latter.

Additionally, the packet processing time at the fabric devices is low, and it does not depend on the discovery algorithm applied or the network size. The reason is that this processing always consists in returning a response packet including the requested information.

We have evaluated several regular topologies, including 2-D meshes and tori, and fixed-arity fat-trees built by using the methodology proposed in [5]. In meshes and tori, each external switch has an endpoint attached. Table 1 includes the complete list, and Figure 5 shows one of them.

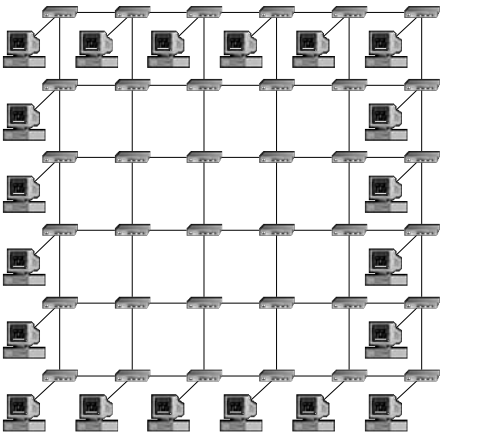
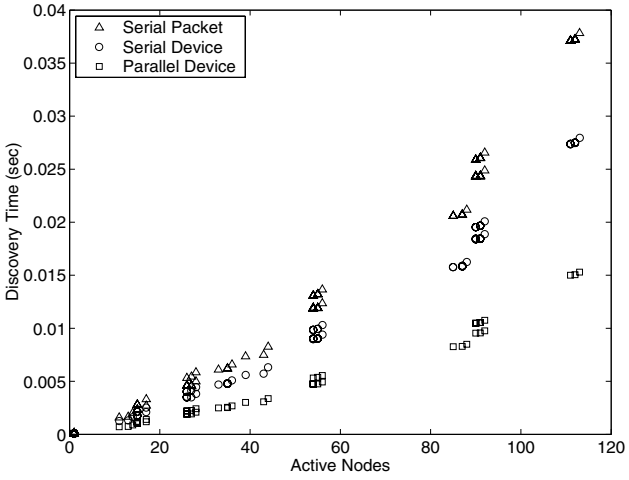
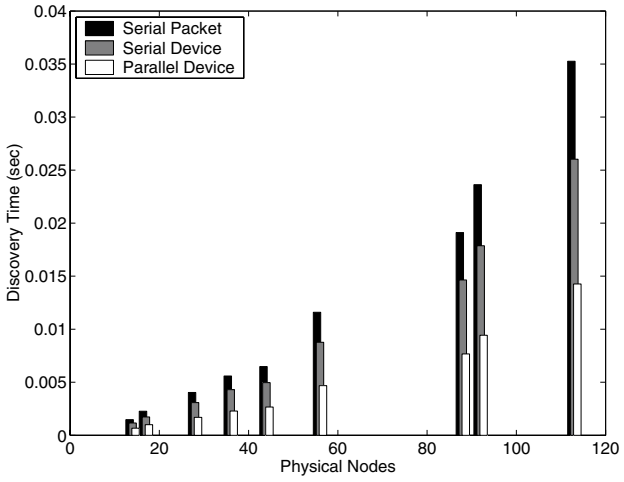


Figure 5. Example of an ASI fabric topology in OPNET (6×6 mesh).



(a) Versus the amount of active nodes



(b) Versus the network size (average results)

Figure 6. Time required by each algorithm to obtain the fabric topology.

The results presented here have been obtained without considering application traffic into the network. We have checked that this traffic scarcely influences the discovery time. The reason is that, in ASI, the management and notification packets have the highest priority when they are transmitted through the fabric.

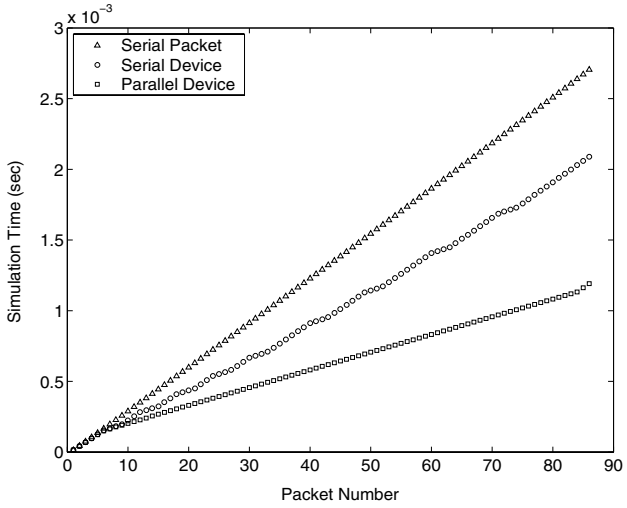
Each simulation begins with a transient period in which fabric devices are activated and the FM gathers the initial topology. After that, we have programmed the occurrence of a topological change, consisting in the addition or removal of a randomly chosen fabric switch. We have chosen a subset of possible causes for change, without lack of generality. For the detection of changes, we have implemented the event-reporting mechanism (PI-5) proposed in the ASI specification. This experiment has been repeated several times for each topology.

4.2. Simulation Results

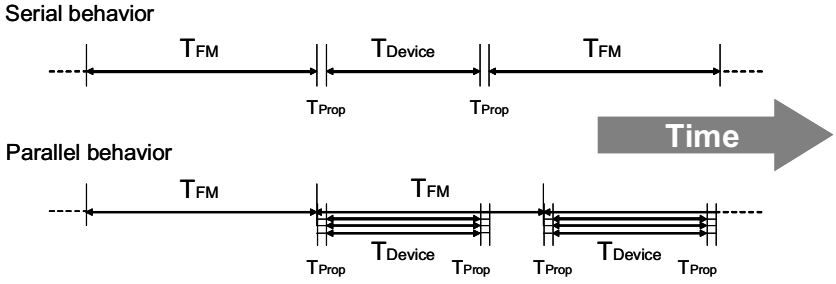
Figure 6a shows the discovery time for each simulation run. Horizontal axis represents the number of active and reachable devices in the fabric after the topological change. Results show that the discovery time is always smaller for the *Parallel Device* algorithm. Note that this improvement is scalable. The *Serial Device* algorithm is also a bit better than the *Serial Packet* one. Another important observation is that this behavior does not depend on the type of topology. Figure 6b shows the same results using average values for each topology in Table 1.

In order to analyze these results, Figure 7a details the time in which each discovery packet is processed at the FM, for the 3×3 mesh topology in Table 1, and assuming that all fabric devices are active.

First, we can observe that the slope of the *Serial Packet* series is constant. The reason is that this algorithm always has a serialized behavior. That means that the FM is idle while it is waiting for a packet response. On the other hand, the slope in the *Serial Device* series varies depending on the operation being performed by the FM. When it is obtaining general information about a new device, the algorithm has a serialized behavior. However, when the FM is obtaining information about the device ports, the serial process has a parallel behavior. That means that there is always a new packet pending to be



(a) Time in which each discovery packet is processed (3×3 mesh)



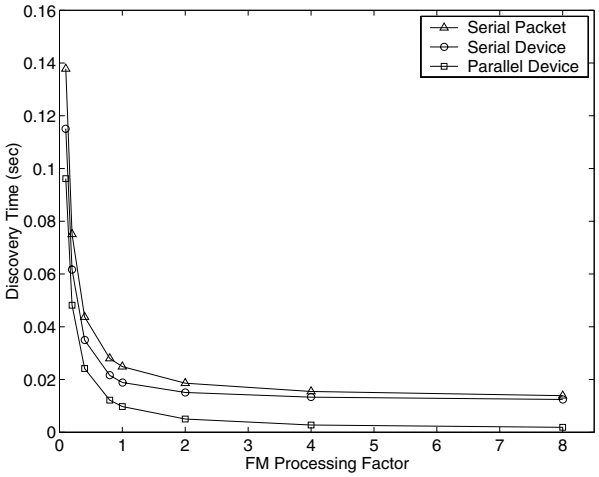
(b) Serial and parallel behaviors

Figure 7. Processing packets at the FM.

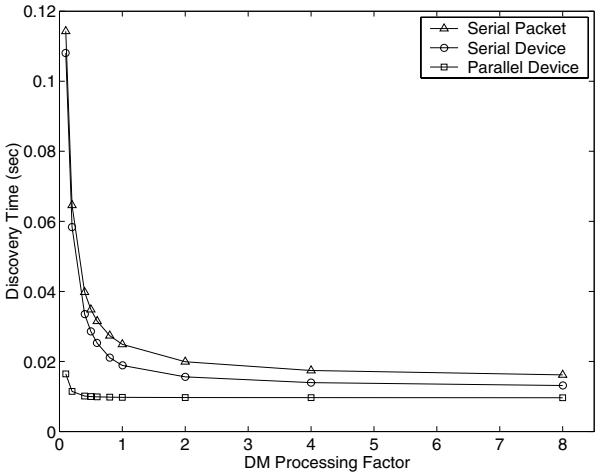
processed when the FM finishes processing the current one. The time to transmit a request packet, to process it at the destination device, and to transmit the corresponding response to the FM is overlapped with the processing of a previous packet. Finally, the slope in the *Parallel Device* series is again constant, because this algorithm has a completely parallel behavior.

Figure 7b represents the serial and parallel ideal behaviors graphically. In the figure, T_{FM} and T_{Device} refer to the time to process a packet in the FM and a fabric device, respectively, and T_{Prop} refers to the time to transmit a request/response packet trough the fabric.

4.3. Modifying the Performance of the Management Entities



(a) Varying the FM factor (device factor=1)

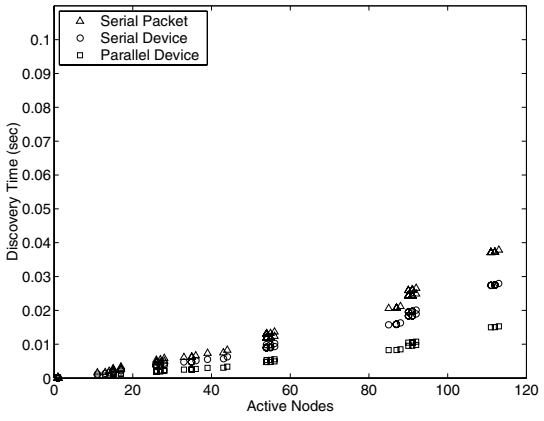


(b) Varying the device factor (FM factor=1)

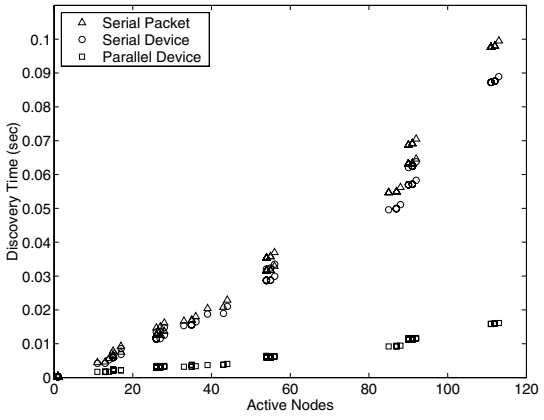
Figure 8. Discovery time for different processing factors (8×8 mesh).

Next, we analyze the effect of varying the performance of the management entities on the time required by the discovery algorithms. To do that, we have conducted new simulations by using a factor to increase or decrease the performance of the FM and the fabric devices. A fac-

(a)



(b)



(c)

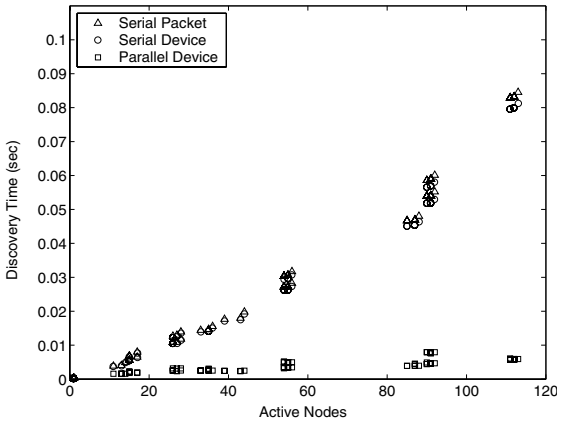


Figure 9. Discovery time for three particular combinations of the processing factors: (a) FM factor=1, Device factor=1; (b) FM factor=1, Device factor=0.2; (c) FM factor=4, Device factor=0.2.

tor of one represents the performance of an Intel Pentium IV (3.00 GHz) microprocessor. Previous results have been obtained by using this value. A processing factor of two indicates that packet processing is two times faster.

Figure 8a shows the discovery time obtained as a function of the FM processing factor applied, for the 8×8 mesh topology in Table 1, and assuming that all fabric devices are active. Results for different topologies are similar. We can observe that as the processing factor grows up, the discovery time decreases, and the difference between the serial and parallel implementations increases. Moreover, the difference between the *Serial Packet* and *Serial Device* algorithms slightly decreases.

As we can notice in Figure 8b, increasing the device processing speed only improves the serial discovery algorithms. The *Parallel Device* algorithm is not affected by the time consumed by the devices, because this process is overlapped with the processing of packets at the FM. Only when devices are too slow (factors < 1/3) the discovery time is affected.

According to these results, we have repeated the initial comparative study. Figure 9a shows the same results than Figure 6a, but adapting the scale in the vertical axis. On the other hand, in Figure 9b and Figure 9c we have fixed the device processing factor to 0.2. The difference between both plots is that Figure 9c shows the results using a FM processing factor equal to 4.

We can conclude that for faster FM and slower fabric devices, the difference between the *Parallel Device* discovery algorithm and the serial ones increases, independently of the fabric size.

5. Conclusions and Future Work

In this paper, several mechanisms to discover the topology of an Advanced Switching fabric are compared. Two of them have a serial behavior, discovering only one device at a time. The other one propagates the exploration through several paths in parallel. We have seen that the *Parallel Device* algorithm obtains the initially expected improvement compared with the serial ones. Additionally, differences between both implementations are more noticeable as the performance of the fabric manager increases and fabric devices are slower.

As future work, we plan to explore other approaches to perform the fabric discovery. One of them is to distribute the entire process through several collaborative fabric managers, in order to increase parallelization. A decentralized mode is more complex to design, because of the fact that several managers –in different locations and manipulating different data structures– must be coordinated.

Another possibility is to explore only the portion of the network affected by the change, instead of the entire fabric. However, as we have checked in previous works [2], the implementation of a discovery technique which reuses previous information is relatively complex.

We also plan to propose and analyze particular implementations for the rest of management tasks involved in the process of assimilating topological changes. In particular, we are interested in tackling the problem of dynamically distributing new paths to fabric endpoints after the occurrence of a change.

References

[1] Advanced Switching Interconnect Special Interest Group, Advanced Switching Core Architecture Specification (Revision 1.0), http://www.asi-sig.org, December 2003.

[2] A. Bermúdez, R. Casado, F. J. Quiles, T. M. Pinkston, and J. Duato, On the InfiniBand Subnet Discovery Process, *In Proc. IEEE International Conference on Cluster Computing*, Hong Kong (ROC), December 2003.

[3] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann Publishers, 2003.

[4] D. Mayhew and V. Krishnan, PCI Express and Advanced Switching: evolutionary path to building next generation interconnects, *In Proc. 11th Symposium on High Performance Interconnects (HOTI'03)*, 2003.

[5] X. Lin, Y. Chung, and T. Huang, A multiple LID routing scheme for fat-tree-based InfiniBand networks, *In Proc. International Parallel and Distributed Processing Symposium*, April 2004.

[6] OPNET Technologies, Inc., http://www.opnet.com/.

[7] PCI-SIG, PCI Express Base Specification (Revision 1.0.a), http://www.pci-sig.org, April 2003.

[8] A. Robles-Gómez, E. M. García, A. Bermúdez, R. Casado, and F. J. Quiles, A Model for the Development of ASI Fabric Management Protocols, *In Proc. Euro-Par 2006 Conference*, September 2006.

[9] T. L. Rodeheffer and M. D. Schroeder, Automatic reconfiguration in Autonet, *In Proc. 13th ACM Symposium on Operating Systems Principles*, October 1991.

[10] M. Rooholamini, Advanced Switching: a new take on PCI Express, http://www.asi-sig.org/press/Articles/, October 2004.

[11] M. Rooholamini and R. Kaapor, Fabric discovery in ASI, http://www.asi-sig.org/press/Articles/, October 2005.