DESIGN AND IMPLEMENTATION OF A FIBRE CHANNEL SWITCH ON LINUX

by

Xiangdong Huang

B.S., Zhejiang University, China (1992)

THESIS

Submitted to the University of New Hampshire

in partial fulfillment of

the requirements for the degree of

Master of Science

in

Computer Science

September, 2000

This thesis has been examined and approved.

Thesis Director, Dr. Radim Bartos
Assistant Professor of Computer Science

Dr. Elizabeth Varki
Assistant Professor of Computer Science

Barry B. Reinhold
Director, InterOperability Laboratory

Date

DEDICATION

To Xiaoyi Zhang.

ACKNOWLEDGEMENTS

I would like to thank Dr. Radim Bartos, my thesis advisor, for all of his support and guidance throughout the course of this work. His valuable advice and encouragement make this work possible. My gratitude is also extended to Dr. Elizabeth Varki for her teachings, valuable discussions, and comments on this research. I am equally indebted to Barry Reinhold for his enthusiasm, his outstanding expertise, and his confidence in me. I also want to thank all the guys who worked with me at the InterOperability Lab's Fibre Channel consortium. I especially appreciate the help and suggestions from Chris Loveland, Ashish P. Palekar, and David Woolf. And finally, I would like to express my special thanks to my family, particularly to my wife, Xiaoyi, for her understanding and patience. Thanks are also due to my friends living in the Forest Park who make my stay at Durham a memorable one.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DESIGN AND IMPLEMENTATION OF A FIBRE CHANNEL SWITCH ON LINUX

by

Xiangdong Huang

University of New Hampshire, September, 2000

Abstract

Fibre Channel is a fast serial transport mechanism that moves data reliably over long distances among heterogeneous systems. It has become the infrastructure for Storage Area Networks (SAN). The Fibre Channel switch is the key device that provides concurrent traffic between servers and storage resources. Many of the benefits of Fibre Channel including the ability to simultaneously scale storage capacity and throughput performance can only be realized with a switched Fibre Channel network.

The lack of interoperability between switches manufactured by different vendors is delaying the widespread adoption of heterogeneous SANs. The goal for this thesis is to build a Fibre Channel switch environment in the UNH InterOperability Lab (IOL) which can aid interoperability tests between switches or between a switch and switch-attached devices. An emulated Fibre Channel switch has been designed and implemented on a PC running the Linux operating system with multiple Fibre Channel host bus adapters as switch ports. The switch is able to perform port initialization, frame forwarding, name server management, principal switch selection, domain ID distribution, and routing. This emulated switch has been developed into a fabric testing tool, which can be used to carry out interoperability and parametric tests between switches and devices attached to switches.

Chapter 1

Introduction

As computer systems have increased in performance and capacity, I/O bandwidth has failed to keep pace. That is why Fibre Channel has been introduced in the last few years.

When the IBM PC-AT was introduced in the early 1980's it had a hard disk capacity of 10 megabytes. In the the past two decades, the amount of disk storage on the PC has increased by a factor of over 2000 with 20 gigabyte disks being widely supplied. Along with the increasing storage capacity of a single disk, disk arrays consisting of large numbers of commodity disks has also been used. This approach consists of either "Just a Bunch of Disks" (JBOD) which has identifiers for each disk or "Redundant Array of Independent Disks" (RAID) which has one identifiers for all disks. Attempting to create large disk arrays in this fashion has stressed the bandwidth and connectivity capabilities of existing parallel I/O interfaces well beyond their limits [9].

The server-to-disk data model has not been changed since the first computer was invented. This model links servers to storage arrays with fixed, dedicated connections. Data on a particular disk can normally be accessed via the SCSI bus only by a single server. The fact that this server-centric model does not meet the high-availability, high-flexibility and high-volume requirements of the growing enterprise networks is the primary impetus for the emergence of the Storage Area Network (SAN) [8].

In the SAN architecture, the storage devices are not attached to the host systems. Heterogeneous storage resources and servers are connected together to construct a subnet for storage. Figure 1.1 shows the architecture of a SAN.

SANs are currently implemented using Fibre Channel. Fiber Channel switch is the key device in building the SAN. It allows concurrent traffic between each server and the shared storage resources. The high bandwidth it provides determines the performance of the whole storage network.

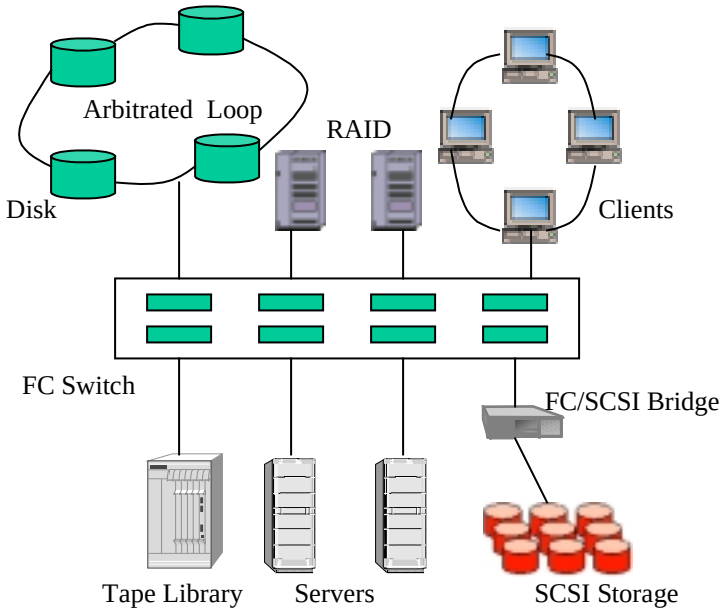

Figure 1-1: An Example of Switched Fibre Channel Storage Area Network.

The primary goal of this thesis is to design, implement and test a Fibre Channel switch using multiple Fibre Channel host bus adapters (HBA). The HBA card that will be used is the Interphase 5526 Fibre Channel PCI card with “Tachyon” Fibre Channel protocol chip manufactured by Hewlett-Packard. The UNH InterOperability Lab (IOL) has developed a Fibre Channel Linux driver for the card [12]. The basic I/O part of this driver will be used for developing the switch.

Chapter 2

Fibre Channel

2.1 Introduction

Fibre Channel is a fast serial transport mechanism that moves data reliably over long distances among heterogeneous systems. It began in 1988 as an outgrowth of a study group investigating strategies to enhance the Intelligent Peripheral Interface (IPI) physical interface. They were confronting a number of problems including limited data rate, limited distance, and limited connectivity. They further found that the problems were not unique to IPI, but rather, were present in all other interfaces based on parallel buses such as Small Computer System Interface (SCSI). The decision was made to define a new physical interface that was independent of any specific command set behavior and flexible enough to support different command sets and their associated architectures. Local Area Network (LAN) technology was in the developing stage at that time, so the ideas associated with LANs were also borrowed to develop the new interface. Historically, Fibre Channel has the benefits of high speed, long distance, and high scalability and meets the needs of both storage channel users and network users.

Fibre channel supports multiple protocols, including SCSI, IP, HIPPI, and IPI. The five layers of the Fibre Channel Standard define a physical layer, encode/decode scheme, frame protocol, general services, and upper level protocol (ULP) mapping [10]. There is a Fibre Channel Arbitrated Loop (FC-AL) protocol between the FC-1 and FC-2 layers. The layer hierarchy is shown in Figure 2.1. Fibre Channel uses the same physical media and 8b/10b encode/decode as Gigabit Ethernet. Fibre Channel allows distance of 500 meters with multi-mode fiber and up to 10 kilometers with single-mode fiber. Storage users are no longer restricted to the 12-foot cable lengths of single-ended SCSI.

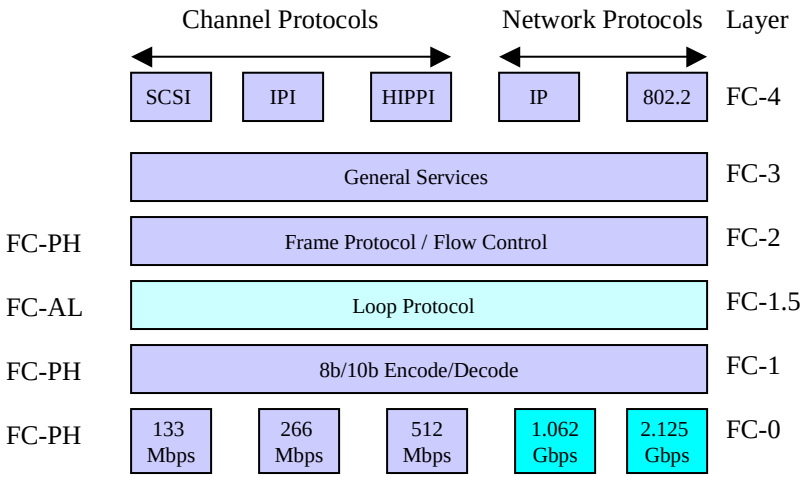

Figure 2-1: Fibre Channel Layers.

2.2 Fibre Channel Ports and Topologies

Equipment connected to Fibre Channel can have one or more ports. Fibre Channel specifies different ports with different purposes. Five different logical ports are defined for three topologies which is shown in Figure 2.2. The five kinds of ports are N_Port, NL_Port, F_Port, FL_Port, and E_Port [11].

- **N_Port(Node_Port)** is the simplest port which can only participate in a point-to-point connection. It only connects to another N_Port or to a F_Port.
- **NL_Port(NodeLoop_Port)** is the port which can participate in arbitrated loop topology.
- **F_Port(Fabric_Port)** is the switch port which connects a N_Port.
- **FL_Port(FabricLoop_Port)** is the switch port which participate in arbitrated loop.
- **E_Port(Expansion_Port)** is the switch port which connects to another switch to build a fabric topology.

All these ports are logically defined for different functions. They perform different port initialization processes before communication can begin. Actually one physical port has the capability of acting as multiple logical ports. Normally the N_Port and NL_Port are supported by a no-switch port. The F_Port, FL_Port and E_Port are supported by a switch port.

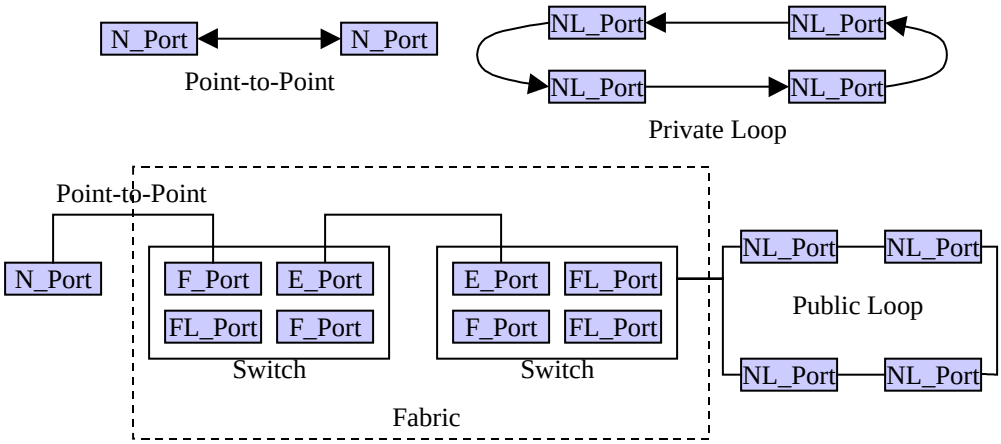

Figure 2-2: Fibre Channel Ports and Topologies.

2.2.1 Point to Point

Point-to-point is a direct channel connection between two ports. No arbitration and addressing are needed in this setup. The devices have access to the full bandwidth of the link.

2.2.2 Fabric

When one or more switches are used, the topology is called fabric. A fabric appears as a single entity to attached nodes, called F_Port if the attached node is a N_Port or FL_Port if the attached node is a NL_Port. The fabric topology is good for interconnecting large numbers of devices and complex configurations. Fabrics can provide name services, broadcast services, alias services, time services, and routing functions.

2.2.3 Arbitrated Loop

It was realized that there would be a need to connect more than one device to an F_Port. One of the reason is that the price per fabric port is too high to connect just one device to a F_Port. Although this point-to-point connection will access the full bandwidth, most nodes do not constantly need the full bandwidth. The Arbitrated Loop offers a loop topology for devices to share the bandwidth. The loop can contain as many as 126 NL_Ports and one fabric-port with all the nodes arbitrating for usage of the loop. If the loop consist of only NL_Ports, it is called a private loop. If there is a FL_Port on the loop, it is called a public loop. Every

device on the loop has a unique one byte long Arbitrated Loop Physical Address (AL_PA). When arbitrating, the device with lowest AL_PA will win the arbitration and it can open communication with any device on the loop.

2.3 Flow Control

The concept of flow control deals with the problem where a device receives frames faster than it can process them. The solution is that a device can transmit frames to another device only when the other device is ready to accept them. Before the devices send data to each other, they need to login with each other. During the process of login, they will report to each other the number of frames a device can receive at a time, which is called credit. After login has been performed successfully, each device knows how many frames the other device can receive. When credit runs out, no more frames can be transmitted until the destination device indicates it has processed one or more frames and is ready to receive new ones. Fibre Channel uses two types of flow control, end-to-end and buffer-to-buffer.

End-to-End: End-to-End flow control is used by a point-to-point port pair in Class 1 and Class 2 services which will be introduced in more detail below. It is performed with end-to-end credit count (EE_Credit_CNT) with EE_Credit as the controlling parameter. When the two ports log into each other, they report how many receive buffers are available for the other port. This value becomes EE_Credit. EE_Credit_CNT is set to 0 after login and increments by 1 for each frame transmitted to the other port. It is decreased upon reception of an Acknowledgement (ACK), Reject (RJT) or Busy (BSY) link control frame from that port. The ACK frames indicate that the port has received and processed 1 frame or N frames. A device will not transmit a data frame unless the allocated EE_Credit is greater than zero and the EE_Credit_CNT is less than the EE_Credit.

Buffer-to-Buffer: This type of flow control is widely used in Classes 1, 2, and 3. Each port on the link exchanges values of how many frames it is willing to receive at a time from the other port. This value becomes the other port's buffer-to-buffer credit (BB_Credit) value and remains constant as long as the ports are logged in. Each port also keeps track of BB_Credit_CNT, which is initialized to 0. For each frame

transmitted, BB_Credit_CNT is incremented by 1. The value is decremented by 1 for each R_RDY primitive signal received from the other port. Similar to end-to-end rules, a device will not transmit a data frame unless the allocated BB_Credit is great than 0 and the BB_Credit_CNT is less than the BB_Credit.

2.4 Classes of Service

Fibre Channel offers the possibility of using different classes of service for the data. Devices are not required to implement all of the classes, most implement just a sub-set. Class 3 is the most frequently used class of service.

Class 1 provides true connection service. The service is circuit switched, dedicated bandwidth connection. This class of service guarantees in order delivery. End-to-end flow control is used in Class 1. Some applications use this guaranteed bandwidth delivery feature to move data reliably and quickly without overhead of a network protocol stack. Class 1 is best suited for continuous and time critical traffic, such as voice or video.

Class 2 is a connectionless service, independently switching each frame and providing guaranteed delivery with an acknowledgement of receipt. As with traditional packet-switched (connectionless) systems, the path between two interconnected devices is not dedicated. The fabric multiplexes traffic from N_Ports or NL_Ports without dedicating a path through the fabric. A BSY or RJT is sent when no buffer space is available. The originating port will re-send the frame. This way no data is arbitrarily discarded just because the switch buffer is full.

Class 3 similar to Class 2, except that no confirmation of receipt is given. It only uses buffer-to-buffer flow control. Storage interface of arbitrated loop or multicasts and broadcasts on networks use Class 3. A logical point-to-point connection is established when using Class 3, so no confirmation is needed.

Class 4 provides fractional bandwidth allocation of the resources of a path through a fabric that connects two N_Ports. **Class 6** provides dedicated connections for reliable multicast for a fabric topology. These two classes are used very rarely.

Class F is a connectionless service between E_Ports, used for control, coordination, and configuration of the fabric between switches. It uses both buffer-to-buffer and end-to-end flow controls. R_RDY is used for buffer-to-buffer flow control. ACK_1 frame is used to perform end-to-end flow control. The Class F frame begins with a Start of Frame Fabric (SOFf) delimiter and terminates by an End of Frame Normal (EOFn) or End of Frame Terminate (EOFt) delimiter. If the switch is unable to deliver the frame to the destination E_Port, the source will be notified by a Fabric Busy (F_BSY) or Fabric Reject (F_RJT) with a valid reason code.

Chapter 3

Fibre Channel Switch

3.1 Introduction

Of the three kinds of Fibre Channel topologies, point-to-point does not show the benefits of Fibre Channel because only one other device can be on the bus. The arbitrated loop provides better connectivity than SCSI can, but it still has bandwidth restriction because bandwidth on a loop is shared by all the devices on the same loop. Many of Fibre Channel's benefits including the ability to simultaneously scale storage capacity and throughput performance can only be realized with a switched Fibre Channel network.

A Fibre Channel switch is composed of the following major components shown in Figure 3.1:

1) One or more **switch ports**: F_Port, FL_Port, or E_Port.
2) A **fabric controller**: The fabric controller is a logical entity that performs the management of the switch.
3) An **address manager**: The address manager is responsible for the assignment of addresses within some portion of the fabric. Within the switch, the address manager is responsible for acquiring a domain and area for the switch, and allocating Port_IDs within the domain and area.
4) A **path selector**: The path selector is a logical entity that establishes frame routing paths.
5) A **router**: The router is a logical entity that performs the routing of Class 2 and Class 3 frames to their final destination.

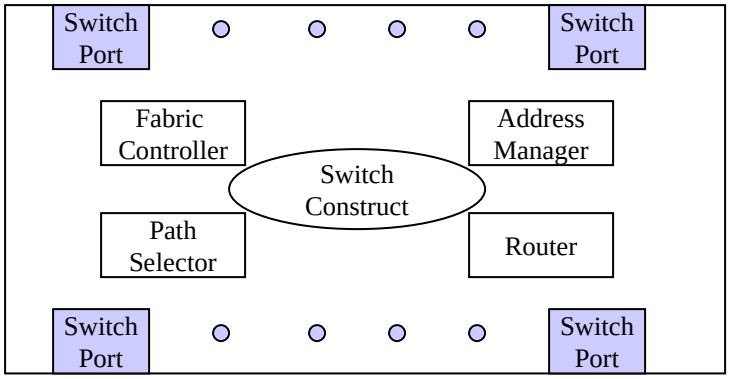

Figure 3-1: Fibre Channel Switch Model.

A Fibre channel switch supports dedicated connection, connectionless service, connection-oriented service or distributed fabric element topology. Multiple dedicated connections may exist simultaneously within the Fabric. The interconnection of F_Ports established by the fabric does not affect the existing interconnection of any other F_Ports, nor does it affect the ability of the fabric to remove those connections, nor does it affect the ability of any other FL_Ports to handle connectionless operations. When a dedicated connection is established, F_Ports and their respective point-to-point links are interconnected within the fabric, such that the links appear as one continuous link for the duration of the connection. As connectionless service is used, frames are multiplexed on frame boundaries between an FL_Port and any other FL_Port and thereby between the NL_Ports attached to them. Connection-oriented service is a virtual connection service that is commonly implemented upon a connectionless service. It may provide a fractional allocation of bandwidth between connected N/NL_Ports. Distributed fabric element topology allows users to combine switches to form a tree topology.

Multiple switches may be joined freely or in a structured fashion to form a larger fabric which is shown in Figure 3.2. A fabric is an entity which interconnects various Nx_Ports attached to it and is capable of routing frames using only the destination identifier (D_ID) information in the frame header.

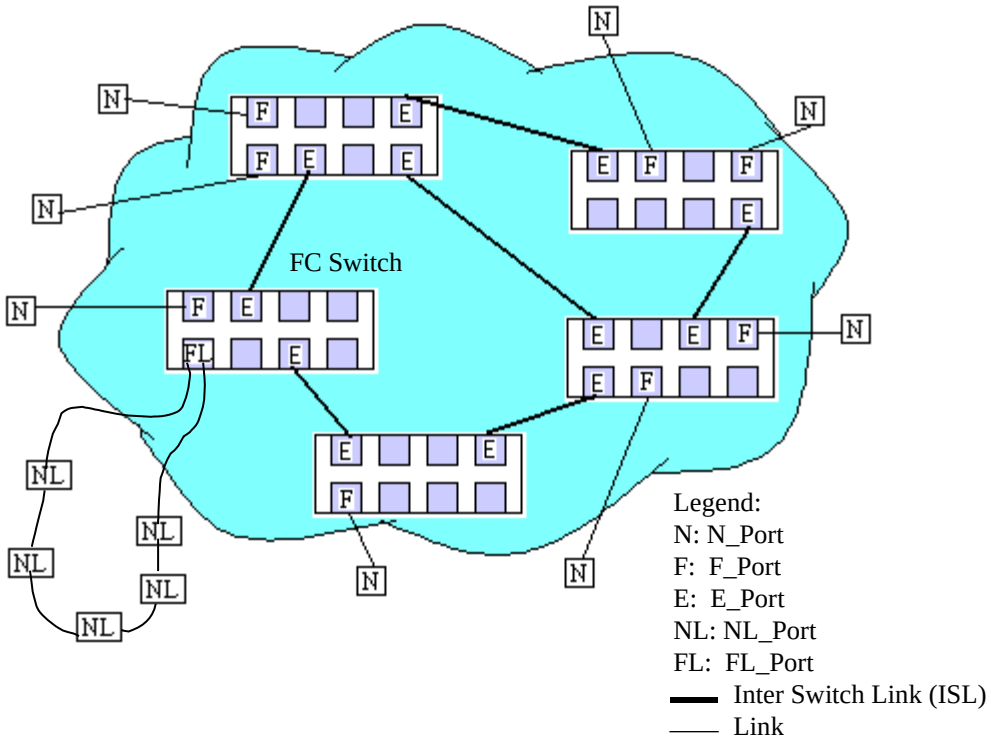

Figure 3-2: Multiple Switch Fabric.

3.2 Name Server

Similar to the 48-bit MAC address scheme in Ethernet, each Fibre Channel device has a 64-bit globally unique world-wide-name (WWN). However the WWN is not used for transporting frames across the network. A 24-bit port address (Port_ID) is dynamically assigned for every device. The addresses of two communicating partners are embedded in the frame header for both the destination identifier (D_ID) and source identifier (S_ID) [4]. By using this shorter address, the frame header and the routing logic are optimized for high-speed switching of frames. The 24-bit Port_ID format is shown in Figure 3.3. Each switch is a Fibre Channel domain. A Domain_ID will be assigned for every switch during the fabric configuration procedure. Every port of a switch has a unique Area_ID which is determined by the switch.

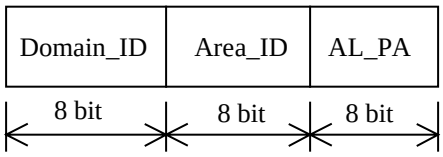

Figure 3-3: Fibre Channel Port_ID Format.

Besides Port_ID and WWN, the Node Name, IP address, symbolic port name, and symbolic node name are also used for addressing in different circumstances. Every port has some attributes related to it such as Fibre Channel layer 4 types (FC-4 type), class of service, port type, and initial process associator. A table is needed to keep track of the name mapping and the attribute relations. The name server is responsible for maintaining this table.

The name server keeps all the information regarding a port, such as Port_Name, Node_Name, Port_ID, class of service and so on [3]. All the port data is stored in the database. Every port login to the switch should register one or more of their name mapping and attributes with the name server. The switch and other devices may use the information kept in the name sever to do device discovery and frame routing. The name server can accept three kinds of request from other ports:

1) Get Objects: read specific information from the table
2) Register Objects: add specific information to a port entry in the table
3) De-register Objects: remove a port entry from the table

There are two possible responses for all these requests: Fibre Channel service accept (FS_ACC) and reject (FS_RJT). For different requests, the ACC payload could be different. Such as if one issues a request to get the Port_Name and another request to get the class of service, the payload in the two ACC would be the Port_Name and the Class of Service respectively. The FS_RJT payload consists of the reason code and explanation code.

3.3 Principal Switch Selection

In a multiple switch environment, one switch is responsible for assigning Domain_IDs to each switch in the fabric. This switch is called the principal switch and it is designated as the domain address manager. Inter-switch links (ISL) are used by switches to transmit frames to and receive frames from other switches. Multi-switch connection establishes a graph with switches as vertices and ISL as edges. To avoid infinite loops during fabric configuration, a spanning tree for the graph is established during the process of principal switch selection. Each switch has its Switch_Priority and Switch_Name. The principal switch is the switch with the lowest value of Switch_Priority and Switch_Name. Switches use Exchange Fabric Parameters (EFP) Switch Internal Link Service (SW_ILS) to compare these values. EFP's payload contains the Switch_Priority and Switch_Name that the transmitting switch believes is the principal switch and the Domain_ID_List containing the assigned Domain_ID and the corresponding Switch_Name. There are three different cases when EFPs are exchanged between different switches [1].

1) If a switch receives an EFP with an empty Domain_ID_List (no Domain_ID has been assigned) and the switch itself has an empty Domain_ID_List too, then the values of the Switch_Priority and Switch_Name are compared and the lower pair is kept as the new value. The switch will also note from which E_Port it received the new value, for the use as the upstream principal ISL.

2) If a switch receives an EFP with non-empty Domain_ID_List (Domain_ID has been assigned) and the switch itself has an empty Domain_ID_List, then the switch retains the received Switch_Priority and Switch_Name values. This ISL should be noted as the upstream principal ISL.

3) If a switch receives an EFP with empty Domain_ID_List and the switch itself has non-empty Domain_ID_List, the switch keeps its own values and transmits an Announce Address Identifier (AAI) frame to another switch.

This process selects one switch to be the principal switch. It is the root of the spanning tree for the fabric graph. The principal ISLs are the edges of the spanning tree. Figure 3.4 shows the result of the principal switch selection from six switches with the same priority and different names (A to F).

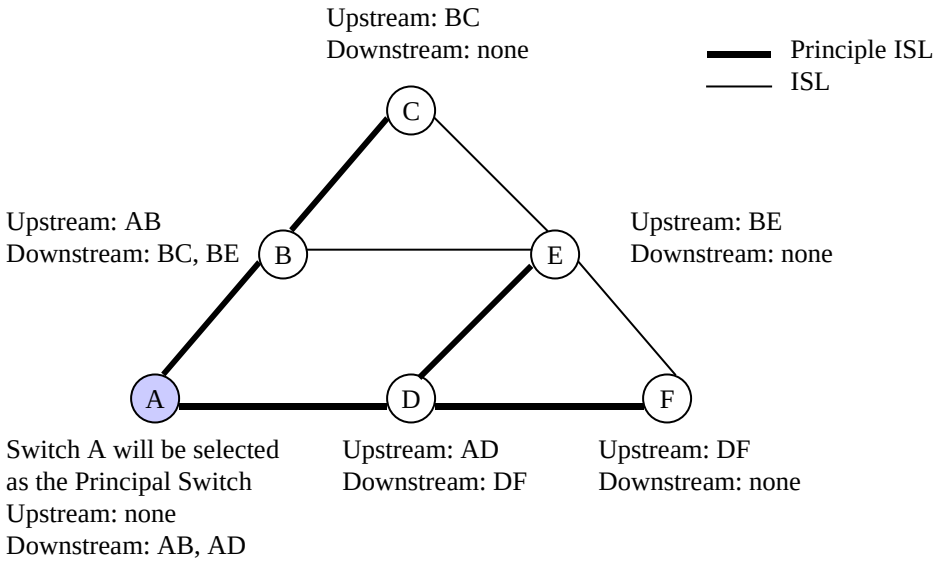

Figure 3-4: Principal Switch Selection.

3.4 Address distribution

Once a principal switch has been selected, this switch will be the domain address manager. It will grant itself a Domain_ID and empty its Domain_ID_List. Then the principal switch will send an AAI frame via all E_Ports. Other switches may request a Domain_ID from the principal switch after getting an AAI frame.

After receiving an AAI frame, a switch can send the Request Domain ID (RDI) frame to the upstream principal ISL. If the principal switch is on the other end of this ISL, the principal switch will pick an ID from the pool of available Domain_IDs and put it in the accept frame to the RDI request. The accepting switch will use this ID as its domain ID. This process is shown in Figure 3.5. If the switch that receives the RDI frame is not the principal switch, the switch will forward the RDI to its upstream ISL. When the reply SW_ACC is received, the SW_ACC will also be forwarded via the downstream principal

ISL on which the initial request was received. Figure 3.6 shows this process. After processing each RDI, the principal switch originates an EFP to all principal ISLs to announce the change to the Domain_ID_List.

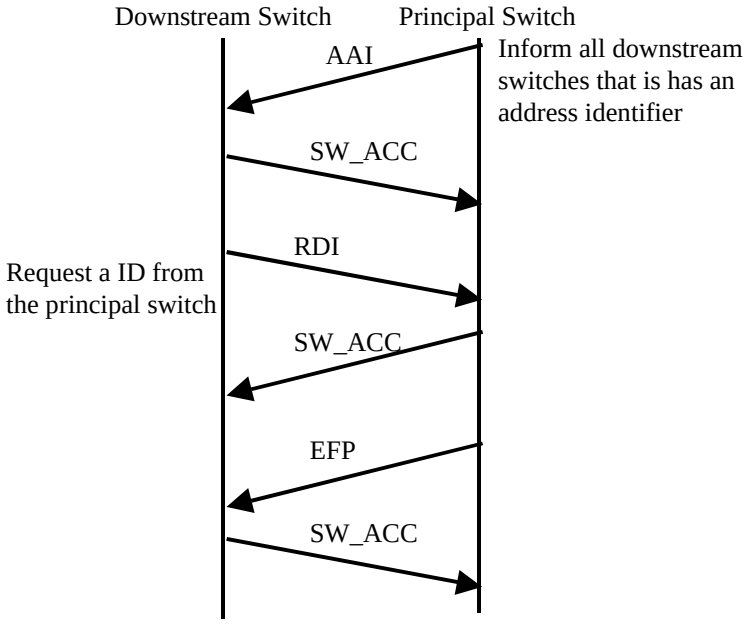

Figure 3-5: Request Domain ID (RDI) Processing by Principal Switch.

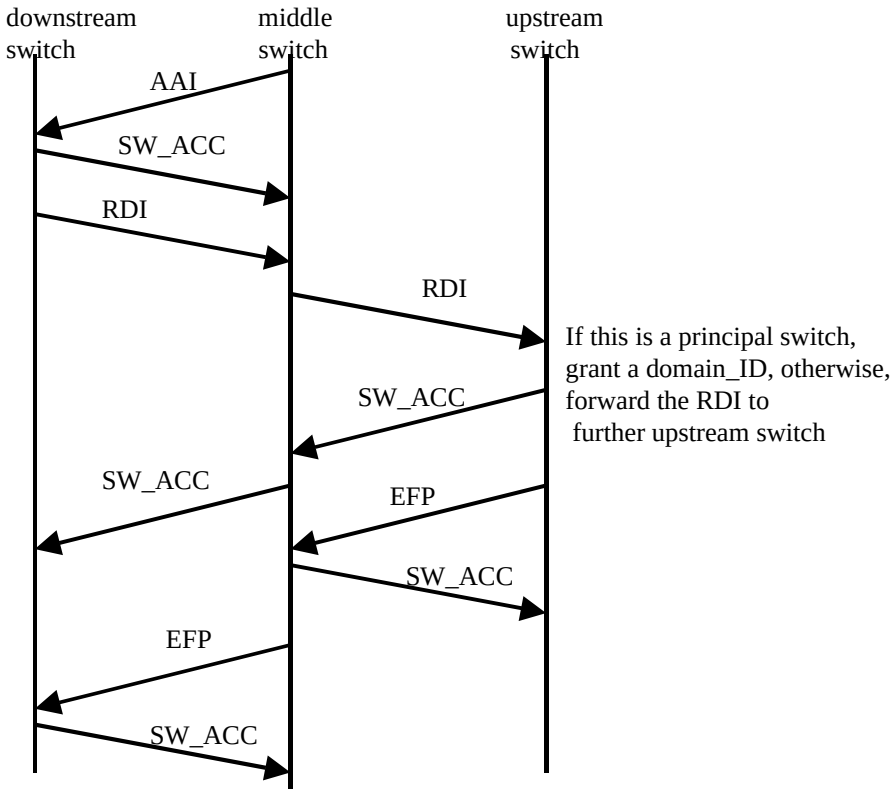

Figure 3-6: Request Domain ID (RDI) Processing by non-Principal Switch.

Chapter 4

Switch Design

4.1 Introduction

The switch developed in this thesis was implemented on the Linux operating system, version 2.2.14 on an Intel-based PC equipped with Interphase 5526 Host Bus Adapters (HBA). The HBA uses the "Tachyon" Fibre Channel chip manufactured by Hewlett-Packard. The Tachyon chip supports loop and link initialization. The login procedure, Registered State Change Notification (RSCN), and other layer-2 frame protocol are not supported.

The Tachyon chip can be initialized as N_Port, NL_Port, F_Port or FL_Port if the login process and Extended Linked Services (ELS) are supported. In order to have full F_Port or FL_Port function, FLOGI, RSCN, and name server registration process should be implemented after the link or loop initialization. The Tachyon chip supports Classes 1, 2, and 3 but not Class F [7]. So the E_Port, which works on Class F, is not supported by the Tachyon chip. However the Tachyon chip will change any invalid Start of Frame (SOF) into SOFf. We can take advantage of this feature to emulate Class F function. Under the non-AL setup, we could transmit the fabric service frames with invalid SOF delimiters. These frames will have the SOFf delimiter. Tachyon reports any frames with SOFf delimiter as unknown frames. The R-Rdy, which is needed in buffer-to-buffer flow control, is not transmitted by the Tachyon chip when an unknown frame is received. Instead of the R-Rdy, the ACK_1, which is needed in end-to-end flow control, is used for the acknowledgement for each valid frame received. The R-Rdy is one of the 4-byte Fibre Channel Ordered Sets. The ACK_1 is a six-word frame. In point-to-point mode, Tachyon can not be made to transmit an order set, but can be made to transmit any frame.

Each HBA is designed to initialize its ports as an F_Port or FL_Port, or E_Port according to what devices are attached to the ports. The switch has a name server that keeps the name mappings and attributes of each switch ports. The fabric controller of the switch provides fabric services, which can perform the principal switch selection and domain address distribution. Each FC HBA is also assigned a fixed Area_ID. When a data frame that does not have a well-known ID is received on a HBA, the driver first takes the Domain_ID from the frame header. If this Domain_ID is not the Domain_ID of the switch receiving this frame, the frame will be forwarded to the HBA through which the Domain_ID has been granted. Otherwise the destination is the local switch. Then this frame will be forwarded to HBA with the corresponding Area_ID.

4.2 Switch Architecture

The switch is implemented on a PC with a Pentium II processor, 128M memory, and four PCI slots. The HBA can be set to work in the F/FL_Port mode or the E_Port mode. As shown in Figure 4-1, two HBAs are designed as F/FL_Ports and the other two as E_Ports. The reason why the E_Port is designed separately is that the Tachyon chip cannot determine whether it is connected to an E_Port or not during the arbitrated loop initi alization.

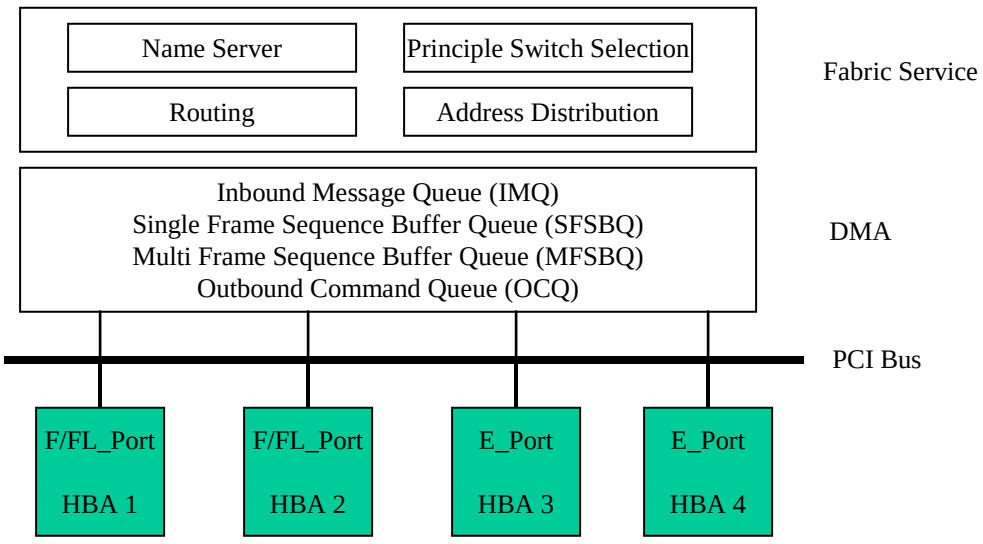

Figure 4-1: Switch Architecture.

Each HBA is allocated certain memory for DMA. Every time a frame comes in, it is saved in the Single Frame Sequence Buffer Queue (SFSBQ) or the Multi Frame Sequence Buffer Queue (MFSBQ) according to its frame size. Tachyon also supports SCSI, but this function is not used in this implementation because the HBA acts as a switching component, not as a host or a hard drive. Once a frame has been received, a hardware interrupt will be generated and the corresponding information about this interrupt is saved in the Inbound Message Queue (IMQ). After the host detects this interrupt, it checks the interrupt type from the IMQ. If the type indicates that a single or multi frame sequence has been received, the address where the data has been saved can also be fetched from the inbound message. Then the host will make a decision based on the contents of the frame received. When the host wants to transmit a frame, an Outbound Descriptor Block (ODB) will be constructed first. This ODB is then put into the outbound command queue (OCQ). At this point, the ODB has all the information needed to transmit a frame including the address of the frame header and payload. When Tachyon detects that a new entry has been added into the OCQ, it will get the frame header and payload from the ODB and then transmit this frame.

The routing process will provide the shortest path to the F/FL_Ports if the destination of the frame is not the current switch. The routing theory and implementation are introduced in Chapter 7. If the frame received from a F/FL_Port or E_Port is heading to another F/FL_Port on the current switch, this frame is forwarded to the destination HBA by the switch driver. The mechanism of the internal frame forwarding is shown in Figure 4-2. From the figure, it can be seen that the data is not copied from the source HBA to the destination HBA. Only the buffer address and the data length are given to the destination HBA. The destination HBA then sets the pointer of outbound data to the buffer address of the incoming data. The performance of the switch is improved by using this method.

Each HBA that acts as the F/FL_Port has an ordered linked list to store all the port identifiers registered to the fabric name server. The details of name server will be introduced in Chapter 6.

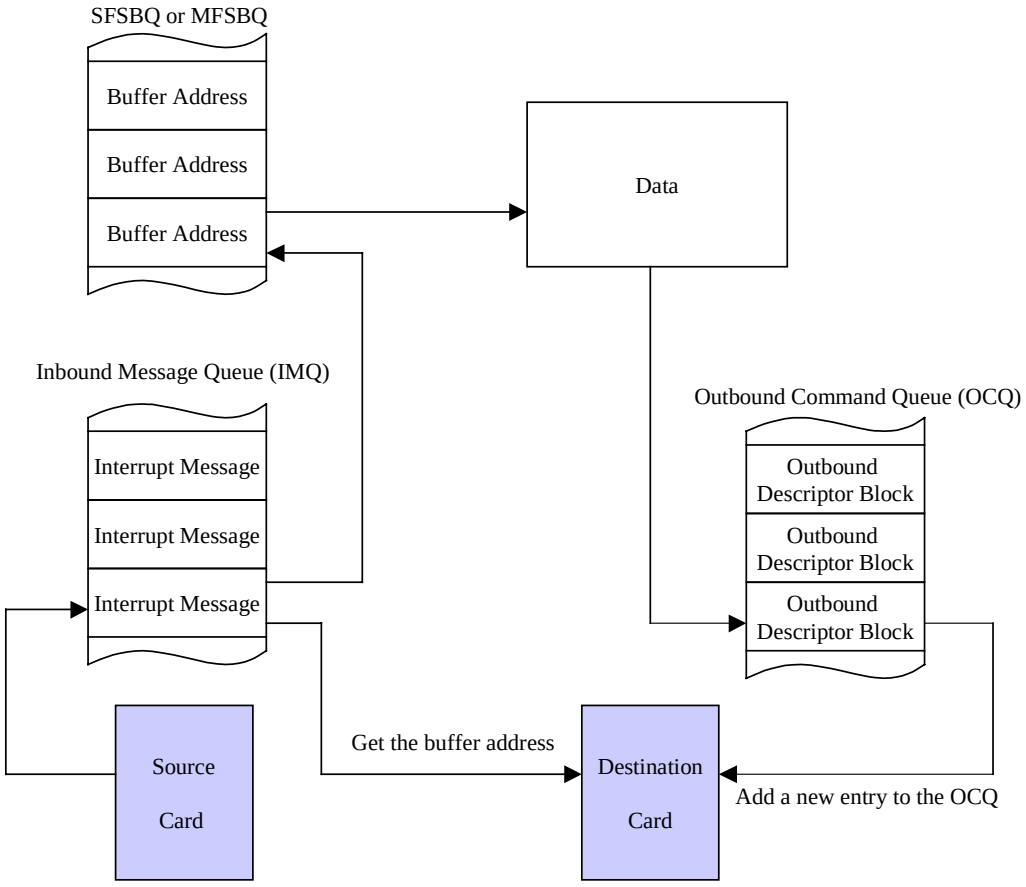

Figure 4-2: Internal Frame Forwarding.

4.3 Switch Port Initialization

A Fibre Channel switch port may be able to operate in F_Port, FL_Port or E_Port mode. One port can support more than one mode. In this design, the F/FL_Ports and the E_Ports are implemented on different HBAs because we have limited access to the low level information which is needed to determine which type of port the switch is connected to. During the F/FL_Port initialization, the switch port can assume a mode for the purpose of determining if that mode is appropriate. If it is successful, the switch port works as the assumed port; otherwise, it tries operating in another mode until it finds a mode in which to operate. The switch F/FL_Port first tries the FL_Port operation. If it fails, it then attempts F_Port. The E_Port HBA

is initialized directly as an E_Port which supports class F, principal switch selection, and domain ID distribution. The E_Port functionalities will be introduced in Chapter 5.

4.3.1 Login Procedure

The Login procedure is a method by which an N/NL_Port establishes its operating environments with a Fabric or another N/NL_Port. This procedure happens right after the arbitrated loop or point-to-point link initialization. The Fabric Login (FLOGI) is performed during the initialization process when a Fabric port is present. The FLOGI frame is directed to the Fabric using the well-known Fabric address 0xFFFFFE. The Port Login (PLOGI) is performed when a port wants to talk to another port other than the Fabric. For example, N/NL_Ports register to the Name Server via the PLOGI. The payloads of the Login frame and its accept frame (ACC) are same as shown in Figure 4-3.

The Login payload contains seven 4-word-long entries:

- 1 Common Service Parameter Entry
- 1 Port Name and Node Name Entry
- 3 Class Service Parameter Entries
- 1 Reserved Entry
- 1 Vendor Defined Entry

The common service parameter entry provides the service parameters common to all three classes: Class 1, Class 2, and Class 3. These parameters include the value of the buffer-to-buffer credit, the time out, and so on. The class service parameter entries provide the other service parameters that may be different to the three classes. These parameters contain the values of the class validity, received data field size, end-to-end credit, open sequences per exchange and, so on. The destination port, should it find the parameters acceptable, will transmit an accept (ACC) in response to the Login frame. The FLOGI ACC has another responsibility to assign a domain ID and an area ID to the N/NL_Port that issued the FLOGI to the fabric.

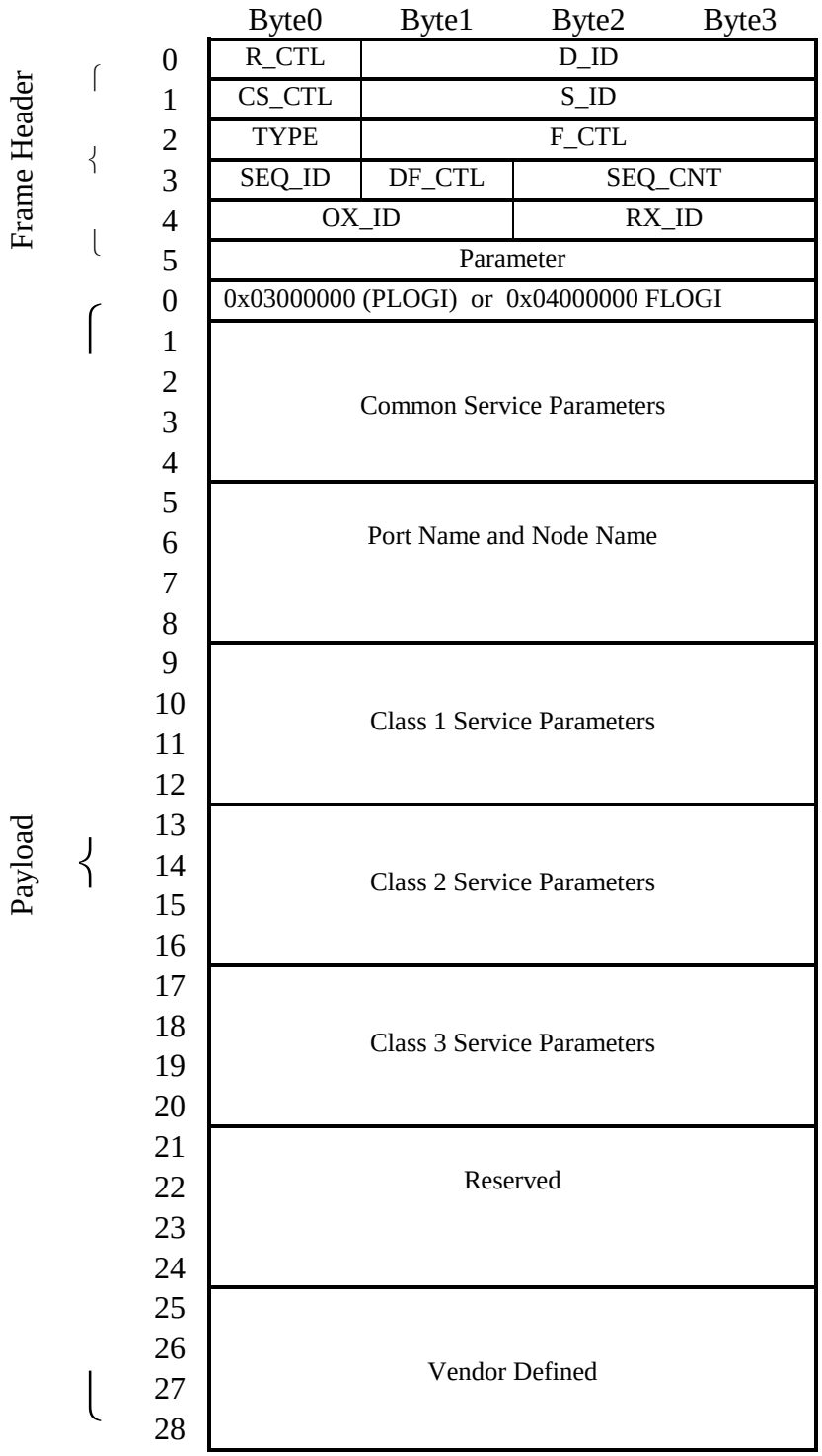

Figure 4-3: Login Frame Format.

4.3.2 FL_Port Initialization

The FL_Port is a port that operates on an arbitrated loop. The FL_Port, which has a fixed AL_PA 0x00, has the highest priority among all the ports on the loop. After the loop is initialized, every NL_Port on the loop gets an AL_PA, but the loop does not have a domain ID and an area ID. These two IDs are assigned by the switch via the FL_Port during the Fabric Login process. There is a login request bit (L_bit) in the Loop Initialization Soft Assigned (LISA) loop initialization sequence [5]. If the L_bit is set to one, all the attached NL_Ports will perform fabric login (FLOGI) to the switch with S_IDs in the format of 0x0000XX, which means they don't have domain IDs and area IDs. The switch will transmit acknowledgement frames (ACC) to these FLOGIs with D_ID in the format of 0xDDAAXX which means the NL_Port is assigned a domain ID 0xDD, an area ID 0xAA and the same AL_PA. If during the loop initialization process, the switch finds out that the NL_Port has already done FLOGI to the switch, the L_bit then will be cleared to zero. In this case, the FL_Port will initiate a Fabric Address Notification (FAN) extended link service (ELS) command to the attached NL_Ports before any other port is able to send a frame on the loop [6]. If the NL_Port determines that the FL_Port has the same address, FL_Port_Name and Fabric_Name, the port initialization process will complete the process. If the FL_Port address has been changed, the NL_Port should redo the FLOGI.

After fabric login, the NL_Ports are required to perform Name Server registration. If the NL_Port wants to communicate other NL_Ports in the same switch, it will query the name server to find the IDs of the NL_Ports. The figure for FL_Port initialization is shown in Figure 4-4.

4.3.3 F_Port Initialization

The difference between F_Port initialization and FL_Port initialization is that the F_Port initialization does not need to transmit the arbitrated loop initialization sequences (LIFA, LIPA, LIHA, LISA, LILP, LIRP). Whenever an order set such as LR, LRR, OLS or NOS is seen on the link, the F_Port begins its initialization. The N_Port attached to the F_Port suspends all open exchanges with all other N_Ports or NL_Ports. After the completion of the link initialization and if the N_Port was implicitly logged out from

the fabric, the N_Port performs FLOGI; otherwise, the N_Port performs FDISC to the fabric. If the F_Port address is not changed, all the suspended exchanges will be resumed and the port initialization is completed. If the F_Port address is changed, all the suspended exchanges will be discarded and re-do all the registration to the Name Server. Because the F_Port is connected directly to the N_Port, the AL_PA is not used on this point-to-point connection. Before the N_Port gets an assigned ID, it uses S_ID 0x000000. During the login procedure, the switch assigns the N_Port an ID in the format of 0xDDAA00 which means the domain ID is 0xDD and the area ID 0xAA.

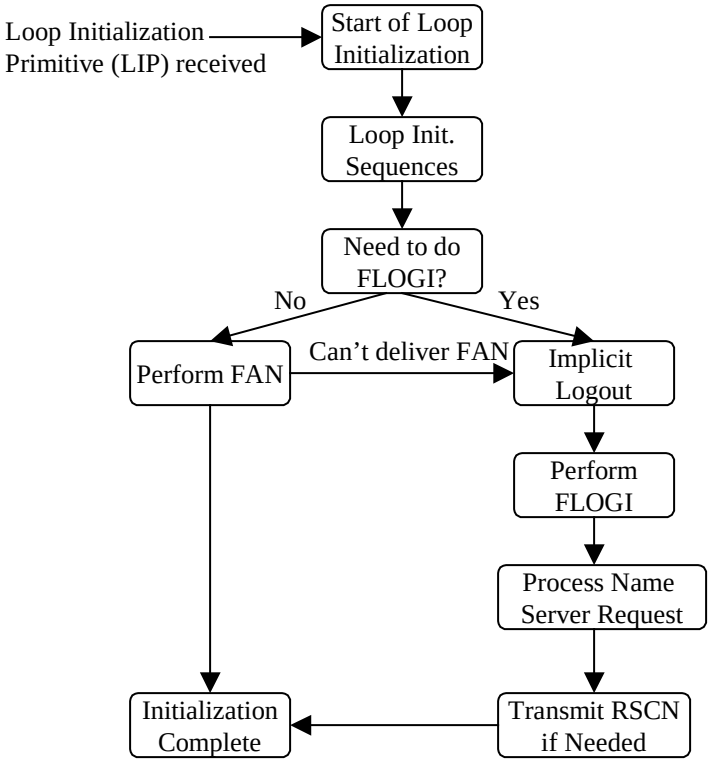

Figure 4-4: FL_Port Initialization.

4.3.4 Registered State Change Notification

When a procedure causes the link re-initialization of the F/FL_Port, such as when a device is added to or removed from an F/FL_Port, the RSCN frame will be broadcasted to all other N/NL_Ports that have registered for this service, except the N/NL_Port itself that had causeed the event. The N/NL_Ports can

register this service by transmitting a State Change Notification (SCN) frame to the switch. If an N/NL_Port receives a RSCN, it will re-start the query to the name server.

Chapter 5

Switch Fabric Services Implementation

5.1 Introduction

Fabric services are the link services that operate internal to the fabric between switches. All fabric service frames are transmitted via the Class F service. The fabric controller with the well-known address 0xFFFFFD handles these fabric services. The ELP, EFP, AAI, RDI, SW_RJT, and SW_ACC frames, as mentioned in Chapter 3, are the fabric services used most often.

- ELP Exchange Link Parameters
- EFP Exchange Fabric Parameters
- AAI Announce Address Identifier
- RDI Request Domain_ID
- BF Build Fabric
- RCF Reconfigure Fabric
- SW_RJT Switch Fabric Internal Line Service Reject
- SW_ACC Switch Fabric Internal Line Service Accept

ELP is used during the E_Port initialization procedure to exchange link parameters such as timeout values, port name, switch name, and service parameters of class 1, 2, 3, and F. EFP is used to exchange fabric parameters such as current principal switch priority and name and domain IDs granted by the fabric. When a switch with a domain ID is connected to a switch without a domain ID, an AAI is transmitted to the un-configured switch to announce that it has the ability to assign a new ID, or ask the principal switch to do the same. When a switch receives an AAI, it may transmit Request Domain_ID (RDI) to get a new

domain ID. Exchange Switch Capabilities (ESC), a newly added fabric service, defines a mechanism for two switches to exchange vendor and protocol information. BF and RCF are used to request fabric reconfiguration. BF requests a non-disruptive reconfiguration whereas RCF requests a disruptive reconfiguration. While doing a disruptive configuration, the addresses allocated to switches may be changed. A non-disruptive reconfiguration of the fabric does not affect the previous addresses. When a principal ISL is lost or two fabrics are joined, a BF request is appropriate. RCF is needed only when an overlapped domain was found or BF failed.

The E_Port is the switch port connected to another switch in a fabric. Frames with a destination other than the local switch or any NL_Port or N_Port attached to the local switch can exit the local switch through an E_Port. Fabric services frames are transmitted through E_Ports.

5.2 E_Port Initialization

An E_Port normally functions as a conduit between the switches for frames destined for remote N_Ports and NL_Ports. An E_Port is also used to carry frames between switches for purposes of configuring and maintaining the fabric. Switch fabric services operate internal to the Fabric between Switches using E_Ports. The principal switch selection and Domain_ID distribution will be implemented after E_Port is successfully initialized.

5.2.1 Class F Service

A switch port may support F_Port, FL_Port, and E_Port at the same time. It requires that the switch have the ability to detect whether there is a switch connected to itself during loop initialization. The switch chooses the arbitrated loop physical address (AL_PA) 0x00 in the loop initialization sequences such as LIFA, LIHA, LIPA and LISA. A switch is present if the AL_PA 0x00 flag is set. The Tachyon chip that is used in this thesis does not provide this information. In this thesis, the F/FL_Port and E_Port are implemented in different HBAs because the loop initialization procedure is done by hardware.

We have to solve two problems if the Tachyon chip can work in the E_Port mode. The first problem is that the Tachyon chip supports F/FL_Port but does not support E_Port and Class F, on which the fabric services depend. This thesis implements the E_Port and Class F on the Tachyon chip by taking advantage of Tachyon's default actions. The host programs the start of frame (SOF) value. Only the SOF for Class 1, 2, and 3 are acceptable; other values are regarded as invalid SOF and substituted by a SOFf, which is needed for Class F service. On the receiving side, a frame with SOFf will invoke the *inbound_unknown_frame_received* interrupt. Now we are facing another problem. The Tachyon provides no flow control for unknown-type frames, which means that no R-Rdy will be transmitted when any Class F frames are received. This problem is solved by transmitting an ACK_1 frame for any inbound Class F frame. In the point-to-point connection the Tachyon chip cannot be programmed to transmit a R-Rdy which is required by buffer-to-buffer flow control, but it can be programmed to transmit any frame such as the ACK_1 which is needed by end-to-end flow control. Fibre Channel switches from all five different vendors support both buffer-to-buffer and end-to-end flow control. The switch implemented in this thesis can talk to these switches by end-to-end flow control.

5.2.2 Exchange Link Parameters

Point-to-point connection links two E_Ports. The following primitive sequences are used for point-to-point communication:

- NOS: Not Operational
- OLS: Offline
- LR: Link Reset
- LRR: Link Reset Response

After the link is operational for frame transmitting, the E_Port will originate an Exchange Link Parameters (ELP) fabric service frame. The ELP payload contains the values of the timeout; port name and switch name; class parameters for Class 1, 2, 3, and F; and flow control parameters. The two E_Ports may

transmit ELP at the same time. The port with lower E_Port_name will transmit a SW_ACC. The other one will transmit a SW_RJT indicating that the ELP is already in progress, which is shown in Figure 5-1.

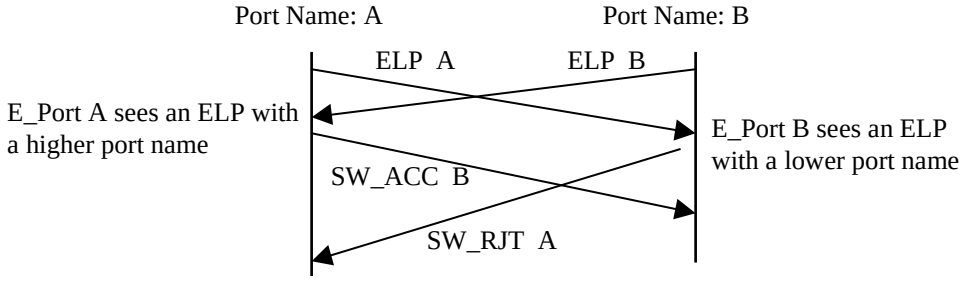

Figure 5-1: Simultaneous ELP Processing.

5.2.3 The Linux Timer

If the switch has not received an ELP_ACC or ELP for Error_Detect_Timeout (E_D_TOV) period, the switch will retry ELP. This timer and the EFP timer, which will be introduced later, are implemented by the Linux timer structure: *timer_list*.

```
struct timer_list {
      struct timer_list *next;         // next timer
      struct timer_list *prev;         // previous timer
      unsigned long expires;           // the timeout, in jiffies
      unsigned long data;              // argument to the handler
      void (*function)(unsigned long); // handler of the timeout
}
```

The entries *next* and *prev* in this structure are used for the internal management of all the timers in a doubly linked list. The timeout value is an absolute value of *jiffies*, which is the number of clock ticks since the operating system was booted. Linux timer interrupt is set to a default frequency of *HZ*, which is an architecture-dependent value. Linux defines *HZ* to be 100 for Intel-based platform, which means that the time interval between two clock ticks is 1/100 second, or 10 milliseconds, with one *jiffy* being equal to 10 milliseconds. For example, if a timer is set to E_D_TOV (2 seconds), the data member *expires* of the *timer_list* struct shall be set to 200. Some of the kernel functions used to act on timers are:

- *void init_timer(struct timer_list *timer)*
- *void add_timer(struct timer_list *timer)*

- *void del_timer(struct timer_list *timer)*

The *init_timer()* function initializes the timer structure. The *add_timer()* function inserts a timer into the global list of active timers. After the timer is added, the system begins to count the clock ticks. When the timeout expires, the handler is called. It might be worth it to keep in mind that the timer is managed by the operating system, not by the driver that added the timer. If the driver, which was inserted as a module, is removed before a timer expires, the timer is still counting the clock ticks and will call the timer handler, which does not exist at that time, when the timeout expires. In this case, a null pointer is called and the system is crashed. For example, if a Hello timer (20 seconds) is added and the driver module is removed, it is extremely difficult to find the reason why the system crashes some point after the driver was unloaded. If a timer needs to be removed from the list before it expires, the *del_timer()* function should be called. When a timer expires, it is removed from the timer list automatically.

If a delay that is less than 10 milliseconds, the time span of a *jiffy*, then the function

void udelay(unsigned long usecs);

should be used. The precision of the function *udelay()* is as short as one microsecond. It is important to know that this is a busy-waiting function.

5.3 Principal Switch Selection

Following the successful completion of ELP, the values of buffer-to-buffer and end-to-end Class F credit are initialized. The E_Port that had originated the successful ELP then resets the link in order to initialize the flow control parameters. After the link reset, the E_Port is ready to perform the principal switch selection and domain ID distribution procedures. Principal switch is the switch that is responsible for granting domain IDs to other switches. Each Fibre Channel switch represents a domain in the Fibre Channel network. The switches exchange fabric parameters between each other to determine who is the principal switch, which has the lowest switch priority and switch name value. A priority of 0xFF indicates the switch is not capable of acting as a principal switch.

5.3.1 Exchange Fabric Parameters

The exchange of fabric parameters is used to establish the address allocation within the fabric. The frame format of EFP fabric service is shown in Figure 5.2. Each EFP frame contains the values of the switch priority and switch name of the switch, which the transmitting switch believes is the principal switch. These two values will be updated if a lower pair is found. Every switch in the network keeps a list of all domain IDs granted to every switch. The EFP frame also contains this domain ID list. Each record of the domain ID list consists of a domain ID and the switch name for that domain ID. A switch that is not yet configured uses its own priority and name as the current principal switch priority and name values. The domain ID list is set to empty. In a non-empty domain ID list, the Principal Switch must have already been selected because only the Principal Switch has the capability to grant a new domain ID. The priority of the principal switch is 0x02.

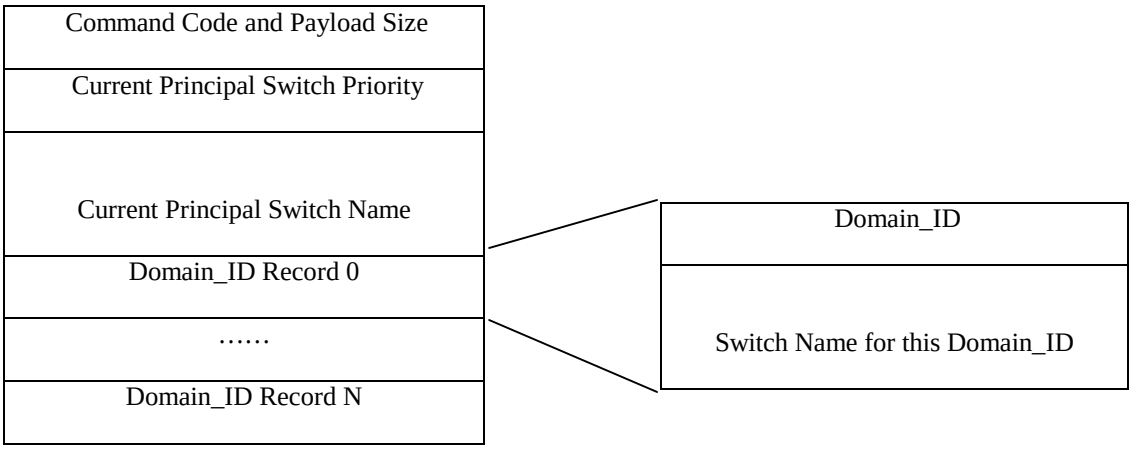

Figure 5-2: Exchange Fabric Parameters (EFP) Frame Format.

5.3.2 Principal Switch Selection State Machine

During the procedure of the principal switch selection, the switch moves to different states depending on what action has taken place. The principal switch selection state machine is show in Figure 5.3. There are six different states in the state machine:

- Non Disruptive: The switch needs a non-disruptive fabric reconfiguration.

- Disruptive: The switch needs a disruptive fabric reconfiguration.
- EFP Idle: Idle state before of after EFP processing.
- EFP Send: EFP processing state.
- Principal: The switch is the principal switch.
- Not Principal: The switch is not the principal switch.

The Fabric Stability Timeout Value (F_S_TOV), which is defined to be 5 seconds, is used to detect inactivity during fabric configuration. The principal switch selection state machine uses twice F_S_TOV to make transitions to certain states.

Following is the implementation of different state transitions:

1) Receiving a BF or RCF causes the transition to Non Disruptive or Disruptive state respectively. A non-disruptive re-configuration will not cause any domain ID changes. On the other hand, a disruptive re-configuration may cause a switch to choose a domain ID other than its former one. After 2 x F_S_TOV expires in both of these state, the switch transits to EFP Idle state.

2) An AAI indicates that a switch is capable of granting domain IDs. Thus any switch that has received an AAI will not become the Principal switch because it was already selected.

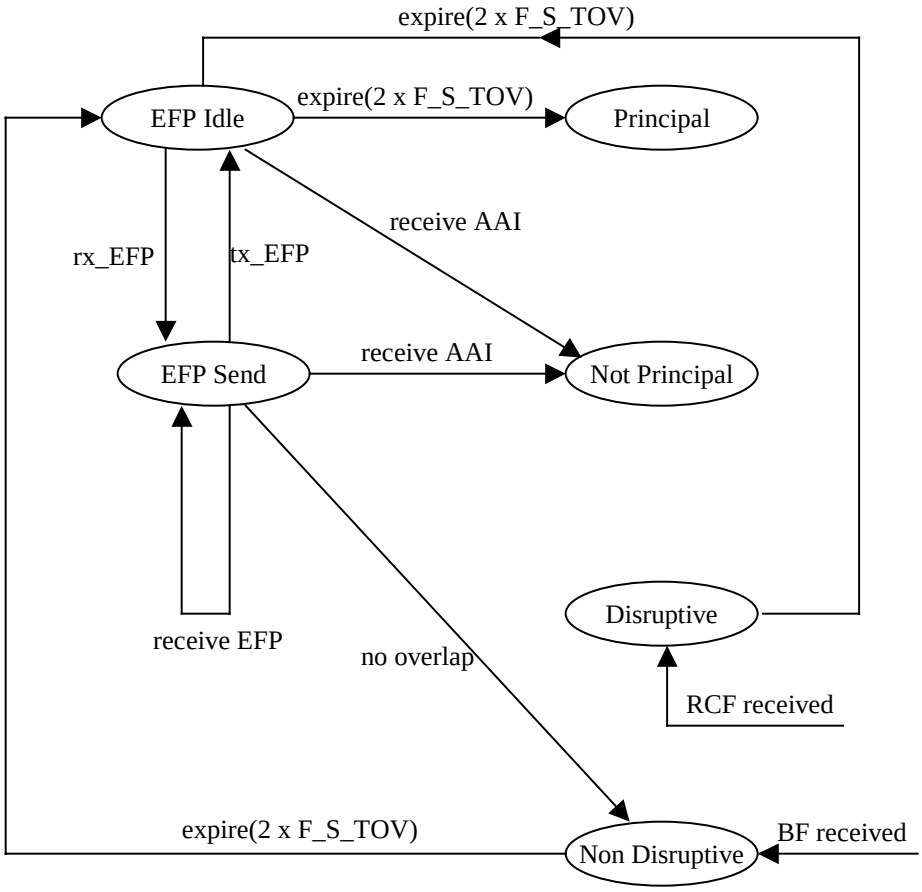

Figure 5-3: Principal Switch Selection State Machine.

3) When an EFP frame is received, the switch performs the following checks:

   a) If the switch with an empty domain ID list receives in an EFP with a non-empty domain ID list, the switch retains the newly received switch priority and name values. This switch will not become the Principal Switch.

   b) If the switch with an empty domain ID list receives an EFP that also has an empty domain ID list, the switch compares the received switch priority and name values to its own. If the received EFP has lower values, the switch sets its own values to the lower ones.

c) If the switch with a non-empty domain ID list receives in an EFP with an empty domain ID list, the switch retains its own parameters and transmits an AAI to the switch that is not yet configured.

d) If the switch with a non-empty domain ID list receives in an EFP also with a non-empty domain ID list, there are two possibilities: the EFP could be transmitted by the Principal Switch when a new domain ID is created or the switch in the current Fibre Channel network could be connected to another Fibre Channel network. In the first case, what the switch needs to do is just update its domain ID list to include the new domain ID. In the second case, a fabric reconfiguration fabric service will be performed in order to reorganize the domain IDs when two Fibre Channel networks are joined.

4) Only the switch that can retain its own switch priority and name for two F_S_TOV and whose priority is larger than 0xFF will be the principal switch. During this timeout, there is no switch that has lower switch priority and name values than these values of the principal switch. If the switch priority is 0xFF, it indicates that the lowest switch priority is 0xFF in this Fibre Channel network. Because the priority 0xFF also indicates that the switch is not capable of acting as the Principal Switch, then all the switches will be in the isolated state.

5.3.3 Keeping Track of Frames

In the process of handling the EFP, AAI and RDI, it is necessary for the switch to keep track of where and when the frame is transmitted. When a SW_ACC or SW_RJT is received, the switch needs to know to which frame the SW_ACC or SW_RJT is responding. Each fabric service frame will be assigned a unique Originator Exchange Identifier (OX_ID). Any response to the frame uses the same OX_ID. After the switch has transmitted a frame, a record is added to an OX_ID list. This record has the values of the type of the fabric service, the OX_ID of this frame, and the HBA order which generated this frame. When a response frame is received, the switch searches for the record in the OX_ID list that has the same OX_ID. This way the switch can know which response is to which frame. *Jiffies*, mentioned in paragraph 5.2.3, can

be used as the time stamp to record when the frame was received. Ten milliseconds time span is precise enough for debugging in this thesis.

The Tachyon chip can generate an interrupt for every transmitted frame. To improve the performance, this capability is disabled. This way, the number of interrupts is reduced by half. In most cases, frames can be successfully transmitted. Even when the outcome completion interrupt is disabled, any error during the frame transmission can still cause an interrupt. The OX_ID value of the frame is reported in the interrupt message. The switch handles this interrupt to find out why the error happened and whether a retry is needed.

5.4 Domain ID Distribution

At the completion of Principal Switch selection, the Principal Switch assumes the role of domain address manager. The Principal Switch first sets its own priority to 0x02. Then it grants itself a domain ID and transmits an EFP to broadcast its domain ID and switch name. After that, the Principal Switch transmits an AAI to announce that it has the capability of granting domain IDs to other switches. The switch that received the AAI may request a Domain ID by transmitting Request Domain_ID (RDI) fabric service. Only the Principal Switch can handle the RDI request. Other switches have to forward the received RDI to the Principal Switch.

5.4.1 RDI Request Processed by Principal Switch

When a RDI is received by the Principal Switch, it first checks its domain ID list to see whether this switch has requested a domain ID before. If it has requested a domain ID before and is asking for the same one, the Principal Switch grants this ID; if it is asking for a different one, this action is prohibited. Then a BF is transmitted to reconfigure the fabric. In the case that the switch has never obtained a domain ID before, the Principal Switch checks whether the requested ID is equal to zero or used by another switch. If it has, the Principal Switch grabs a new available ID and grants this one. If the requested ID is a valid domain ID and is available, the ID is assigned to the requesting switch. It is possible that all available domain IDs are used

and no more are available. In this case, a SW_RJT is transmitted with the reason “Domain ID not available”. Figure 5.4 shows the RDI processing by the Principal Switch.

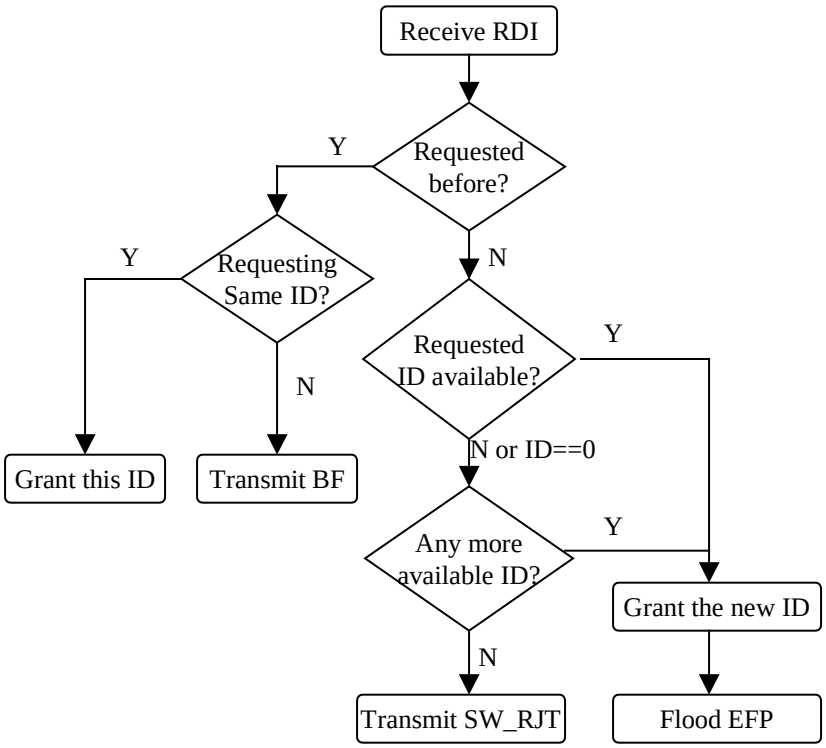

Figure 5-4: RDI Processing.

5.4.2 RDI Request Received by non-Principal Switch

If a non-Principal switch receives a RDI request, this RDI is forwarded to the Principal Switch. The switch developed in this thesis keeps track of which HBA is connected to the principal switch. After the RDI_ACC from the Principal Switch is received, it is forwarded back to the requesting switch. What the non-Principal Switch needs to do is to function as a conduit between the requesting switch and the Principal Switch. The non-Principal Switch will transmit an AAI when it detects that an unconfigured switch is connected to it.

Chapter 6

Name Server

6.1 Introduction

The switch name server, also called the directory server, maintains tables that correlate port identifiers with port attributes such as port name, node name, classes of services, FC-4 types, and IP addresses. These tables will be used to provide the port information in the local domain or any other specific domains. Name service registration, de-registration, and queries are managed by the name server through FC service protocols. The N/NL_Port attached to the switch utilizes the name server to get the information of other N/NL_Ports attached to other switch ports.

6.2 Name Server Structure

The names and attributes of the port attached to the fabric are saved in the name server. The name server maintains a database with an entry for each port [2]. Table 4-1 shows the organization of the database.

Primary Key	Indexed Fields	Secondary Key	Indexed Fields
Port Identifier	Port Name		
	Node Name		IP Address (Node)
	Class of Service		Initial Process Associator
	FC-4 TYPEs		Symbolic Node Name
	Symbolic Port Name		
	Port Type		
	IP Address (Port)		
	Hard Address		

Table 6-1: Name Server Database Organization.

After fabric login, devices are required to register to the name server, which has the well-known ID of 0xFFFFFC. In the registration procedure, the attributes are registered according to different Port_ID or Node_Name, whereas in the query procedure, information related to corresponding attributes can be obtained.

6.3 Implementation

In order to build a Fibre Channel switch on a PC, multiple Fibre Channel PCI host bus adapters (HBA) will be used as different switch ports. Each HBA maintains an ordered linked list with the information of all the N_Ports or NL_Ports attached to that HBA. When the link is initialized, the linked list is set to empty. Whenever a new port is connected to the HBA, a new entry of the linked list is added. The linked list uses data structure *port_info* to save the port attributes.

6.3.1 Define a Port

The *port_info* structure as shown below is defined to hold the attributes that a port could have.

```
struct port_info {
      u_int portID;           // Port Identifier
      u_int port_name_high;   // Port name high word
      u_int port_name_low;    // Port name low word
      u_int s_port_name[64];  // Symbolic port name
      u_int node_name_high;   // Node name high word
      u_int node_name_low;    // Node name low word
      u_int s_node_name[64];  // Symbolic node name
      u_int initial_pa[2];    // Initial process associator
      u_int ip[4];            // IP address (node)
      u_int class_service;    // Class of service
      u_int FC_4_types[8];    // FC-4 types
      u_int portType;         // Port type
      u_int ip_port[4];       // IP address (port)
      u_int fabric_name_high; // Fabric name high word
      u_int fabric_name_low;  // Fabric name low word
      u_int hard_address;     // Hard address
      u_int SET_FLAG;         // Attributes indicator
      struct port_info *next; // Pointer to next port
};
```

The FC_4_type field contains a map to indicate which Fibre Channel layer 4 services are supported. For example, every Fibre Channel device should support SCSI-FCP (Fibre Channel Protocol for SCSI), but not all devices need to support IP.

6.3.2 Registration and De-registration

Fibre channel switches require the nodes that are connected to the F/FL_Ports to perform a fabric login and name server login. The goal of fabric login is to ask for a domain identifier and area identifier from the fabric to construct a full-length port identifier. After the nodes get their identifiers, they perform login to the switch name server. This way, the nodes on other switch ports can know that a new node has been added to the switch.

As shown in Figure 4-3, the login frame contains the values of the port name, node name, and class validities of the new node. When a login frame is received to the name server, the name server checks whether any record has the same port identifier. If the record already exists, the name server goes further to check whether the port name, node name, and class validities of the existing record are the same as the values to be registered. In case any change is found, the name server will register the new values and transmit a Registered State Change Notification (RSCN) frame to other nodes. In fact, the port name, node name, and class validities are very unlikely to be changed during the time when the node is attached to the fabric. If the name server cannot find any record with the registering identifier, a new record is created for the new node and the corresponding port name, node name and, class validities are registered. Other entries of the new record are set to empty. These entries need other corresponding registration frames to fill them out.

The port identifier and node name are the two searching keys used in registration commands. Figure 6-1 shows an example of the format of the register FC-4 types (RFT_ID) frame, which is the most commonly used registration frame. When registration is done, the corresponding flag will be set to indicate that this attribute has been set. The flags will be checked during the query command processing. When the de-registration frame is received, a flag is set to indicate that this record is now empty.

Byte 0	Byte 1	Byte 2	Byte 3
Reserved	Port Identifier		
FC-4 Types			

Figure 6-1: RFT_ID Frame Payload.

6.3.3 Query

The attached nodes use query procedure to get certain information from the switch name server, such as which ports have already been registered and what characteristics they have. All the *port_info* entries other than the symbolic names can be served as the searching key. Some query commands only ask for certain values. For example, the GA_NXT is used to get the port record of the next higher valued port identifier. If there are no registered port identifiers higher valued than the value in the GA_NXT request, then the registered record of the lowest registered port identifier is returned. Query request such as GA_NXT can get one specific answer every time, whereas some other query requests may get more than one. GID_FT is the query request used to get all of the port identifiers which have been registered as support for the specified FC-4 type. One or more port identifiers, having been registered as support for the specified FC-4 type, are returned as shown in Figure 6-2. Each returned port identifier is preceded by an 8 bit control field. Bit 7 of the control byte is set to zero if the port identifier following the control field is not the last port identifier to be returned. Otherwise this bit is set to one.

Byte 0	Byte 1	Byte 2	Byte 3
0rrr rrrr	Port Identifier No. 1		
0rrr rrr	Port Identifier No. 2		
……			
1rrr rrr	Port Identifier No. n		

r: reserved

Figure 6-2: GID_FT Accept Payload.

The GA_NXT and GID_FT are the most often used query requests. A node can keep using GA_NXT until it gets a port identifier that is same as an identifier it received before. This way all the port identifiers registered to the name server are retrieved. Normally, using one GID_FT request can get all the port identifiers which support SCSP-FCP (Fibre Channel Protocol for SCSI).

Chapter 7

Routing

7.1 Introduction

In the Fibre Channel storage area network, each switch is a domain. Data frames can be forwarded to their destination domain when guided by the routing protocol. The Fibre Channel switch community has not had a routing protocol until the Fabric Shortest Path First (FSPF) routing protocol was accepted by the five switch companies in late June, 2000. The common routing protocol is a significant step toward the open Fibre Channel-based storage area network. FSPF is conceptually based on the Open Shortest Path First (OSPF) internet routing protocol. It is a Link State path selection protocol, which keeps track of the state of the links on all switches in the fabric topology. A cost is associated with each link. By choosing the path that has the least-cost, the protocol computes paths from a switch to all the other switches in the fabric.

The routing protocol has four major components:

- A Hello protocol
- A replicated topology database
- A path computation algorithm
- A routing table update

The Hello protocol is used to establish communication between a switch and the neighboring switches that are directly connected to it. When the Hello is done, the switch knows that the inter-switch link is active and available as a two-way path for frames. The topology database synchronization consists of an initial database synchronization and an update mechanism. The initial database synchronization is used when a switch is reconfigured; the update mechanism is used when there is a link state change. The

shortest path computation algorithm takes advantage of the link costs in the topology database to calculate the least-cost paths to every switch in the fabric. These results will be saved in the routing table.

The switch developed in this thesis implements the Hello protocol. Dijkstra's algorithm is used as the shortest path computation algorithm. For each destination switch in the fabric, the routing table keeps the information on which switch is used as the first hop and which output port is used to route a frame to that switch.

Due to time restrictions, only a small subset of the topology database update is implemented. The Hello protocol, the Dijkstra's algorithm, a simplified link state update method, and the routing table have been implemented. The periodic link state database synchronization, the reliable LSA flooding, the dynamic link cost calculation, and other features of the routing have not been added to the routing implementation.

7.2 Hello Protocol

After a switch detects that one of its ports is directly connected to another switch, it starts to transmit a Hello message to its neighbor. The Hello frame contains the domain ID and the port ID of the originating switch and the destination switch domain ID. At the beginning, the destination ID is unknown to the originating switch and is set to the broadcast address 0xFFFFFFFF. When a Hello frame is received, the switch knows the domain ID and port ID of the neighboring switch. If a switch detects that its own domain ID is in the received Hello frame, it concludes that the two-way communication with the neighboring switch has been established. The switch knows that the inter-switch link is active and available as a two-way path for frames.

The switch keeps transmitting Hello to its neighbors every 20 (Hello Interval) seconds to make sure that its neighbors are all active. If it has not received a valid Hello from the neighbor in 80 (Dead Interval) seconds, the local switch assumes that the link to the neighbor is down for some reason and the link will be initialized. This procedure happens to every switch. The *hello_timer* is in charge of transmitting

Hello frame every 20 seconds and the *dead_timer* monitors the incoming Hello frame. When the Hello process is done, the switch in the fabric knows the domain IDs which are connected to the local ports of the switch. Figure 7-1 shows an example of the Hello procedure. The port ID is also named as the area ID in the 24-bit Fibre Channel address (see Figure 3-3). The domain ID is assigned by the principal switch after E_Port initialization. Thus the first Hello frame is transmitted only when the switch gets a domain ID.

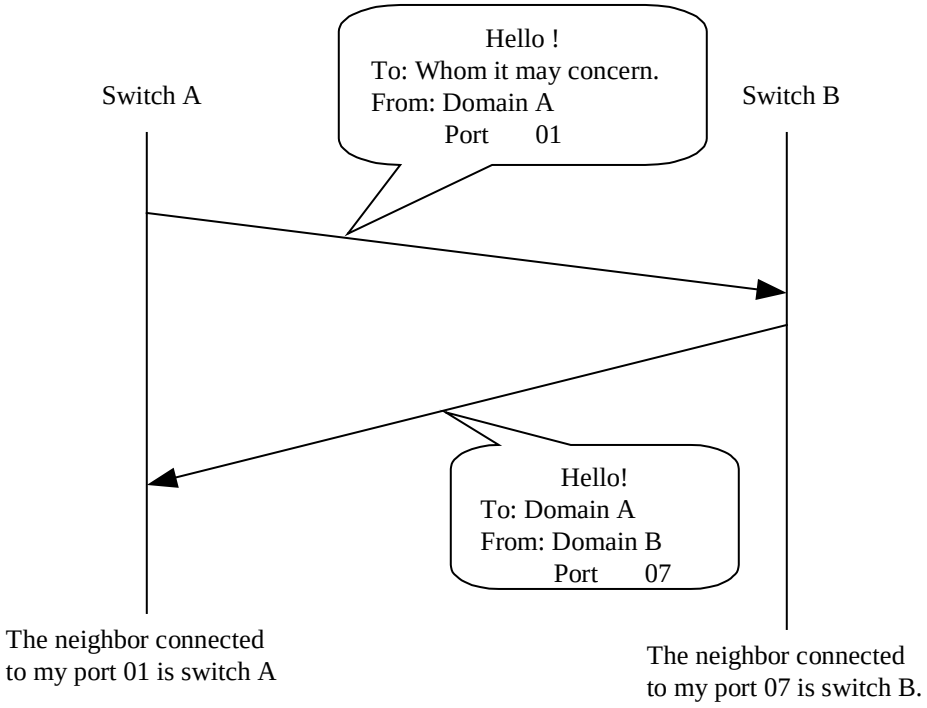

Figure 7-1: Hello Protocol

7.3 Link State Update

The topology database is the most important part of the link-state routing protocol. It is a replicated database. All switches in a fabric must have the exactly same copy of it at all times in order to prevent routing problems such as infinite loops or black holes. Keeping the topology database synchronized across all switches is achieved by Link State Update (LSU) messages. When an inter-switch link comes up or goes down or switches are added or removed from the fabric, an LSU will be issued by the affected switch to let other switches know that some changes have occurred. This way, every switch in the fabric can update their topology database. The LSU message contains one or more Link State Records (LSR) that make up the

topology database. Each LSR, which represents the link information for one switch, is the complete list of its inter-switch links. There is one link descriptor for each link. The LSU and LSR frame formats are shown in Figure 7-2. LSR age is the time in seconds since the record has been generated. It is used to flush the old records from the database. The number of the links in the LSR matches the number of ports that have valid links to their neighboring switch. In the real application, the link cost is calculated based on the available bandwidth, baud rate, and the administratively set factor. In this thesis, there are no calculations for the link cost. It can be set to a constant value. The path cost from the source switch to the destination switch is the sum of the traversed link costs. The Link State Acknowledge (LSA) frame is sent in response to the LSU frame. The purpose of the LSA is to provide a container to transmit LSR acknowledgements. It contains only the header of the LSRs that are acknowledged and does not have the link date descriptor fields.

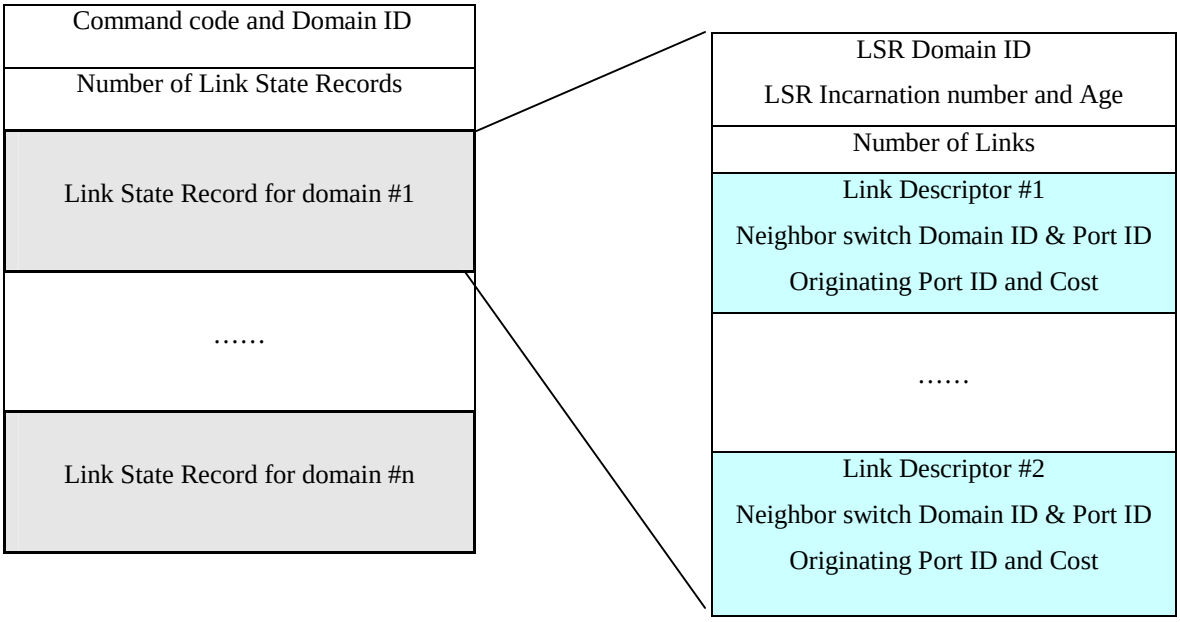

Figure 7-2: LSU and LSR Format

7.3.1 Initial Topology Database Synchronization

After the two-way Hello is established on a link, each switch exchanges its entire link state database with its neighbor using the LSU message. The recipient of that LSU checks the LSRs in the LSU. If an LSR contained in the incoming LSU does not exist in the local database, this LSR is added to the local database.

On the other hand, if the LSR already existed in the local database, the incarnation number and the age field of the LSR will be checked to help determine which of the two instances of an LSR is more recent. The incarnation number is a progressive number that identifies the LSR. When an LSR is first created, it has an initial incarnation number. The number is incremented by one each time a new instance of the LSR is issued. Thus, the one with a larger incarnation number is the most recent. The age field represents how long a particular instance of an LSR has been sitting in the link state database. When other LSR fields are equal, this field is used to determine who is younger. The old record will be updated if the incoming LSR is more recent. Figure 7-3 shows the progress of the topology database synchronization.

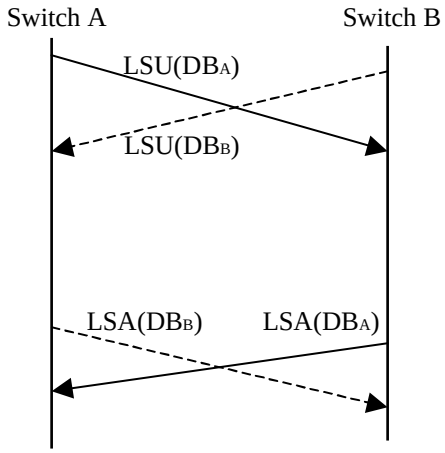

Figure 7-3: Topology Database Synchronization.

The entire database may be larger than one LSU can contain. Therefore, a database may need several LSUs to hold all the LSRs. In this case, the last LSU contains no LSR, which means the database exchange has been completed.

7.3.2 Topology Database Update

When the link state databases are in synchronization, further database updates can proceed. If a switch finds that one of its link state has been changed, this switch will issue an LSU which contains only the link state record of itself. Unlike the database synchronization process in which the LSU contains the entire database, during the database update procedure, only the changed LSR is contained in the LSU. Each

switch will retransmit the LSU to the entire fabric. Then all the switches know the changes and updates of their database.

After the new LSR has been transmitted and acknowledged on all of the switch's inter-switch links, the switch computes the paths to all the other switches in the fabric again. The routing table will also be updated. All the other switches in the fabric will do the same work, having received a new LSR.

7.4 Dijkstra's Algorithm

The link state database has all the topology information of the entire fabric. The costs between any inter-switch links are used for the shortest path computation. The least-cost path calculation algorithm used in this thesis is Dijkstra's algorithm. It functions by constructing a shortest-path tree from the initial vertex to every other vertex in the graph. Dijkstra's algorithm is one of the most efficient algorithms for solving the shortest-path problem. In a weighted graph (otherwise known as a network), it is frequently desired to find the shortest path between two nodes. The weights attached to the edges can be used to represent quantities such as costs, distances, or times. In general, the distance along a path is the sum of the costs of that path. Dijkstra's algorithm is implemented in this thesis in three stages.

1: Initialization. All the nodes in the network except the source are pushed into a linked list V. The link costs for the neighboring switches are obtained from the link state database.

2: Find the next node. Find the neighboring node, say node A, in the linked list V which has the least-cost path from the source node. Remove this node from V because this node was already calculated and will no longer be computed.

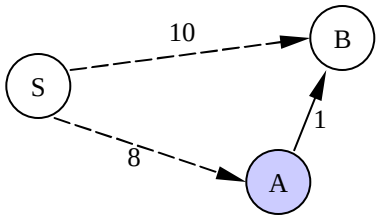

Figure 7-4: An Example of the Least-cost Path Update.

3: Update other paths. Check every node in the linked list V to determine whether this node is adjacent to node A. For instance, if node B is adjacent to node A, the least-cost path to node B may go through node A because node A was proven to be a least-cost path. Set the least-cost of node B to:

$$L(B) = \min[\, L(B), L(A) + w(A, B)]$$

where w(A, B) is the cost between node A and B. Figure 7-4 shows the example under this condition. After step 3, the path to node B should go through node A and the least-cost should be changed from 10 to 9.

The algorithm terminates when all nodes in the linked list have been calculated. At termination, every node knows its least-cost path from the source node. In addition, the least-cost paths establish a spanning tree of the original graph.

7.5 Routing Table

After the shortest path is computed, the switch knows the least-cost paths to every other switch in the fabric. What is really useful to the switch is the next hop to the destination because all nodes in a graph generate the same shortest paths in a graph. The first node on the shortest path is the next hop to the destination. During the Hello process, the switch already knew which neighbor is directly connected to which local port as mentioned in Chapter 7.2. Then the switch has the port ID which can lead to the destination switch. The next hop domain ID and the local port ID are stored in the routing table. Whenever a frame is received with the destination other than the local switch, the local switch checks the port ID for the destination switch from the routing table. Then the local switch forwards this frame through that port. An example of the Routing Table is shown in Figure 7-5.

Output from the Dijkstra's Algorithm

To:	Shortest Paths:
A	B A
B	B
C	C
D	C D

Routing Table

TO:	Port:	Next:
A	1	B
B	1	B
C	3	C
D	3	C

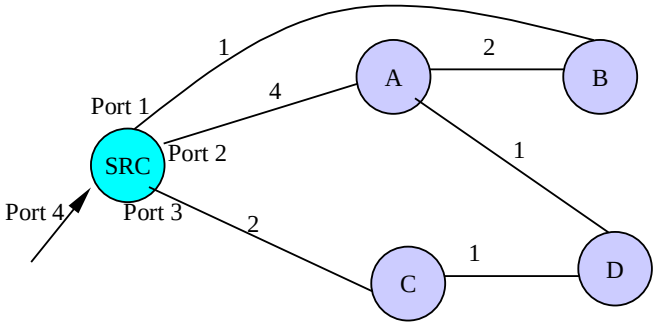

Figure 7-5: Routing Table.

Chapter 8

Testing and Evaluation

8.1 Introduction

The emulated switch was run through the test suites developed by the InterOperability Lab's Fibre Channel consortium of the University of New Hampshire. There are two standards which describe the operation of Fibre Channel switches: Fibre Channel Switch Fabric (FC-SW) and Fibre Channel Generic Services (FC-GS). The Fabric Loop Attachment (FC-FLA) mainly defines the behavior of the fabric attached N/NL_Port. The test suites that were run on this emulated switch were the FC-SW-1 and FC-GS-2 test suites. The FLA test suite is not developed for fabric ports, but the emulated switch can be developed as a FLA testing tool. The development of a switch testing tool will be introduced in this chapter. Apart from the conformance testing, tests to measure the performance of the emulated switch were also made.

8.2 Fibre Channel Conformance Testing

8.2.1 FC-SW-1 Testing

The FC-SW-1 test suite tests the conformance of a Device Under Test (DUT) to the Fibre Channel Switch Fabric Standard. The test suite essentially tests the following aspects:

- Class F Service
- E_Port Initialization
- Principal Switch Selection
- Address Distribution

There is an AL_PA bitmap in the loop initialization sequences such as LIFA, LIPA, LIHA, and LISA. The AL_PA bitmap shows which AL_PAs are selected by the devices on the loop. The FC-SW requires that if the LISA sequence indicates only one or two ports are on the loop, the switch shall try point-to-point link

initialization. The reason for this procedure is that the point-to-point flow control can save time by not using the arbitration process on the loop connection. Unfortunately, the AL_PA bitmap of the Tachyon chip is not available to programmers. Thus the testing items that require the AL_PA bitmap information are not testable. The setup for FC-SW tests is shown in Figure 8-1. Most of the test cases can be performed via setup Figure 8-1(a). However, some other test cases need to check whether the switch under test can forward a frame to other inter-switch links. In this case, a reference switch is needed to establish another inter-switch link with the switch under test. In this setup, which is shown in Figure 8-1(b), a multi-channel monitor such as I-Tech tester is required to trace four channels at the same time. The test report is not included here because it is very detailed and voluminous. Copies of the report can be obtained from the UNH InterOperability Lab's web page at www.iol.unh.edu/consortiums/fc/fc_linux.html.

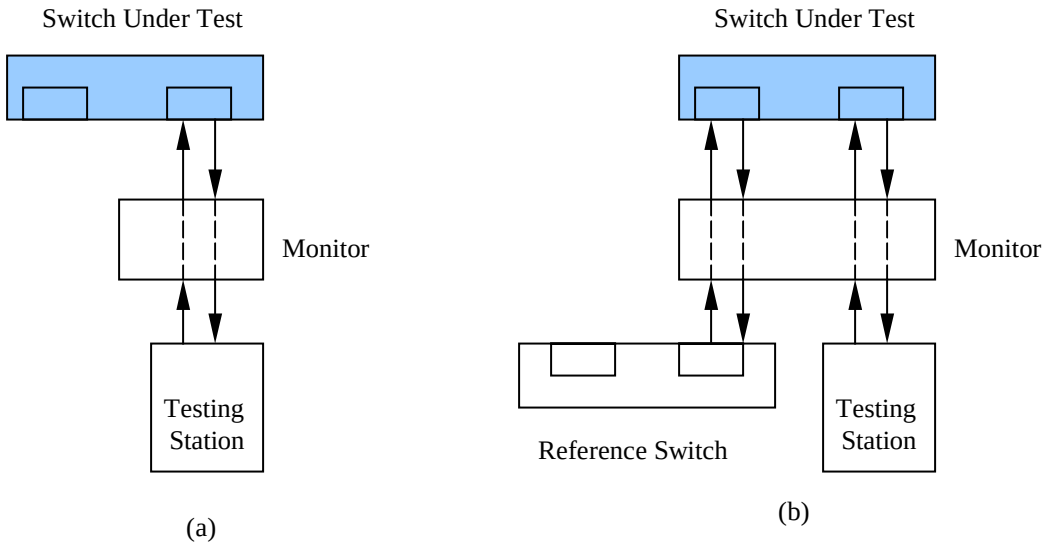

Figure 8-1: Setup for FC-SW Tests.

8.2.2 FC-GS-2 Testing

The FC-GS-2 test suite tests the conformance of the DUT to the Fibre Channel Generic Services standard. The test suite essentially tests the features of the switch name server. As mentioned in Chapter 6, there are three kinds of name server requests, which have more than thirty test cases:

- Query requests
- Registration requests

- De-registration request

In order to speed the testing process, the test cases are automated into two parts. One is the conformance testing and the other is the error-handling testing. The conformance testing is performed by the following procedure:

- Step 1: Transmit all registration requests to the name server with pre-defined parameters such as Port_Name, Node_Name, IP address, and so on.
- Step 2: Transmit all query requests to conform that all the parameters are correctly registered.
- Step 3: Transmit GA_NXT after DA_ID to make sure that DA_ID removes all the registered objects.

The error-handling test is performed in the following steps:

- Step 1: Transmit the ID registration requests, which use port ID as the searching index, to register pre-defined parameters.
- Step 2: Transmit the Node Name registration requests, which use Node Name as the searching index, with the Node Name that has not been registered. Expect to receive FS_RJTs with correct reason code.
- Step 3: Transmit all query requests and DA_ID request with the indexes which have not been registered. Expect to get FS_RJTs with correct reason code.

The setup for FC-GS tests is the same as the setup shown in Figure 8-1(a). The test report is not included here because it is very detailed and voluminous. Copies of the report can be obtained from the UNH InterOperability Lab's web page at www.iol.unh.edu/consortiums/fc/fc_linux.html.

8.3 Performance Testing

The emulated switch is built on a PC with 450 MHz Pentium II CPU and 128 MB RAM running Linux 2.2.14. The configuration for the tests is shown in Figure 8-2. The initiator used in the performance tests is a Hewlett-Packard (HP) Host Bus Adapter (HBA), which is installed on a host. The target is a Seagate hard drive, which is a fabric-attached disk. The monitor is the I-Tech IFC-3016 Fiber Channel Analyzer. IOMeter, a software for I/O testing, is installed on the PC which has the HP HBA. Using IOMeter, users can define the transfer request size, transfer time, read-write percentage, and other features of the I/O. The

same tests were performed on the emulated switch and on commercial switches. The results show that the emulated switch has a totally different data transfer mechanism from that of the real switches.

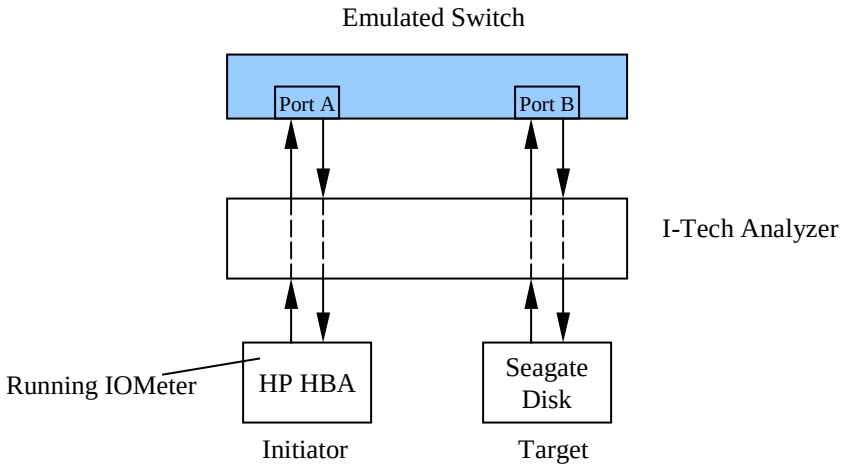

Figure 8-2: Setup for Performance Tests.

The frame forwarding between different ports in the real switch is achieved by hardware typically through the use of Application Specific Integrated Circuit (ASIC). When a real switch receives a sequence, the switch decodes the D_ID of the first frame and decides which port is the destination port. Figure 8-3 shows an example of the frame switching in the real switch and the emulated switch. In Figure 8-3 (a), the real switch receives a four-frame sequence on port A. After the switch knows that the destination port is port B, it begins transmitting the data frame to port B, even while port A is still receiving the data of that frame. The switch response time is the time from the point when port A receives the first SOF to the point when port B transmits the first SOF. Normally the response time of the real switch is as short as about 2.15 microsecond.

Because the emulated switch is a kind of software switch, its switching mechanism, shown in Figure 8-3 (b), is very different from that of the real switch. The emulated switch waits for the completion of the incoming sequence before it begins to forward this sequence to another port. From the figure, the emulated switch response time consists of two parts: the receiving time and the processing time. The receiving time represents the time cost for receiving the entire data of a sequence on port A, which is from

the SOF of the first frame to the EOF of the last frame on port A. After all the frames of a sequence are received on port A, it generates an interrupt to the PC on which the emulated switch is built. When the interrupt is received, the switch fetches the corresponding pointers on the registers of the Tachyon chip.

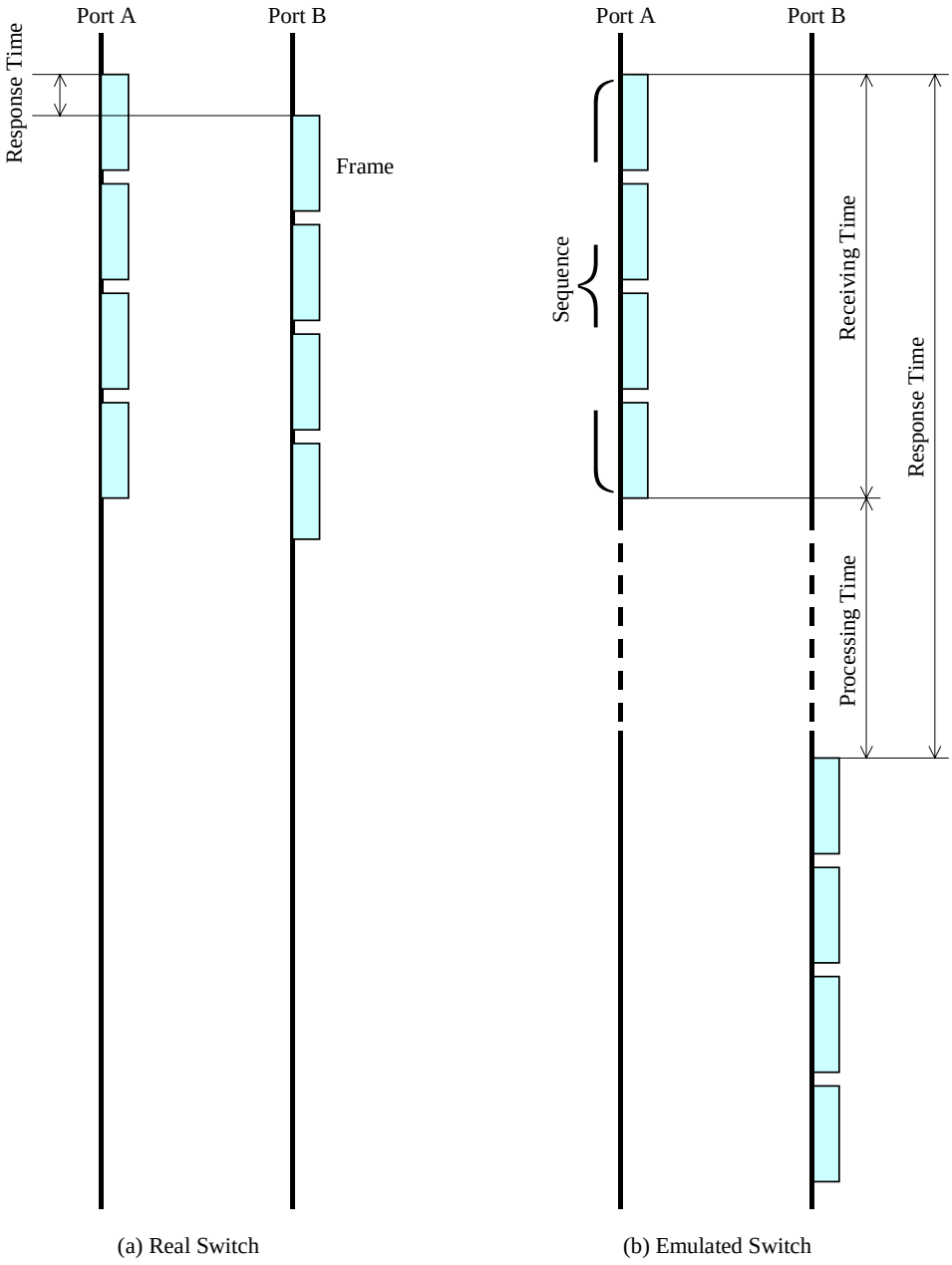

Figure 8-3: Data Transfer in the Real Switch and the Emulated Switch.

The switch then uses these pointers to reconstruct the frame header and data buffers of the received sequence, which is shown in Figure 4-2. After the switch notifies port B that the outbound data is ready, the Tachyon chip on port B begins transmitting the sequence. The processing time is between the EOF of the last frame on port A and the SOF of the first frame on port B. Both the receiving time and processing time vary with the length of the sequence, whereas the response time of the real switch is independent of the sequence length. The performance of the real switch and the emulated switch are in very different levels.

Sequence Length (Bytes)	**512**	**1024**	**2048**	**4096**
Receiving Time (µs)	5.1	10.0	19.6	39.5
Processing Time (µs)	50.0	80.6	141.4	260.0
Response Time (µs)	55.1	90.6	161.0	299.5

Table 8-1: Response time of the Emulated Switch.

Table 8-1 shows the response times of the emulated switch for different sequence lengths. The time value in Table 8-1 was the average of the sum of three results which were obtained under the same conditions. The discrepancy of the three results is less than 2% of their average. It can be seen that the processing time occupies about 90 percent of the entire response time, which means the software switching is the most time consuming part of the emulated switch. The receiving time depends mostly on the transmitting time of the frame and is proportional to the sequence length. The processing time is not proportional to the sequence length because the processing time consists of two parts: one is the time spent on moving the data from the host memory to the HBA by Direct Memory Access (DMA), another is the time spent on creating the frame header by collecting necessary information from the interrupt message. The DMA time is proportional to the sequence length, but the frame header creation time does not change with the sequence length. For the processing time listed in Table 8-1, a reasonable explanation is that the frame header creation time is about 20µs, thus the DMA time is about 30µs, 60µs, 120µs, and 240µs respectively. This DMA time is also proportional to the sequence length. Because it is very hard to get the

real time spent on DMA, this problem is up to further study. The performance of the emulated switch is limited by the hardware restrictions. It has to be noted that the performance does not affect the goal of this thesis: to develop a testing tool for switches and switch-attached devices.

Chapter 9

Testing Tool Development

9.1 Introduction

The main goal of this thesis is to build an emulated switch which can act as a testing tool for the Fibre Channel switch or switch-attached device testing. Even though there are several Fibre Channel switches in the IOL, they act as closed boxes and cannot be controlled to transmit certain frames and sequences of frames. A real switch cannot generate well-defined error conditions which are important in performing switch tests. Also more and more Fibre Channel vendors are requesting fabric frame-based testing. The switch developed in this thesis can act in two different roles depending on how the testing flag of the driver is set. If the testing tool is set to non-zero, the driver acts as a testing tool and intentionally generates predefined response to the incoming frames, otherwise the emulated switch works normally as a real switch.

9.2 Frame-based testing

For different Fibre Channel layers as shown in Figure 2-1, there are different test procedures, such as the FC-1 Test is for the physical layer and the AL test is for the Arbitrated Loop layer. The FC-2 and FC-3 layers are the General Services and Frame Protocol layers. The tests for these two layers are listed below:

- Switch Fabric (FC-SW)
- Generic Services (FC-GS)
- Private Loop Direct SCSI Attach (FC-PLDA)
- Fabric Loop Attachment (FC-FLA)
- IP

All these layer FC-2 and FC-3 tests are frame-based tests. Unlike the FC-0 and FC-1 tests, the frame-based tests require the testing station to generate a lot of frames. In the Fibre Channel Consortium of the UNH InterOperability Lab, there are three kinds of testing data generator:

1) UNH IOL Tiger Board: Used only for physical layer tests.
2) Finisar Trace Generator: Used for order-set-based tests, has 500-line limitation.
3) Xyratex Generator: Frame-based tester, but no support for Fabric.

There is no fabric frame-based generator in the market right now, but more and more fabric-related tests are requested by the vendors. This is the reason the IOL develops a frame-based tester for lab's use. The frame-based tester is a kind of interactive testing tool, which means the response of the testing tool depends on the contents of the received frame. A non-interactive testing tool does not have this kind of intelligence because the outgoing data of a non-interactive testing tool is predefined.

In most cases during a frame-based test, the testing tool has to respond to some frames correctly. A response frame must match the Originator Exchange Identifier (OX_ID) from the Device Under Test (DUT). Because the OX_ID varies from time to time, it is very difficult for a non-interactive tester to keep track of the OX_ID from the DUT. Figure 9-1 shows the switch E_Port initialization procedure. Suppose we are performing the rx_RDI test, a common test case in FC-SW-1, that checks whether the DUT responds properly to a received RDI. From Figure 9-1, we can clearly see that the testing tool has to transmit five ACCs before transmitting the RDI. The Acknowledgement frame, which is transmitted every time when a fabric service frame is received, is not shown in the figure. We can image how time consuming it is to perform this test on a non-interactive testing tool.

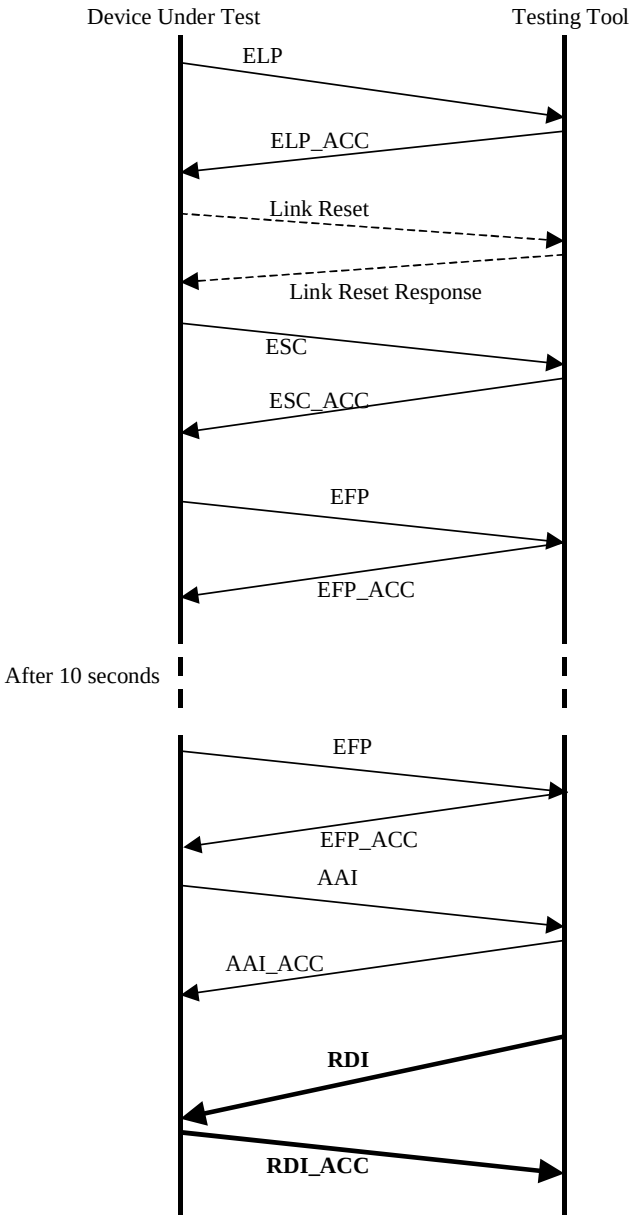

Figure 9-1: E_Port Initialization Procedure.

9.3 The *ioctl* System Call

This testing tool has two parts: one is the user interface and another it the device driver. The user interface accepts the testing request such as which fabric service frame is going to be transmitted. Because the user

interface runs in the user space, a connection between the user space and kernel space needs to be established, in order to pass testing requests to the device driver, which runs in the kernel space.

The *ioctl* system call offers a device-specific entry point for the testing tool to issue testing requests. The *ioctl* function is device-specific in which, unlike the common I/O methods such as *read* and *write,* it allows applications to access specific features of the hardware being driven. It is applicable for all I/O devices such as char devices, block devices, and network devices. In the thesis, the device driver is registered as a network device. The *ioctl* system call in the user space corresponds to the following prototype:

int ioctl(int fd, int cmd, …);

The first argument is a descriptor, which could be file descriptor or a socket descriptor. The second argument is the control command to the device. The dots of the third argument are there to prevent type checking during compilation. The actual nature of the third argument depends on the specific control command being issued (the second argument). Some commands take no arguments, some take an integer value, and some take a pointer to other data. The *ioctl* function used in this thesis has the following prototype:

*int ioctl(int sd, int cmd, struct *ifreq);*

The first argument must be a socket descriptor. The second argument is a socket control command. If the command is system predefined, it is handled by the operating system. If the command is not recognized by the protocol layer, it is passed to the device layer. The device-related *ioctl* commands accept a third argument from user space, a *struct ifreq **, which is the interface request structure used for socket *ioctl*'s. The *ioclt* implementation for sockets recognizes 16 commands as private to the interface: *SIOCDEVPRIVATE* throught *SIOCDEVPRIVATE + 15* [14]. Normally one command number is not enough for the I/O control. In most situations, a control command takes some parameters. For instance, if the testing tool is asked to transfer a fabric service frame, it needs to know the payload of that frame. The third argument contains the data related to the command, such as to which device this command is issued and what data is passed to the device. A pointer is used to pass arbitrary data to the device. When one of the private socket control commands is recognized, the *dev->do_ioctl* is called in the relevant device driver:

int (do_ioctl) (struct device *dev, struct ifreq *ifr, int cmd);*

The *do_ioctl* function receives the same *struct ifreq* that the user-space *ioctl* function uses. The using of the *ioctl* system call is shown in Figure 9-2 [13].

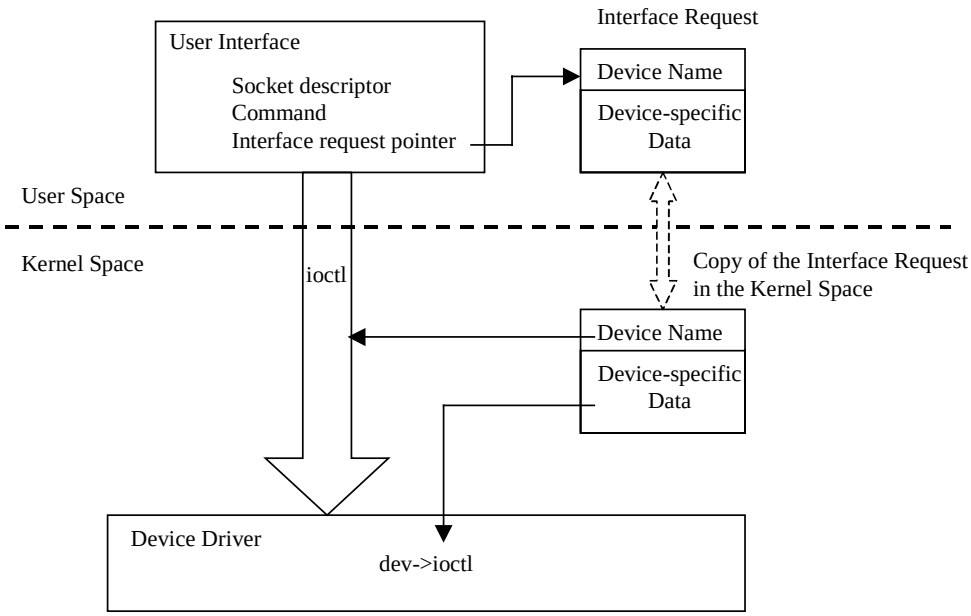

Figure 9-2: The ioctl() System Call.

The interface request is copied to the kernel space when the *ioctl* is called, and the kernel-space address of the interface request is then passed to the *do_ioctl*. After the driver processed the command, the interface request is copied back to the user space; the driver can thus use the private command to both receive and return data.

The testing tool developed in this thesis fully takes advantage of the features of the system call *ioctl*. Because there are only sixteen private control commands available, only the very frequently used testing case is allocated a command number. Other testing case commands are passed to the driver using the interface request data. Two testing tools are developed in this thesis: the switch testing tool and the non-switch testing tool.

9.4 Using the testing tool

The switch testing tool works only as a switch, which means the ports can only be initialized as E_Port, F_Port, or FL_Port. Because the switch port and non-switch port have very different behavior, they are separately implemented. Following is the switch testing tool user menu.

```
Initialize as FL_Port --->    1
Initialize as F_Port ---->    2
Initialize as E_Port ---->    3

Tx SW_ILS --------------->    5
Tx FCS command: --------->    8
Tx ELS :  --------------->    9

Tx SCSI command:  ------------>   11
Tx SCSI single-frame data:---->   13
Tx very generate  frame:------>   14

Tx an ABTS: ----------------------------> 80
Tx FCS all name_server commands: --------> 88
Tx FCS all error handling  commands:-----> 89

E_Port testing:
      1) rx_ELP.ELP: Higher E_Port Name ----------->   41
      2) rx_ELP.ELP: Lower E_Port Name ------------>   42
      3) rx_BSY_RJT.ELP: F_BSY -------------------->   43
      4) rx_BSY_RJT.ELP: P_BSY -------------------->   44
      5) rx_BSY_RJT.ELP: F_RJT -------------------->   45
      6) rx_BSY_RJT.ELP: P_RJT -------------------->   46
      7) rx_SW_RJT.ELP:        -------------------->   47
      8) rx_other_frames.ELP   -------------------->   48
      9) rx_no_valid_response.ELP ----------------->   49
      10) zero_list_rx_EFP.principal: zero List ---------->   50
      11) zero_list_rx_EFP.principal: non-zero List ------>   51
      12) non_zero_list_rx_EFP.principal: ---------------->   52
      13) rx_EFP_no_overlap.principal: ------------------->   53
      14) rx_EFP_overlap.principal: ---------------------->   54

Quit--------------------->      0
Selection Option:
```

From them menu, we can see that the E_Port (FC-SW) testing is embedded in the driver. After users inputs a command, the driver will work in the testing mode. For example, if the input command is 41, the “rx_ELP.ELP: Higher E_Port Name” test case, the driver is reinitialized to E_Port and waits for a incoming ELP. Then the driver takes the DUT’s E_Port name from the ELP frame received, increments the value of the name, and put the new name in to outgoing ELP frame. This action, of course, does not happen during a normal E_Port initialization process. When initialized as an F_Port or FL_Port, the testing tool can

be used for FLA testing. Different Extended Link Service frames can be transmitted through the testing tool from a pseudo public port ID.

The non-switch testing tool works as a non-switch device, which means the ports can only be initialized as N_Port, or NL_Port. The name server tests (FC-GS), private loop tests (FC-PLDA), and IP tests are heavily dependent on the testing tool. In the FC-GS, FC-PLDA, and FC-FLA tests, there are many similar test cases. For instance, in the name server tests, different name server requests are transmitted to the DUT. The only difference among these frames is the payload data. They have the same D_IDs, S_IDs, frame types, control bits, and so on. In this case, it is better to automate these tests so that all the similar frames can be transmitted together. The testing tool achieved the automated testing by reading the payload contents from files and issuing *ioctl*s to the device driver.

Chapter 10

Summary and Future Work

10.1 Summary

The lack of interoperability between switches from different vendors is delaying the widespread adoption of heterogeneous SANs. The goal for this thesis is to build a Fibre Channel switch environment in the UNH InterOperablity Lab (IOL) which can aid interoperability tests for switches or switch-attached devices. Currently, more and more Fibre Channel vendors are requesting fabric frame-based tests. Even though there are several Fibre Channel switches in the IOL, they act as closed boxes and cannot be controlled to transmit well-defined error conditions, which are important in switch testing.

An emulated Fibre Channel switch has been designed and implemented on an Intel-based PC running Linux by using multiple Fibre Channel host bus adapters. The switch, which complies with the FC-SW and FC-GS standards, is able to perform F/FL_Port or E_Port initialization, frame forwarding, name server management, principal switch selection, and domain ID distribution. A subset of the routing protocol, which includes the Hello protocol, Dijkstra's algorithm, and a simplified link state update method, has been implemented. A Fibre Channel switch testing tool, which can be used to perform the FC-SW, FC-GS, and FC-FLA tests, has also been developed on the emulated switch. The switch has accomplished the proposed goal of developing a tester for fabric interoperability testing.

10.2 Future work

The Fibre Channel switch has seven elements of interoperability:

1) Link Initialization
2) Principal Switch Selection

3) Domain ID Distribution

4) Distributed Name Server

5) Routing

6) Distributed State Change Notification

7) Zoning

The link initialization has been implemented except for some features that depend on the hardware support. The principal switch selection and the domain ID distribution have been fully implemented in this thesis. The name server developed in this thesis is a local name server. In order to collect all the port information on the Fibre Channel network, the distributed name server has to be developed. The routing and the distributed state change notification are partially implemented and need further development. The zoning remains undone.

The additional features that need to be implemented on the emulated switch are briefly described below.

- The distributed name server could be developed based on the current local name server. When a name server query request is received, the local name server only checks the local port database; whereas the distributed name server will broadcast the request to all switches in the Fibre Channel network and collect the results from all switches.
- The FSPF routing protocol could be implemented more accurately after the standard is fully published. The periodic link state database synchronization, reliable LSA flooding, dynamic link cost calculation, and other features could be added to the routing implementation.
- Currently, the RSCN is only sent to the local switches. The distributed state change notification could be added to let the entire fabric know the changes happened on the local switch.
- Zoning could be implemented to set zones for different groups of devices.
- The testing tool, a high level application, could be implemented to get the low level testing data from the driver using system call *ioctl*. The testing tool could be further developed to have a graphical user interface.
- Some storage management function could be added to the emulated switch. The switch could be more aggressive to transmit SCSI commands to find out the features of the attached devices. Then

these features, such as the device type (initiator or target) and the target capacity, could be used for storage management.

BIBLIOGRAPHY

[1] ANSI T11/Project 959-D/Rev 3.3, October 21, 1997, Fibre Channel Switch Fabric (FC-SW).

[2] ANSI T11/Project 1134-D/Rev 5.3, November 25, 1998, Fibre Channel Generic Services-2 (FC-GS-2).

[3] ANSI X3T11/Project 958D/Rev 3.5, August 7, 1996, Fibre Channel Fabric Generic Requirements (FC-FG).

[4] ANSI X3T11/Project 755D/Rev 4.3, June 1, 1994, Fibre Channel Physical and Signaling Interface (FC-PH).

[5] ANSI X3T11/Project 960D/Rev 4.5, June 1, 1995, Fibre Channel Arbitrated Loop (FC-AL).

[6] ANSI X3T11/Project 1235DT/Rev 2.7, August 12, 1997, Fibre Channel Fabric Loop Attachment (FC-FLA).

[7] Hewlett-Packard Company, May, 1996, TACHYON User's Manual, First Edition, Revision 5.0

[8] Tom Clark, “Designing Storage Area Networks,” Addison-Wesley. 1999.

[9] Robert W. Kembel, “Comprehensive Introduction to Fibre Channel,” Solution Technology. 1998.

[10] UNH InterOperability Lab, Fibre Channel Tutorial, http://www.iol.unh.edu/consortiums/fc.

[11] University of Minnesota, Fibre Channel Howto, http://www.globalfilesystem.org.

[12] Vineet M. Abraham, “Design, Implementation and Evaluation of a Fibre Channel Driver for IP on Linux,” MS Thesis, University of New Hampshire, August, 1999.

[13] Maurice J. Bach, “The Design of the UNIX operating system”, Prentice Hall, 1986.

[14] Alessandro Rubini, “Linux Device Drivers”, O’REILLY, 1998.

APPENDIX A: Abbreviations and Acronyms

AAI	Announce Address Identifier
ACC	Accept
ACK	Acknowledgement
AL	Arbitrated Loop
AL_PA	Arbitrated Loop Physical Address
ANSI	American National Standards Institute
ASIC	Application Specific Integrated Circuit
BB_Credit	Buffer to Buffer Credit
BF	Build Fabric
D_ID	Destination Identifier
DUT	Device Under Test
E_D_TOV	Error_Detect_Timeout Value
EE_Credit	End to End Credit
EFP	Exchange Fabric Parameters
ELP	Exchange Link Parameters
ELS	Extended Link Service
EOF	End of Frame
EOFn	End of Frame Normal
EOFt	End of Frame Terminate
F_BSY	Fabric Busy
F_RJT	Fabric Reject
F_S_TOV	Fabric_Stability_Timeout Value
FAN	Fabric Address Notification
FC	Fibre Channel

FC-GS	Fibre Channel Generic Services standard
FC-PH	Fibre Channel Physical and Signaling Interface standard
FC-SW	Fibre Channel Switch Fabric standard
FLA	Fabric Loop Attachment
FLOGI	Fabric Login
FSPF	Fabric Shortest Path First
GA_NXT	Get All Next
GCS_ID	Get Class of Service
GID_FT	Get Port Identifiers with the FC-4 Type
HBA	Host Bus Adapter
IOL	UNH InterOperability Lab
IPI	Intelligent Peripheral Interface
ISL	Inter-Switch Link
JBOD	Just a Bunch of Disks
LAN	Local Area Network
LISA	Loop Initialization Soft Assigned
LR	Link Reset
LRR	Link Reset Response
NCITS	National Committee for Information Technology Standards
NOS	Not Operational
OLS	Offline
OX_ID	Originator Exchange Identifier
FC-PLDA	Fibre Channel Private Loop Direct SCSI Attach
R_A_TOV	Resource_Allocation_Timeout Value
RAID	Redundant Array of Independent Disks
RCF	Reconfigure Fabric
RDI	Request Domain_ID
RFT_ID	Register FC-4 Types

R-RDY	Receiver Ready
RSCN	Registered State Change Notification
S_ID	Source Identifier
SAN	Storage Area Network
SCR	State Change Registration
SCSI	Small Computer System Interface
SCSI-FCP	Fibre Channel Protocol for SCSI
SOF	Start of Frame
SOFf	Start of Frame Fabric
SOFi3	Start of Frame Initiate Class 3
SW	Switch
SW_ACC	Switch Fabric Service Accept
SW_ILS	Switch Internal Link Service
SW_RJT	Switch Fabric Service Reject
ULP	Upper Level Protocol
WWN	World Wide Name