

POLITECNICO DI TORINO

II Facoltà di Ingegneria
Corso di Laurea in Ingegneria Elettronica

Tesi di Laurea

**Analisi di SAN
(Storage Area Networks)
di Grosse Dimensioni**

Analysis of Large-scale Storage Area Networks



Relatore:
Prof. Fabio Neri

Candidato:
Enrico Schiattarella

Ottobre 2002

This work is licensed under the Creative Commons Attribution-NonCommercial License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.0/> or send a letter to Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

The author kindly asks to be informed about usages of this work.

Please send an email to [schiattarella _at_ mail.tlc.polito.it](mailto:schiattarella_at_mail.tlc.polito.it)

*... because the only people for me are the mad ones,
the ones who are mad to live, mad to talk, mad
to be saved, desirous of everything at the sa-
me time, the ones who never yawn or say a
commonplace thing, but burn, burn, burn like
fabulous yellow roman candles exploding like
spiders across the stars and in the middle you
see the blue centerlight pop and everybody goes
“Awww!”*

Jack Kerouac, “On the Road”

Sommario

Introduzione

Al giorno d'oggi l'informazione è una delle risorse più importanti per ogni tipo di organizzazione, in particolare aziende e istituzioni. E' fondamentale per lo svolgimento di qualsiasi attività che il patrimonio informativo sia disponibile prontamente e in maniera continuativa.

I sistemi di storage tradizionali

La maggior parte dei sistemi informativi odierni è centrata attorno ai *server* che ricevono richieste dagli utenti su una rete locale o geografica e le soddisfano accedendo ai dischi e agli altri dispositivi ad essi collegati (nastri, memorie ottiche, etc.), detti dispositivi di "storage".

Ogni server è direttamente connesso ai dischi per mezzo di un'interfaccia dedicata e vi accede in maniera esclusiva. Il paradigma che prevede la connessione diretta e dedicata di server e dispositivi di storage è detto "Directly Attached Storage" (DAS).

La quantità di informazione che deve essere immagazzinata e protetta sta crescendo in questi anni in misura esponenziale. L'aumento del numero dei server e della capacità richiesta da ognuno di essi hanno messo in luce una serie di limitazioni di questo tipo di sistemi in termini di scalabilità, affidabilità e semplicità di gestione.

Al fine di superare queste limitazioni, sono stati introdotti nella connessione tra server e dispositivi di storage schemi e tecnologie mutuati dal mondo delle reti.

Le Storage Area Networks

Una "Storage Area Network" (SAN) è una rete che collega tra loro i server e i dispositivi di storage. Si tratta di una infrastruttura dedicata e separata dalla rete locale, capace di connettere ad alta velocità un gran numero di dispositivi distanti tra loro anche decine di chilometri.

Una SAN può essere realizzata impiegando diverse tecnologie di rete, a patto che queste siano in grado di fornire una infrastruttura di trasporto veloce, a bassa latenza e

priva di errori. La tecnologia ad oggi più diffusa per l'implementazione di SAN è Fibre Channel.

Fibre Channel si propone di coniugare le caratteristiche di velocità ed affidabilità di una connessione diretta con la flessibilità di una rete.

Nella sua configurazione più generica una rete Fibre Channel è costituita da un insieme di apparati di commutazione (switch) connessi tra di loro in una struttura a maglia. Questa infrastruttura è detta *fabric* e ha lo scopo di fornire canali virtuali ad alta velocità tra coppie di nodi.

Il ruolo degli switch in una fabric è duplice: da una parte devono commutare i pacchetti che ricevono, in modo che questi possano raggiungere la loro destinazione finale; dall'altra devono scambiarsi informazioni per coordinare le loro attività e offrire una serie di servizi ai nodi.

Tra i servizi più importanti erogati dagli switch vi sono: instradamento, assegnamento degli indirizzi, servizi di directory e controllo di accesso. Questi servizi riguardano l'intera rete e sono di natura distribuita; per questo motivo ad ognuno di essi corrispondono uno o più protocolli che gli switch usano per scambiarsi le informazioni necessarie.

Scopo del lavoro

Si prevede che in futuro le SAN Fibre Channel cresceranno di dimensione fino a includere centinaia di switch e migliaia di nodi.

Al crescere delle dimensioni della rete, aumentano il volume del traffico di controllo che deve essere scambiato dagli switch e il numero di operazioni che questi devono svolgere. Le inefficienze di un singolo switch possono compromettere il funzionamento dell'intera fabric. Da ciò si deduce che i protocolli utilizzati e le loro implementazioni software sono componenti critiche per la scalabilità di una SAN.

Scopo del lavoro descritto in questo documento è analizzare in dettaglio i protocolli impiegati in una fabric e sviluppare una piattaforma per testarne le implementazioni degli switch.

Il Large Network Emulator

Il modo più diretto per osservare il comportamento di uno switch in una SAN di grosse dimensioni è realizzare la SAN stessa. Questo in generale è particolarmente sconsigliato dato il costo, l'ingombro e la difficoltà di gestione degli apparati Fibre Channel.

Per ovviare a questi inconvenienti è stato realizzato il "Large Network Emulator" (LNE), uno strumento che consente di riprodurre il comportamento di una rete Fibre Channel senza implementarla fisicamente.

L'utente dà una descrizione della topologia della rete che vuole emulare, in particolare specifica da quali switch è composta e come questi sono interconnessi. Il LNE, una volta lanciato il processo di emulazione, trasmette sulle sue interfacce gli stessi pacchetti che gli switch costituenti la rete specificata originerebbero. Gli apparati connessi a LNE non sono in grado di distinguere la rete emulata da una reale e reagiscono come se fossero effettivamente connessi a una rete con quelle caratteristiche.

Variando dinamicamente la topologia della rete emulata o generando altri eventi si stimolano i processi che girano sugli apparati connessi a LNE e si possono raccogliere dati sperimentali sul loro comportamento.

Conclusioni

I protocolli utilizzati dagli switch per scambiarsi informazioni e le relative implementazioni software sono stati identificati come le componenti più critiche di una SAN ai fini della scalabilità.

E' stato realizzato uno strumento capace di emulare reti con topologie complesse e rapidamente variabili. L'utilizzo di LNE consente di raccogliere dati sperimentali sul comportamento di uno switch, in particolare può essere impiegato per:

- verificarne il corretto dimensionamento delle risorse di calcolo
- identificare colli di bottiglia e componenti suscettibili di ottimizzazione nelle implementazioni dei protocolli della fabric
- verificarne la capacità di gestire correttamente situazioni limite
- confrontare strategie e scelte alternative per l'implementazione dei protocolli

LNE si è rivelato uno strumento utile e efficace ed è probabile che in futuro il suo sviluppo continuerà, con l'aggiunta di nuove capacità di generazione di traffico e la realizzazione di interfacce utente più versatili e intuitive.

Table of contents

Sommario	III
Preface	IX
Acknowledgments	XI
1 Introduction	1
1.1 Information trends	1
1.2 Limits of directly-attached storage	2
1.3 Requirements for next-generation storage subsystems	3
2 Storage Area Networks	5
2.1 Overview	5
2.2 New storage architectures	7
2.2.1 Storage consolidation	7
2.2.2 Remote replication and disaster recovery	7
2.2.3 Server clustering	7
2.2.4 LAN-free, server-free backup	8
2.2.5 Storage resources management	8
2.3 Protocols and technologies	8
2.3.1 Fibre Channel	8
2.3.2 SCSI	9
2.3.3 Internet SCSI (iSCSI)	9
3 Fibre Channel Internals	11
3.1 Topologies	11
3.1.1 Point-to-Point	11
3.1.2 Arbitrated Loop	11
3.1.3 Switched Fabric	13
3.1.4 Port types	14
3.2 Protocols architecture	15

3.2.1	FC-0	15
3.2.2	FC-1	15
3.2.3	FC-2	16
3.2.4	FC-3	16
3.2.5	FC-4	16
3.3	Classes of service	16
3.3.1	Class 1 - Dedicated Connection	17
3.3.2	Class 2 - Multiplex	17
3.3.3	Class 3 - Datagram	17
3.3.4	Class 4 - Fractional	17
3.3.5	Class 6 - Multicast Connection	17
3.4	Naming and addressing	18
3.5	Sequences and exchanges	18
3.6	Frame format	20
3.7	Flow control	21
3.8	Error recovery	22
4	Fabric concepts and operations	23
4.1	Fabric model	23
4.1.1	Switches	23
4.1.2	Nodes	24
4.2	Addressing	24
4.3	Flooding	25
4.4	Port operations	25
4.4.1	Node login operations	25
4.4.2	Inter-switch link parameters exchange	27
4.5	Domain management	27
4.5.1	Fabric parameters exchange	28
4.5.2	Principal Switch selection	29
4.5.3	Domain_IDs distribution	29
4.6	Routing	32
4.6.1	Topology database	32
4.6.2	Hello protocol	34
4.6.3	Initial database exchange	34
4.6.4	Path selection	34
4.6.5	Database update	35
4.7	Distributed Services	36
4.7.1	Directory services	36
4.7.2	Management services and zoning	37

5	The Large Network Emulator	38
5.1	Fabric scalability	38
5.2	Fabric emulation	39
5.2.1	Nomenclature	41
5.3	LNE architecture	41
5.3.1	Design considerations	41
5.3.2	Architectural layers	42
5.3.3	Hardware and software platform	44
5.4	Component description	44
5.4.1	HBAs and drivers	44
5.4.2	Segmentation and reassembly unit	44
5.4.3	Topology Manager	44
5.4.4	Port Manager	45
5.4.5	Domain Assignment Module	45
5.4.6	FSPF Module	46
5.4.7	Command Dispatcher	46
5.4.8	Management facilities	46
5.5	Usage and command set	47
5.5.1	Configuration phase	47
5.5.2	Run phase	48
6	Conclusions	49
A	LNE emulation session	50
A.1	Scenario	50
A.2	List of generated frames	52
A.3	Frame analysis	54
A.3.1	Link parameters exchange	54
A.3.2	Fabric parameters exchange	55
A.3.3	Domain_ID assignment	55
A.3.4	Hello packets	57
A.3.5	Initial database exchange	58
A.3.6	Link state updates	58
A.3.7	Other frames	62
A.4	LNE fabric information	63
	Bibliography	65

Preface

In recent years various issues concerning scalability, reliability and manageability of storage systems have risen. Storage Area Networks have imposed themselves as the solution to these problems and are already widely adopted in large organizations. Given the rate at which stored data are increasing and the evolution of storage architectures, in the future SANs will grow steadily in size and complexity.

Fibre Channel has proved to be a valid network technology for SANs and is still being improved and expanded to satisfy the needs of complex network architectures and demanding applications.

The goal of the work described in this document is to analyze Fibre Channel SANs, focus on the components that might pose scalability issues and develop a platform to test them.

Chapter 1 presents traditional storage systems, discusses their limitations and states the requirements for their successors.

Chapter 2 introduces SANs as the model for next-generation storage systems, explains how they solve the problems identified previously and shows some architectures allowed by the presence of a SAN. Some of the technologies that can be used to build a SAN are briefly introduced.

Chapter 3 is dedicated to basic Fibre Channel concepts, such as architectural layers, topologies, frame format, etc. Fibre Channel is a very complex technology that aims to solve a broad range of challenging problems. A comprehensive description of Fibre Channel is outside the scope of this document, therefore only the topics that are essential to understand the work have been introduced.

Chapter 4 focuses on switched fabrics. These are the most important Fibre Channel networks because they unleash the power of SANs and allow them to fully provide their benefits. A switched fabric is a complex environment and switches, the devices that represent its building blocks, must perform many tasks to keep it operational. The role of the switches in the fabric, the most important operation they perform and the protocols they use are described.

Chapter 5 presents the Large Network Emulator (LNE), the tool that has been developed to observe the behavior of switches in very large fabrics and test critical components. The features and capabilities of the LNE are illustrated and its usage is explained.

Chapter 6 contains a summary of the goals that have been achieved.

Appendix A shows a LNE usage session. The LNE is connected to a real switch and emulates a simple topology. The generated traffic is captured and analyzed.

Wherever appropriate, we have proposed comparisons between Fibre Channel and TCP/IP, trying to expose differences and similarities and to investigate the principles and assumption that influenced the design of the two technologies.

Discussing the scalability issues of SANs and Fibre Channel devices might seem a bit premature at the time of writing. However, a number of trends constantly perceived by industry experts, suggest that today's SANs will become an important part of future networking scenarios. Among these trends are:

- the consolidation of resources at all levels (host, device, network, etc.)
- the convergence of different infrastructures (SAN, LAN, WAN) with multi-layer, multi-protocol devices
- a shift toward “utility computing”, the idea that information systems can be provisioned by specialized providers, just like electricity and telecommunication services.

If these trends persist, today's SAN switches can be considered among the forerunners of a new generation of networking devices.

Acknowledgments

My internship at Andiamo Systems has been a very educational and enjoyable experience. I consider it an excellent conclusion of my university studies and a solid starting point for my professional career.

I am sincerely grateful to prof. Silvano Gai for giving me the opportunity to join his group. He and all the people the team, with their human and professional qualities, have allowed me to achieve important results and have a really good time. To all of them goes my gratitude for being so friendly and helpful. Besides, I would like to acknowledge Dinesh G. Dutt, for directly supervising my work and making precious contributions, and Andrea Locatelli for his concrete participation in the development of the software.

I would also like to thank once more prof. Fabio Neri: I owe very much to his support and guidance during the university years.

Last but not least, I want to thank my family that has always been ready to help whenever necessary and has succeeded in the difficult task of advising me without questioning my decisions.

I dedicate this work to the people mentioned above and to all my friends on both sides of the ocean: our long walks and endless discussions have made it all possible.

Chapter 1

Introduction

Today information is a critical asset for enterprises, institutions and even individuals. Most corporate processes, including enterprise resource planning (ERP), supply chain management and customer relationship management (CRM) run on the company information system. With the advent of the Internet much business has moved on-line, requiring enterprise systems to interact with millions of users and handle the associated data flow. Data must be available promptly and continuously. Failure to access them directly translates in significant costs and loss of revenue.

1.1 Information trends

According to a report by the University of California at Berkeley [1], the world produces nowadays between 1 and 2 *exabytes* of unique information per year, the great majority of which is in digital form and is stored on magnetic media. This sheer amount of information needs to be processed, distributed and stored. Since the early days of computing, processing power has steadily grown at exponential rates and parallel/distributed computing techniques have been developed to further increase processing capabilities. In recent years, local and geographic networks have been deployed to distribute data and share resources, and their bandwidth has grown by orders of magnitude thanks to optical transmission media and advances in integrated electronics.

Data stored electronically is roughly doubling every year and, although the capacity of storage devices has followed similar trends, storage infrastructures are being stretched to their capabilities. The task of administering large amounts of data is getting more difficult and costly.

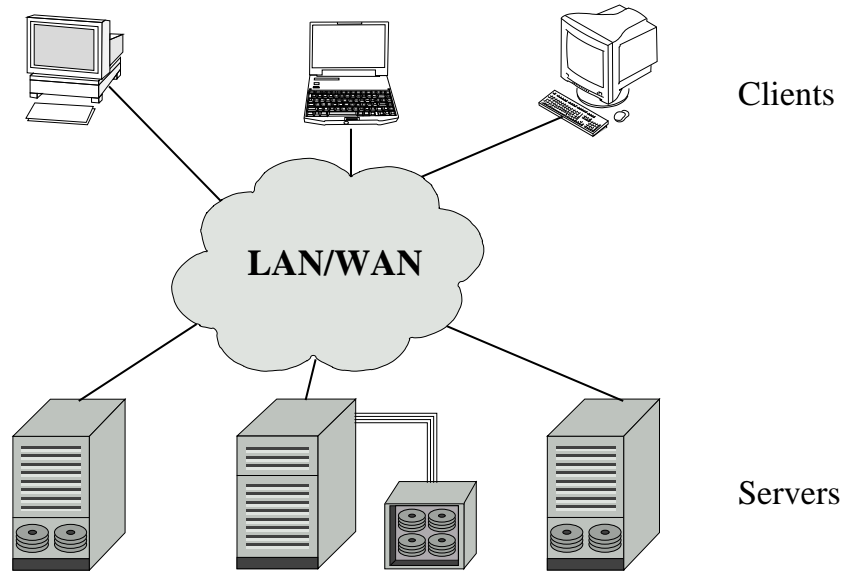


Figure 1.1. Directly-attached storage scenario

1.2 Limits of directly-attached storage

In the great majority of computing environments, servers are at the center of the information system. They store data on their disks and access them to satisfy the users' requests that arrive on the LAN. Each server is connected to its disks by means of fixed, dedicated channels, such as the SCSI¹ parallel bus.

Storage resources connected to a server are exclusively accessed and managed by that server. This paradigm is called "Directly-attached Storage" (DAS). An information system employing DAS is shown in figure 1.1. As servers grow in number and request additional capacity, several different problems arise. The most important are summarized below.

Scalability

The number of devices that can be attached to a disk controller is limited to few tens. Even with multiple controllers in the same server, the total available capacity might be insufficient.

¹Small Computer System Interface, a very common interface used to connect various peripherals to computers.

Performance

As the physical media is shared, adding devices results in less bandwidth being available to each of them.

Distance limitations

Parallel buses are limited in length few tens of meters by electrical issues, such as skew. Skew is a phenomenon typical of parallel transmission. Electrical pulses transmitted on different line of the parallel bus do not reach the target device at exactly the same time. If the delay between the first and the last arriving pulse is comparable to the time slot occupied by the pulse itself, the receiver cannot correctly decode the transmitted bit string.

Availability

Devices attached to the bus cannot be added or removed without putting the whole string off-line. This causes downtime every time the storage subsystem needs to be reconfigured.

Data protection

Each server must be equipped with proper devices (for example, tape drives) to backup its data. With hundreds or even thousands of servers, this is costly and quickly becomes an administrative burden, as each server must be backed up separately. If backup operations are performed through the LAN, the performance of the corporate network might be severely impacted for long time frames.

Efficiency

Disk space not used by a server cannot be relocated to another one. The administrators may need to buy and install additional storage devices even if free space exists on those already available.

A close look to the problems listed above suggests that many of them are intrinsic to the DAS model and cannot be solved just by enhancing the interface between the server and the disks.

1.3 Requirements for next-generation storage subsystems

The list of limitations of DAS can be used as a starting point to quantitatively state the requirements of future storage subsystems.

Such systems must:

- offer almost unlimited scalability, allowing the interconnection of thousands of devices distributed across distances spanning tens of Kilometers or more
- provide high, dedicated bandwidth, at least comparable to that offered by LAN technologies (few gigabits/s)
- allow large movement of data between storage devices (for example, for backup or replication purposes) without involving neither the servers nor the LAN
- allow the reconfiguration of the system and almost any other maintenance operation without downtime
- offer advanced and centralized management capabilities

In the rest of the document, paradigms and technologies able to meet such requirements will be illustrated.

Chapter 2

Storage Area Networks

In this chapter Storage Area Networks (SANs) are introduced as a mean to overcome the limitations of directly-attached storage and meet the requirements for new storage subsystems stated previously. Protocols and technologies that can be used to build a SAN are briefly introduced.

2.1 Overview

A SAN is a dedicated network connecting servers and storage devices. It is separated from either the LAN or the WAN. Servers and storage devices are nodes of the SAN, which allows meshed, any-to-any connectivity. This is often called “networking behind the servers”. The layout of a SAN environment is shown in figure 2.1.

The introduction of networking concepts and technologies as a replacement of a single, direct connection allows to completely rethink the server/storage relationship and build new storage infrastructures. SANs are networks in all respects and present all the features typical of networking technologies. The most important characteristics inherited from the networking world are:

- serial transport, that allows shipping of data over long distances at high rates
- packetizing of data, to achieve high efficiency and fair sharing of network resources
- addressing schemes that support very large device population
- routing capabilities, to provide multiple, redundant paths between source and destination devices
- a layered architecture: by dividing the architecture in multiple, independent layers, different protocols can be transported at the upper layers and different interfaces can be used at the lower ones.

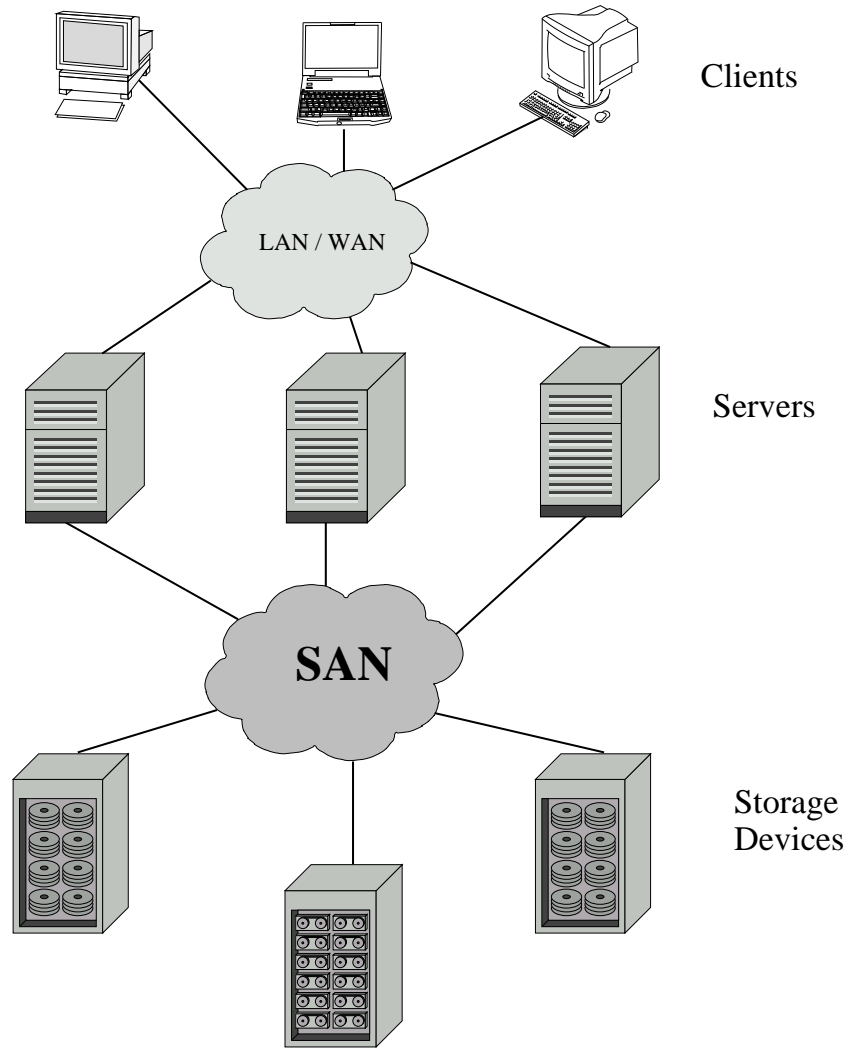


Figure 2.1. Storage Area Network layout

It is important to remember that servers, operating systems and applications still expect from the storage interface a “channel-like” behavior, which means high-speed, low-latency, error-free communications. Networking technologies used to implement SANs must therefore be carefully chosen and implemented in order to satisfy the peculiar requirements of storage traffic.

2.2 New storage architectures

The deployment of a SAN allows storage subsystems to be built in new ways, providing several benefits. Storage devices are now at the center of the information systems and servers become the frontend towards the users.

Even though some of the solutions described below were partially available in DAS environments¹, their effectiveness can be fully realized only when a SAN is deployed. Concrete examples and real-world case studies can be found in [2] and [3].

2.2.1 Storage consolidation

As servers are no-longer directly connected to disks, all the disks can be physically relocated in one or more disk arrays. Disk arrays are devices able to host tens or hundreds of physical disks. By using the management interface of the disk array, the storage administrator can allocate to each server a proper fraction of the total capacity, given by the sum of the capacity of the hosted disks. Additional space can be provided simply by adding disks to the array and reconfiguring it. Storage consolidation can take place even across multiple disk arrays.

2.2.2 Remote replication and disaster recovery

Disk mirroring is a method used to protect the storage subsystem from faults. A pool of physical disks of equal capacity is combined in a single, virtual disk of the same capacity. Data written to the virtual disk is physically stored on all the disks in the pool. If any of the disks in the pool fail, data is immediately available on the others and the server can continue its operations without disruption. As a SAN can connect devices located tens of Kilometers away, data can be replicated on remote sites, providing protection even in case of disasters, such as natural calamities or terrorist acts.

2.2.3 Server clustering

A cluster is a set of servers working concurrently on the same set of data. Clustering provides higher performances (as the servers work in parallel) and higher reliability (if

¹Most of the times achieved by combining the SCSI bus with RAID technologies.

one of the server fails, it simply goes out of the cluster). Although complex issues exist at the operating system level (process relocation, concurrent data access, etc.) a SAN effectively promotes clustering because it allows multiple servers to access data on a single disk.

2.2.4 LAN-free, server-free backup

As SANs allow communication between storage devices, data can flow directly from disks to tapes. Data stored in multiple disk arrays can be backed up using large, shared tape drives. All operations are scheduled and managed from a single, central location. Data flow does not impact the LAN and does not involve the servers.

2.2.5 Storage resources management

The ability to have a consistent and unified view of all the storage devices greatly simplifies management and planning. All the resources can be monitored and allocated from a single, centralized location. This reduces staff costs and the total cost of ownership (TCO).

2.3 Protocols and technologies

SANs can be built using different networking technologies, however storage traffic is very demanding and not all the existing technologies are able to satisfy its requirements.

2.3.1 Fibre Channel

Fibre Channel is a multi-purpose, standard-based networking technology. It tries to combine the performance and reliability characteristics of an I/O interface with the flexibility and connectivity of a network. Thanks to its layered architecture, it supports multiple serial transmission media² (optical fiber and copper) at gigabit speed and is capable of transporting multiple protocols (SCSI, IP and others).

Fibre Channel was specifically designed for computing environments and is based on the assumption that the transport media is reliable³, therefore error recovery mechanisms are reduced to a minimum and are mostly left to upper layer protocols. All the components of the network have been specifically designed to avoid frame dropping due to congestion. A simple, credit-based mechanism is used for flow and congestion control. Data is fragmented and encapsulated in network protocols with minimum overhead

²The English spelling of the word “fi bre” in “Fibre Channel” was chosen to highlight the fact that other media are supported besides the optical fi ber.

³Standards require bit error rate to be less than 10^{-12} .

in order to achieve high efficiency. These characteristics of the Fibre Channel data-path make an almost fully hardware-based implementation feasible, with minimum software intervention.

2.3.2 SCSI

The SCSI bus is the most common interface used to attach servers and large computers to storage devices. Work on SCSI began in 1982 as an attempt to provide a standard interface for computer peripherals and the first release of the standard (known as SCSI-1) was released in 1986.

Besides describing the physical layer of the interface, the standard defines a transaction protocol and a command set for a variety of peripherals, including all storage devices.

SCSI I/O operations on storage devices are *block oriented*, which means that the host sees an attached device as a set of contiguous blocks and does not have to worry about how data is physically stored and retrieved. All the intelligence required to execute SCSI commands is embedded in the storage devices. Complex operations such as formatting a disk or mounting a cartridge in a tape library can be triggered with a single SCSI command. When the user issues a request to access data, the operating system running on the host determines the appropriate sequence of SCSI commands and transmits them on the physical media. The host is connected to the SCSI bus by means of a “host bus adapter” (HBA).

As SCSI has become an industry standard for servers and is supported by all operating systems, the decision was taken to leverage its command set and protocols for I/O operations on a SAN. This allows the transition from DAS to SANs with minimum issues on the server side. SCSI-3, the latest version of the SCSI standards, includes a protocol called “Fibre Channel Protocol” (FCP) that defines the transport of SCSI traffic over Fibre Channel.

2.3.3 Internet SCSI (iSCSI)

TCP/IP is the most widespread protocol suite in the networking world. It is the foundation of the Internet and is also used extensively over Ethernet in corporate LANs. The question arises whether it is feasible to use such a popular and consolidated technology to transport storage traffic.

The implementation of a SAN based on TCP/IP technology would provide several benefits. Ethernet is a mature but evolving technology, providing progressively decreasing costs and increasing performance. The SAN might be built using the same equipment already used in the corporate LAN, reducing the total number of devices and the need for trained IT staff. In general, the convergence at the infrastructure level of the LAN and the SAN could greatly reduce implementation and maintenance costs. Besides this, storage

traffic transported on TCP/IP might easily go past the boundaries of the LAN, reaching the Internet or being transported on carrier WANs.

The major problem is that TCP/IP protocols were not designed for such a specialized application as storage networking. TCP assumes that the network is potentially unreliable and instable and employs complex mechanisms to achieve optimal link usage, control flow and deal with error conditions. For this reason, TCP implementations are heavily software-based and demand significant processing resources to the host. IP packet format and addressing scheme require some deal of processing by intermediate devices for packet routing and forwarding. TCP, IP and Ethernet headers introduce a considerable overhead that greatly reduce the efficiency of the network architecture. In general TCP/IP networks are based on a “best-effort” paradigm which is not acceptable for storage traffic. Some of the problems cited above can be mitigated by using LAN devices with QoS capabilities and network interface cards (NICs) with hardware acceleration for some of the tasks performed by TCP.

Given the significant interest in using IP-based networks to transport block storage traffic, and believing in its feasibility, the Internet Engineering Task Force (IETF) has created a “IP Storage” (ips) working group⁴. The goal of the group activities is to provide mechanisms to encapsulate SCSI (and possibly Fibre Channel) protocols in IP-based transports without requiring modifications to either the encapsulated or the encapsulating protocols unless strictly unavoidable.

The group has produced several documents, including [4] and [5] that describe *iSCSI*, a standard to transport SCSI protocols over TCP/IP networks. The standard tries to address all the issues that have risen, including the transport itself, security and management.

At the time of writing *iSCSI* has not yet been standardized, however *iSCSI* products are already on the market. Even though they are not able to offer the performances and features of Fibre Channel products, they seem to be valid building blocks for a SAN.

In the rest of the document, only Fibre Channel SANs will be discussed. IP networks and their scalability issues are well-know topics and extensive literature about them is available.

⁴<http://www.ietf.org/>

Chapter 3

Fibre Channel Internals

This chapter briefly introduces Fibre Channel terminology and concepts needed to understand the rest of the document. A detailed description of these topics can be found in [6] (and the other the standards cited along the way) or in [7].

3.1 Topologies

In a Fibre Channel network a *node* is the source or destination of the information flow. Each node has one or more *ports* that connect it to the network through *links*. Each link is composed by a pair of fibres, one for transmission and the other for reception.

Fibre Channel provides for three distinct interconnection schemes or *topologies* to build the network infrastructure: Point-To-Point, Arbitrated Loop and Switched Fabric. No matter how the network is physically structured, the goal of Fibre Channel is always to provide a full-duplex, high-bandwidth, reliable connection between two nodes.

3.1.1 Point-to-Point

A Point-To-Point topology is simply a direct connection between nodes (figure 3.1). As the connection is direct and dedicated, there is no need for media access control or routing. The full bandwidth of the link is available to the connected nodes.

3.1.2 Arbitrated Loop

The Arbitrated Loop is a connection scheme where each device is connected to the two adjacent ones to form a ring, as shown in figure 3.2. Frames are transmitted from one node to the next one, until they ultimately reach the destination node.

The loop topology can be physically implemented by daisy-chaining devices (connecting the transmission fibre of a node to the receive plug of the next one) or by employing a

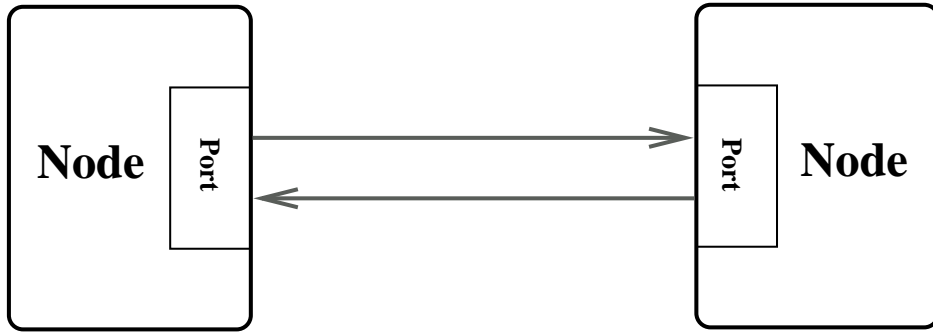


Figure 3.1. Point-to-Point topology

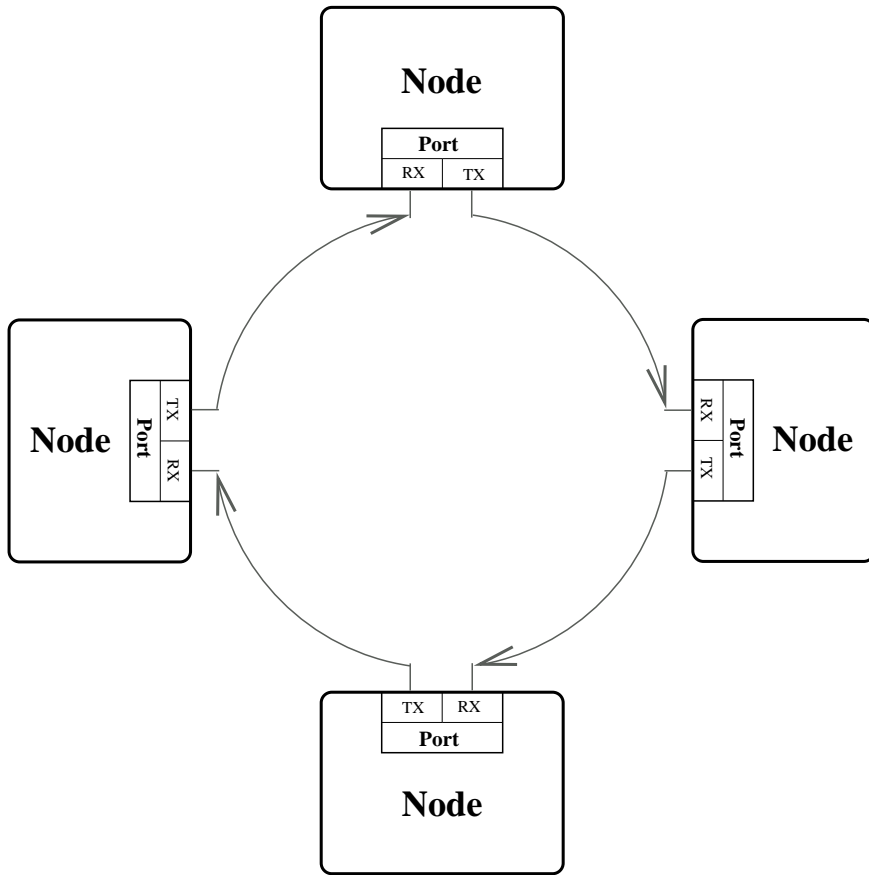


Figure 3.2. Arbitrated Loop topology

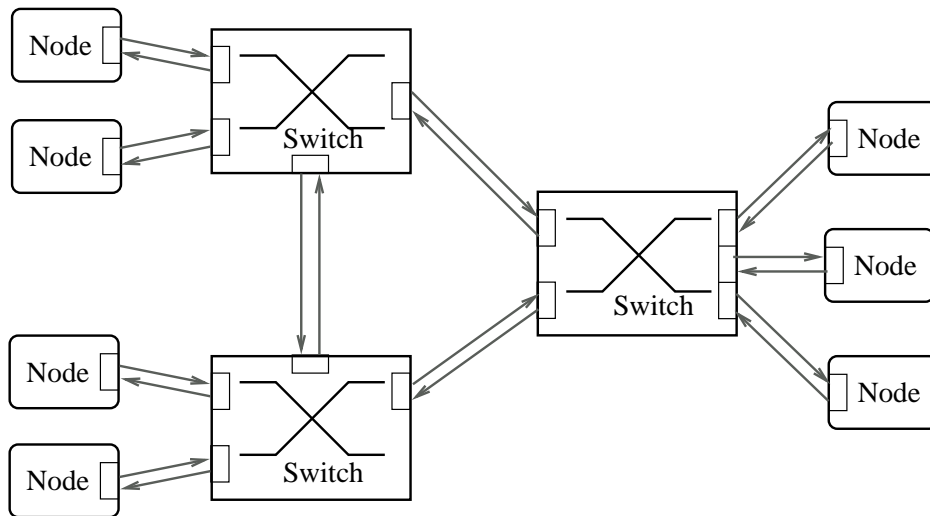


Figure 3.3. Switched Fabric

hub that internally implements the loop and transforms the physical topology into a star. This way is generally preferred because if a device fails or needs to be put off-line, the hub can automatically bypass it with no need for re-wiring and without disruption of the loop activities. Hubs operate only at the physical layer and do not participate in any activity on the loop.

The physical media is shared and only one device can be active on the loop at a time, so an arbitration mechanism is employed. As many devices contend for access, the average bandwidth available to each of them is inversely proportional to the total number of devices on the loop. According to the standards, there can be up to 127 ports on a loop.

3.1.3 Switched Fabric

The Switched Fabric is the most versatile and flexible interconnection scheme. It makes use of devices called *switches* that connect to other switches and nodes. A sample fabric is shown in figure 3.3.

Switches support multiple concurrent data flows, so the bandwidth available to a device is not shared with other devices. A switched fabric can theoretically interconnect millions of devices, providing almost unlimited scalability. As switches provide routing capabilities, the fabric can be arbitrarily meshed and multiple paths between source and destination are considered for redundancy.

The fabric is a complex and variable environment, where links can change status and nodes can join or leave at any time. Switches use several protocols to distribute to other

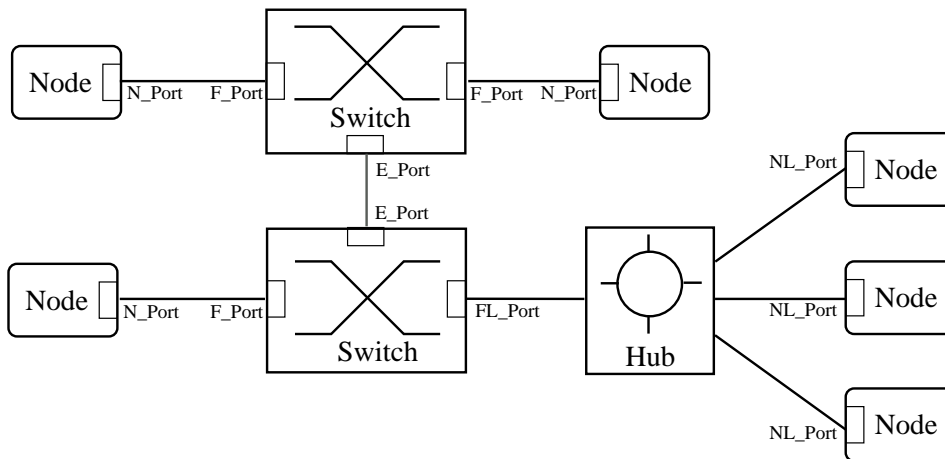


Figure 3.4. Hybrid topology with switches and hubs

switches the information they need to perform their tasks, including routing, address assignment, directory services, access control. Node are not involved in fabric operations, they only exchange some control traffic with the switch they are attached to.

Switches are costly and complicated devices. As the performance they offer are not always indispensable, hybrid topologies employing both switches and hubs can be built. In this case, devices that do not need the full bandwidth of the link are connected in a loop together with one port of a switch that provides connectivity to the rest of the fabric. In figure 3.4 a generic hybrid topology is shown; port names are explained below.

3.1.4 Port types

Each topology requests specific behavioral and functional characteristics to ports. For example a port connected to a loop must be able to participate to arbitration sessions. Ports performing different roles in the network get different names:

- A node port that connects to another node or to a switch is called *N_Port*.
- Ports that connect two switches are called *E_Ports*.
- Switch ports connected to a node are *F_Ports*.
- Node ports and switch ports that are connected to a loop are respectively called *NL_Ports* and *FL_Ports*.

Node and switch ports capable of operating both in a loop and in a direct connection are called *Nx_Ports* and *Fx_Ports*. A link connecting two E_Ports is called *Inter-Switch Link* (ISL).

3.2 Protocols architecture

Fibre Channel protocols are organized in a layered architecture shown in figure 3.5. If a node has multiple ports, it has a separate instance of the levels FC-0 to FC-2 for each of them and a unique instance of FC-3 and FC-4.

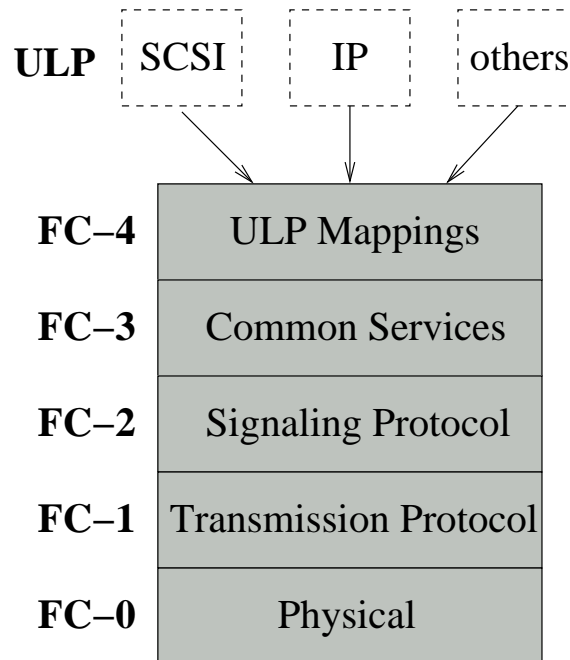


Figure 3.5. Fibre Channel protocols architecture

3.2.1 FC-0

This layer contains the description of the physical interface, including supported media types (multi- and mono-mode optical fiber, twisted pair, copper coax, etc.), cabling requirements, connectors, transmitters and receivers specifications (wavelengths, data rates, signal levels, etc.).

Supported data rates, according to the standard [8], are 1.06 Gbit/s, 2.12 Gbit/s (commercially available) and 4.25 Gbit/s. 10 Gbit/s specifications are under development.

3.2.2 FC-1

The transmission protocol layer defines the encoding scheme used on the link and low-level link control facilities (initialization, reset, etc.).

Data are encoded before transmission on the physical media and decoded upon reception. The encoding scheme that Fibre Channel uses is called “8B/10B” and was originally developed by IBM. Every 8-bit sequence is encoded in a 10-bit one that has important characteristics that increase the robustness of the transmission. The encoding guarantees that in every sequence there is a minimum number of transitions from the “low” to the “high” level (or vice-versa) and that an overall balance of high and low bits is maintained on the link. This prevents receiver synchronization problems and aids in error detection.

3.2.3 FC-2

The signaling protocol defined at this layer specifies the rules to transfer data blocks end-to-end. It defines the frame format, addressing, information unit segmentation and reassembly, flow control, error detection and recovery.

3.2.4 FC-3

The FC-3 layers deals with services that are common across multiple Nx_Ports of a node. As no such services have been defined, this layer can be considered a placeholder for future facilities.

Functions that might fit at this level are encryption and compression.

3.2.5 FC-4

Fibre Channel is capable of transporting multiple upper-layer protocols (ULP). This layer defines how each of this protocols must use the services offered by the lower layers to transport its information units across the Fibre Channel network.

For example, FCP [9] is the mapping for SCSI-3 and specifies how SCSI commands, data, parameters and status information are to be packed into FC-2 entities. The IP mapping [10] deals with the encapsulation of IP packets in Fibre Channel frames and the resolution of IP addresses.

3.3 Classes of service

In order to accommodate the needs of different traffic typologies (in terms of bandwidth, reliability and timeliness of delivery) Fibre Channel provides a number of “classes of service”. Classes 1, 2 and 3 can be used with any topology, Classes 4 and 6, on the contrary, require the presence of a fabric.

3.3.1 Class 1 - Dedicated Connection

This class of service provides a connection oriented mode of operation. A dedicated, full-bandwidth, bidirectional, connection is established between the source and the destination nodes. In-order delivery of frames is guaranteed and received frames are acknowledged. The circuit is maintained until one of the parties requests its removal. The source and destination nodes cannot be involved in other communications as long as the circuit is active¹.

3.3.2 Class 2 - Multiplex

The service provided by this class is connectionless, as no dedicated circuit is established between the parties. Consecutive received frames might come from different sources and the transmitter might send consecutive frames to different destinations. Received frames are acknowledged but frames might be delivered out of order. There is no guarantee that a certain amount of bandwidth is available for the communication to take place.

3.3.3 Class 3 - Datagram

Class 3 service is similar to Class 2 but there is no acknowledge of frame reception, so there is no guarantee that a frame has been received. No flow control between sender and receiver is available. Error recovery mechanisms are completely left to upper layers.

3.3.4 Class 4 - Fractional

Class 4 service provides multiple, connection oriented “virtual circuits”. Each of these circuits is set up individually, possibly with different destination nodes, and reserves a fraction of the available bandwidth of the link. On every circuit in-order delivery of frames and the bandwidth initially requested are guaranteed. Delivered frames are acknowledged.

3.3.5 Class 6 - Multicast Connection

This class provides a one-to-many, connection oriented, unidirectional service. Full bandwidth and in-order delivery of frames is guaranteed for the connection. When a frame is delivered, the destination node sends an acknowledge frame to a Multicast Server that collects all the responses and notifies the sender.

Although the standards give detailed specifications for all the classes enumerated above, some of them have no practical use in the SAN world, while others have proven to

¹A feature called “Intermix” is provided to allow the flow of frames external to the Class 1 connection. This is fundamental when a fabric is in place and important control frames must be exchanged.

be too tough or too costly to implement. Most commercially-available switches support only Class 2 and Class 3 services.

3.4 Naming and addressing

Nodes and ports in a Fibre Channel network are uniquely identified by 64-bit values called *Name_Identifiers*, statically assigned by the manufacturer. *Name_Identifiers* are world-wide unique, so they are commonly referred to as *World Wide Names* (WWNs) or simply “names”. WWNs include a “Organizationally Unique Identifier” (OUI) assigned to each company by the IEEE Registration Authority².

Every node has a single node name and one port name for each port. The same applies to switches and their ports.

WWNs are not addresses because they are not used for frame delivery on the network. The actual addresses used in a frame header to identify the source and destination port are 24-bit values respectively called *Source_ID* (S_ID) and *Destination_ID* (D_ID). These addresses are normally represented as three hex-digit pairs, as in ‘0A 1B 2C’. They need to be unique within the network and are assigned dynamically and automatically with mechanisms that depend on the topology.

3.5 Sequences and exchanges

A *sequence* is a set of one or more related data frames transmitted unidirectionally from one node port to another one. Each sequence has a context, that is called *exchange*.

The exchange is the fundamental mechanism used to coordinate the interchange of information and data between two *Nx_Ports*, an “Originator” and a “Responder”. It is made up by a set of non-concurrent sequences, that flow in the same or in the opposite direction.

The Originator opens a new exchange and transmits one or more sequences. When it is done, it can close the exchange or transfer the initiative to the Responder, so that it can transmit its sequences. The interchange goes on in this half-duplex way until the exchange is terminated by either the Originator or the Responder.

A sample “session” made of an exchange with 5 sequences between two nodes is shown in figure 3.6.

A new sequence cannot be started in a certain exchange until the previous one has been terminated. A port can have many different exchanges with different ports active at the same time and transmit or receive on each of them independently. This possibility to multiplex exchanges promotes efficient link utilization.

²<http://standards.ieee.org/regauth/>

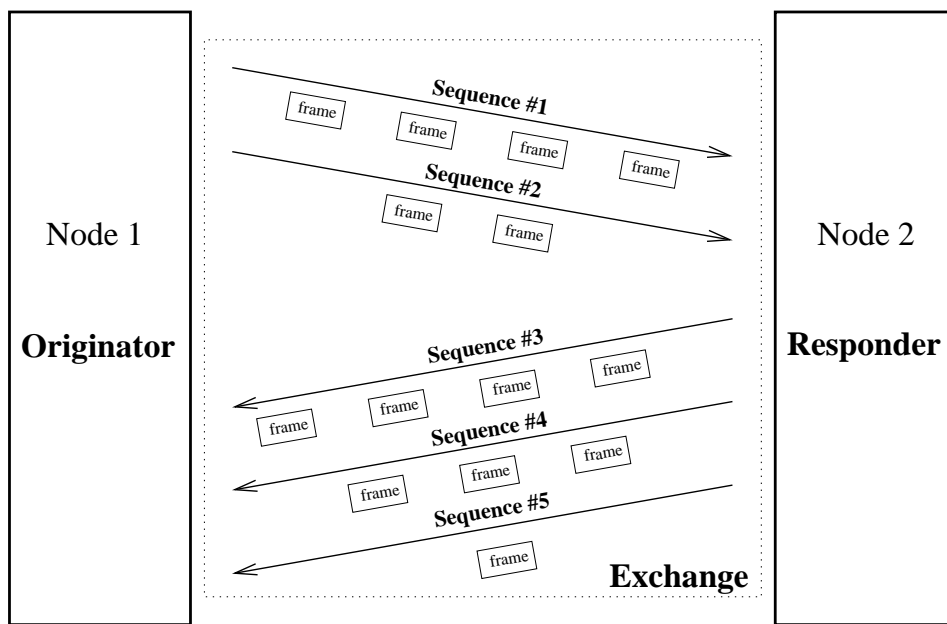


Figure 3.6. A Fibre Channel exchange

Each frame carries information about the sequence and the exchange it belongs to in its header. Exchanges are determined by a pair of IDs called “Originator ID” (OX_ID) and “Responder ID” (RX_ID). The FC-2 layer of the receiving node reassembles frames in the appropriate sequence and passes it up to the FC-4 layer. The hierarchical structuring in exchanges, sequences and frames provides high flexibility for upper layer protocol mapping.

3.6 Frame format

The format of a Fibre Channel frame is shown in figure 3.7. The maximum size of a frame is 2148 bytes and the size must be evenly divisible by 4, as the minimum unit that the transmission layer accept is a 4-byte word.

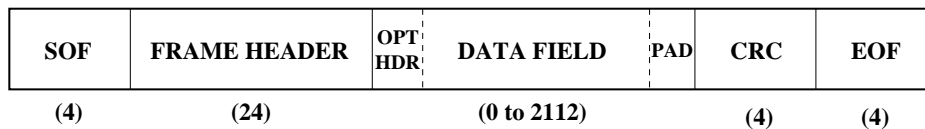


Figure 3.7. Frame format (sizes are in bytes)

SOF

Each frame starts with a delimiter, called “Start-of-Frame” (SOF). This preamble marks the beginning of the frame, serves multiple purposes and can take different forms. It declares the class of service of the transported frame and may request the allocation of resources in case of connection-oriented services. It also informs the receiver whether the frame is the first of its sequence or belongs to an already active one.

Frame Header

The next field is the frame header, which contains source and destination addresses, exchange and sequence identifiers, numbering, transported protocol and other control information.

Data Field

The data field contains the payload of the frame and is of variable length. Optional headers containing additional control fields can take up to 112 bytes of the data field. The length of the data field must be evenly divisible by 4, so up to 3 bytes of padding can be added to the actual payload.

CRC

The Cyclic Redundant Check is a 4-byte value used to verify the data integrity of the frame. It is calculated with a well-known algorithm on the frame header and the data field prior to encoding for transmission and after decoding upon reception.

EOF

The “End-Of-Frame” (EOF) delimiter marks the end of the frame. A particular value of this field is used to notify the receiver that the frame is the last of its sequence. In case of connection-oriented classes of service, specific values of the EOF indicate that the dedicated connection can be removed.

3.7 Flow control

The rate at which a device in a network can receive frames depend on the characteristics of the device itself and on the available resources. Flow control mechanisms are used to regulate the rate at which a transmitter sends frames in order to achieve efficient bandwidth utilization without overwhelming neither the network nor the receiver.

In Fibre Channel networks there are a number of flow control mechanisms and they are all based on the concept of *credit*. A credit represents the ability of a receiver to accommodate one frame. The receiver grants to the transmitter an initial number of credits, typically proportional to the size of its buffers. The transmitter is authorized to send one frame for each credit it has received; after that it has to stop until it receives more. As soon as the receiver has finished processing an incoming frame (for instance, it has passed it to upper layers) it can free the resources that were used by that frame and grant a new credit.

Fibre Channel provides two levels of flow control: “end-to-end” and “buffer-to-buffer”. The first one operates between the source and destination nodes and does not involve the network. If a node has many active sequences, it performs end-to-end flow control on each of them separately. The second one takes place between pairs of adjacent ports, such as connections between a node and a switch or between two switches. Buffer-to-buffer credits are managed on a port basis. The two levels are illustrated in figure 3.8.

Flow control operations depend on the class of service. For instance, Class 3 service is datagram-like and does not provide end-to-end flow control.

Credit-based flow control mechanisms guarantee that a device accepts incoming frames only if it has the resources to service them. Network or node congestion cannot occur and frames are never dropped³

³This is radically different from what happens in TCP/IP networks, where TCP uses frame drop as a flag for network congestion. The transmitter sends increasing amounts of data; when a frame is dropped, it

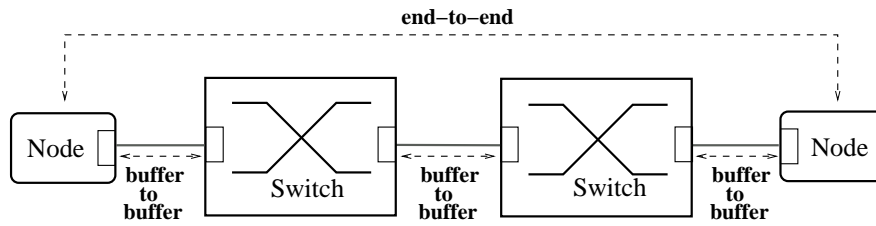


Figure 3.8. Flow control levels

As links are full-duplex, the receiver can receive frames and at the same time grant new credits to the sender, thus ensuring a continuous frame flow and maximum bandwidth utilization.

3.8 Error recovery

The standards distinguish between two fundamental levels of errors: link integrity and sequence integrity.

Link integrity errors are the result of events such as loss of signal, loss of synchronization and invalid transmission words. In this case, link-level management operations (e.g. link reset, reinitialization, etc.) are performed automatically to try to reestablish link integrity.

Sequence integrity errors include frame loss, reception of damaged frames, abnormal termination of a sequence. In this case several policies can be adopted, according to the requirements of the involved FC-4 layer. Generally it is preferred to discard the whole sequence and let the upper layers perform the recovery operations rather than trying to track single frames and request retransmission.

realizes that the network is congested and slows down.

Chapter 4

Fabric concepts and operations

The first three sections introduce concepts that are particularly relevant to switched fabrics. After that, the operations that are performed by switches to initialize the fabric and keep it working are described in detailed. The characteristics of the protocols and algorithms used during these operations are critical for the scalability and reliability of the fabric.

An attempt has been made to keep a conversational style without sacrificing correctness. The book [12] can be used for a more comprehensive and detailed treatment of the topics presented in this chapter. For implementation details the standard [11] should be referred to.

4.1 Fabric model

4.1.1 Switches

In the architecture of a switch, two logical components can be identified: the *data plane* and the *control plane*.

The data plane is focused on transferring incoming frames to outbound ports at very high rates. The switch determines the correct outbound port by looking up a part of the destination address in a table and forwards it. No other processing is done on the frame. Fibre Channel frame format, headers and addressing scheme were purposely designed to keep the data path simple and allow very fast frame switching.

The control plane, on the contrary, performs difficult operations that involve the usage of complex protocols. It is responsible for the collection and distribution of fabric information, coordination with other switches and provision of various services to nodes. Among the services provided are: routing, address assignment, directory services, access control and network management. As Fibre Channel tries to offer a reliable and predictable transport service, switches exchange a large amount of control information.

Control information exchanged by switches use a particular class of service, called Class F¹. Class F service is similar to Class 2: it is not connection oriented and require each frame to be acknowledged. Switches must support Class F frames on all E_Ports.

Most control communications between switches take place by using special information units called “Switch Fabric Internal Link Services” (SW_ILS). SW_ILSs always use Class F and follow specific rules for format and usage.

4.1.2 Nodes

Nodes are unaware of the physical and logical structure of the fabric. They see it as a cloud with the switch they are attached to as its entry point; they rely on its capabilities to forward frames to any destination and provide all the information they need.

For example, if a node needs the address of another node it knows by name, it simply sends the request to a well-known address. The attached switch is responsible to collect the information needed (possibly from other switches) and send the response back to the node.

4.2 Addressing

In a fabric, switches are responsible to assign addresses to the connected Nx_Ports. The 24 bits of the Fibre Channel address are partitioned in 3 fields of 8 bit each that have a particular semantic. The most significant byte is called the *Domain_ID*, the middle one the *Area_ID* and the least significant one the *Port_ID* (figure 4.1).

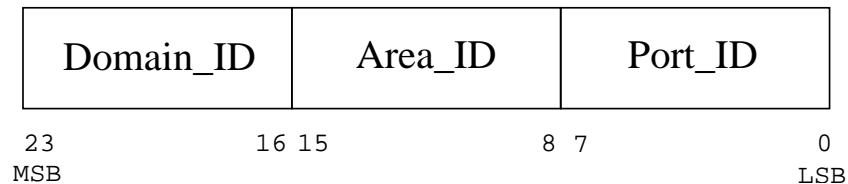


Figure 4.1. Address partitioning in a fabric

A domain is a group of one or more switches responsible for the assignment of Area_IDs and Port_IDs. All the Nx_Ports in a domain have addresses with the same Domain_ID. Valid domain values are in the range 01–EF, the other are reserved.

Even though the standard allows multiple domains for a switch and multiple switches for a domain, most implementations assume that to each switch corresponds one and

¹When it is only necessary to distinguish between control and data traffic, Classes 1–6 are collectively referred to as ‘Class N’.

only one domain. When this is the case, each switch is unambiguously identified by its Domain_ID and determining to which switch a node is attached is immediate. This assumption has relevant consequences on many protocol implementations and will be kept through the whole document.

A switch is free to assign to the attached N_Ports any Area_ID and Port_ID combination. If a switch port operates as an FL_Port, an Area_ID is reserved for that port and all the NL_Ports connected to the same loop.

Some of the reserved addresses are used to identify special entities running on a switch and are called “well-known addresses”. These entities are processes that provide services to nodes or switches (see section 4.7).

Because of this partitioning scheme, in a fabric there can theoretically be up to 239 switches with 65536 ports each.

4.3 Flooding

It is often the case that a switch needs to communicate information to all the other switches in the fabric but it does not know how to reach them, for example because routing information (or even Domain_IDs) haven’t been distributed yet. To do so, it uses a technique known as *flooding*: the originating switch sends the SW_ILS containing the information out of all its E_Ports; a switch receiving the SW_ILS responds and in turn retransmits it out of all its E_Ports where it has not yet received it. The SW_ILS will ultimately propagate to all the switches in the fabric.

It is not possible to know exactly when all the switches have received the information. When a switch responds for the first time to a SW_ILS, it starts a timer; when the timer expires, the flooding procedure is supposed to be finished.

A sample flooding procedure is shown in figure 4.2.

4.4 Port operations

When a switch detects a link coming up on one of its ports, it does not know yet what kind of device it is connected to and, as a consequence, which is the correct behavior of the port.

4.4.1 Node login operations

If the switch is connected to a N_Port, the node at the other end of the link will start a “Fabric Login” (FLOGI) procedure that consists in the exchange of information such as the node and port names, supported classes of service and available credits. When the switch receives an FLOGI, it realizes that the attached device is a node and that its port

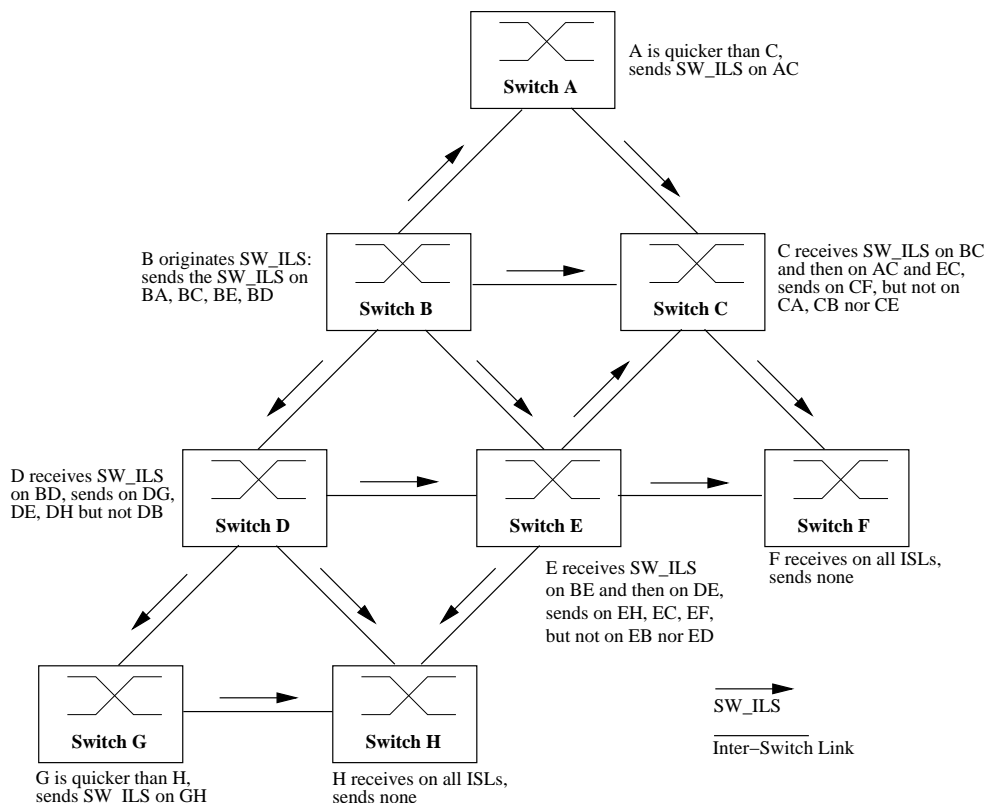


Figure 4.2. A sample fboding procedure [11].

must behave as a F_Port. The switch responds to the FLOGI, provides its own parameters and assigns an address to the N_Ports.

A node may also need to login with the N_Port of the remote node; this procedure is called “Port Login” (PLOGI). Login operations for an N_Port are summarized in figure 4.3

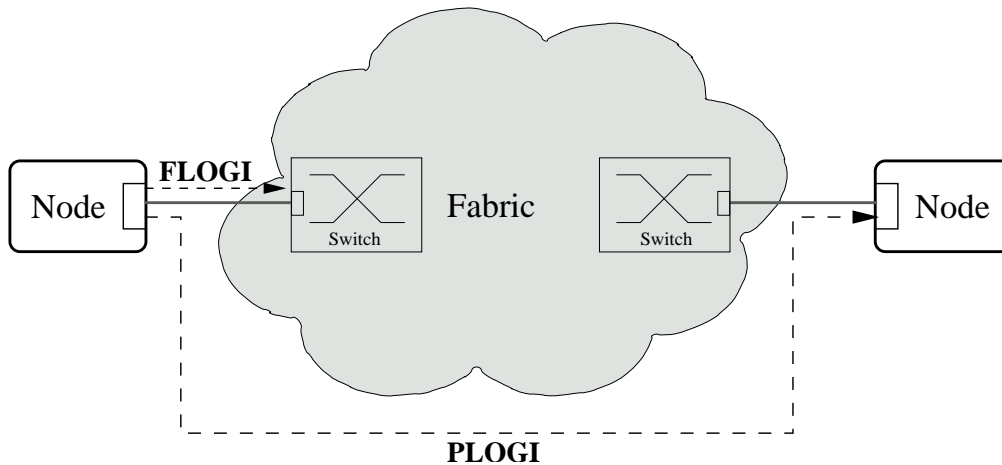


Figure 4.3. Login operations for an N_Port

As soon as the login procedures are completed, the link is considered operational and frames can flow. A similar, but more complicated, initialization procedure occurs if a loop is attached to the switch port.

4.4.2 Inter-switch link parameters exchange

If the device at the other end of the link is another switch, it will start a different initialization procedure. The switches will exchange data about their parameters and capabilities (allowed classes of service, timeout values, supported protocols, etc.) by means of an SW_ILS called “Exchange Link Parameters” (ELP) and an optional one called “Exchange Switch Capabilities” (ESC). If the parameters are mutually accepted, switches proceed with further operations (described in the next sections), otherwise the link is isolated.

4.5 Domain management

In a fabric each switch must have a Domain_ID in order to assign addresses to the connected Nx_Ports. To guarantee the uniqueness of Domain_IDs, they are managed by a single switch, known as the *Principal Switch*. The Principal Switch is chosen among the switches present in the fabric with a process called “Principal Switch Selection”.

It is essential that in a configured and working fabric all the switches have information about the identity of the Principal Switch, the links that can be used to reach it and the domains that have already been assigned. When the topology of the fabric is modified by the addition or removal of an ISL, such information might be invalidated. This may require a fabric reconfiguration process, that includes the reselection of the Principal Switch and redistribution of Domain_IDs. Whenever possible, fabric reconfiguration is performed in a non-disruptive manner; in this case switches can continue to forward data frames and are guaranteed that, at the end of the process, they will retain their current Domain_ID, even if the identity of the Principal Switch changes. In other cases this is not possible and fabric operations must be interrupted during the reconfiguration process.

During fabric reconfiguration, switches are identified by a pair of values: the first is called *Priority* and is administratively set², the other is the 64-bit switch WWN.

4.5.1 Fabric parameters exchange

After link parameters have been negotiated on an ISL, the two switches exchange the information they have about the fabric with a SW_ILS called *EFP*, (“Exchange Fabric Parameters”). Information contained in an EFP includes the identity of the Principal Switch and the list of assigned domains. At this point, four cases are possible:

1. One of the two switches has an empty Domain_ID list, the other has a non-empty one. This means that an unconfigured switch is joining an already configured fabric. The unconfigured switch learns from the configured one who is the Principal Switch and which domains have been assigned. The new-coming switch can proceed and request a Domain_ID to the Principal Switch. There is no need for fabric reconfiguration.
2. Both switches have non-empty Domain_ID lists and the lists do not overlap. In this case two previously disjointed fabrics are connected by the new ISL.

Both switches know about the Domain_IDs that have been previously assigned in their respective fabrics. As there is no overlapping (no Domain_ID previously assigned to a switch in a fabric has also been assigned to a switch in the other fabric) the two fabrics can be joined without disruption. However, the new fabric must have a single Principal Switch and a non-disruptive fabric reconfiguration must be started by issuing a SW_ILS called *BF* (“Build Fabric”).

3. Both switches have non-empty Domain_ID lists and the lists overlap. In this case the new link connects two already configured fabrics, but these fabric cannot be joined because Domain_IDs would not be unique. Fabric reconfiguration must be

²It can be used to favor a switch in the selection process. A switch with a lower priority has more chances to become the Principal Switch.

disruptive as each switch cannot be sure that at the end of the process it will retain its previous Domain_ID. Disruptive reconfiguration is triggered by a SW_ILS called *RCF* (“Reconfigure Fabric”). A disruptive reconfiguration interrupts fabric operations for a significant time, therefore it is not started automatically by the switch. The switch simply isolates the ISL; if the fabric administrator wants to join the fabrics, he or she will have to manually issue the RCF.

4. Both switches have empty Domain_ID lists. In this case both switches are unconfigured and fabric reconfiguration is implicitly happening (see below).

4.5.2 Principal Switch selection

When a switch wants to start a fabric reconfiguration, it floods a BF or RCF. Every switch in the fabric clears its assigned Domain_ID list and resets the name and priority of the current Principal Switch to its own values.

When the BF or RCF flooding timer expires, switches flood EFPs telling other switches who they believe to be the Principal Switch. If a switch receives an EFP containing a priority/name pair that is lower than the one it currently knows, it discards the old value and retains the new one; if the received value is higher, the switch keeps its current value and replies by sending it back. Each time a switch learns about a lower priority/name value, it floods the corresponding EFP and remembers on which link it was received; that link is marked as “Principal Upstream ISL” (or simply “Upstream ISL”) and is used during Domain_IDs distribution.

When a certain period of time is elapsed without lower values for the priority/name pair being received, the switch identified by that pair is the Principal Switch.

At the end of the procedure, all the switches in the fabric know who the Principal Switch is and know a link (the Principal Upstream ISL) that can be used to reach it. Switches and Upstream ISLs constitute a tree whose root is the Principal Switch. A possible result of Principal Switch Selection process is shown in figure 4.4.

If a Principal ISL goes down, a non-disruptive fabric reconfiguration is always triggered. All the switches must be sure that they have a way to reach the Principal Switch. The removal of the ISL may also cause the partition of the fabric in two isolated parts. In this case, each new fabric must reselect its Principal Switch and rebuild the Upstream ISL tree.

4.5.3 Domain_IDs distribution

After the Principal Switch and Upstream ISLs have been identified, Domain_IDs distribution can start.

Domain assignment proceeds according to these rules:

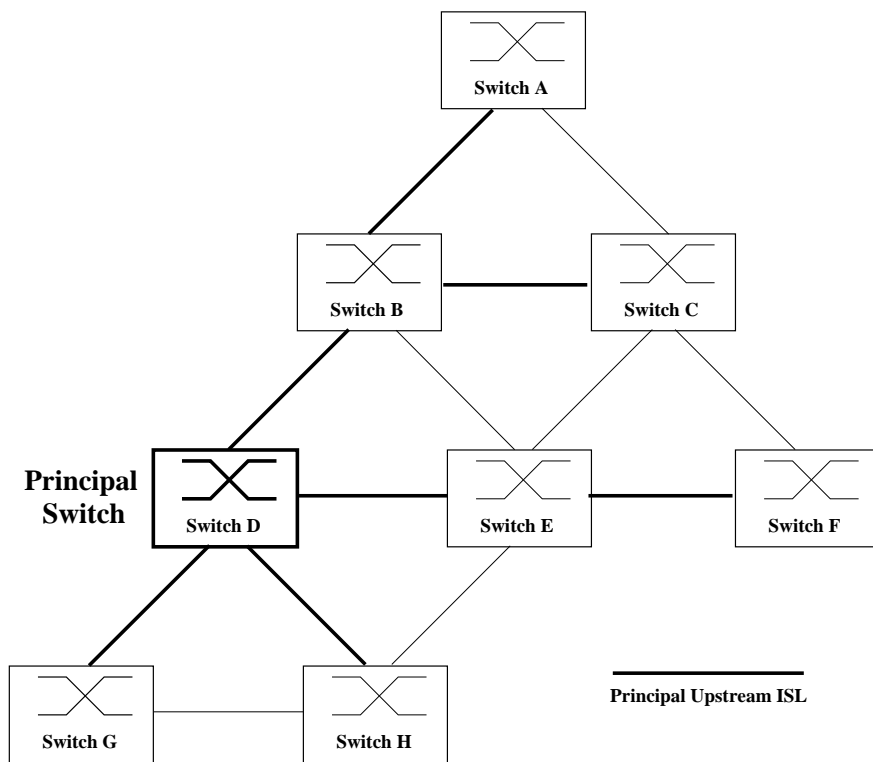


Figure 4.4. Upstream links to the principal switch

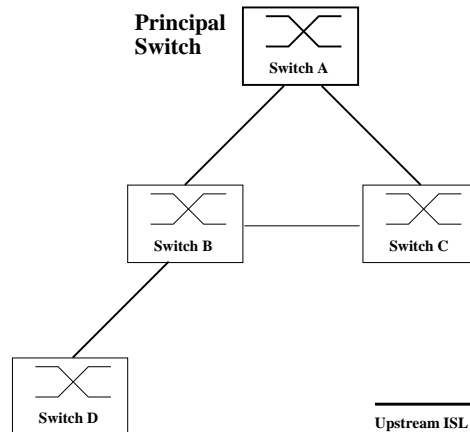


Figure 4.5. Sample Domain_IDs distribution session

- A switch that has been assigned a Domain_ID sends to its neighbors a “Domain_ID Assigned” (DIA) SW_ILS.
- A switch cannot request a Domain_ID until it has received a DIA on its Upstream ISL; when this happens, it proceeds by sending a “Request Domain_ID” (RDI) on the Upstream ISL.
- When the Principal Switch receives an RDI, it allocates a Domain_ID and communicates it in the response to the RDI.
- When a non-Principal switch receives an RDI, it relays it, that is to say, it retransmits it on its Upstream ISL. By reiterating this process, the RDI eventually reaches the Principal Switch. The response from the Principal Switch goes down the tree hop-by-hop in the same way.
- Every time the Principal Switch grants a new Domain_ID, it floods an EFP containing an updated list of assigned domains.

Consider, for example, the situation depicted in figure 4.5. Domain_IDs distribution steps, assuming all the switches were unconfigured, are the following:

1. Switch_A realizes that it has been elected Principal Switch; it grants himself a Domain_ID, send a DIA to Switch_B and Switch_C, floods out an EFP containing the newly assigned Domain_ID.
2. Switch_B and Switch_C both receive the DIA from Switch_A on their Upstream ISL. Both send an RDI to Switch_A and wait for an answer.

3. Switch_A receives the requests, assigns the Domain_IDs, responds to Switch_B and Switch_C, issues (two times) an updated EFP.
4. When Switch_B receives its Domain_ID, it sends a DIA to Switch_D.
5. Switch_D sees a DIA on its Upstream Link and sends an RDI.
6. Switch_B receives the RDI from Switch_D but, as Switch_B is not the Principal Switch, it forwards it on the Upstream ISL to Switch_A.
7. Switch_A allocates the Domain_ID for Switch_D, sends the response to Switch_B, issues a new EFP.
8. Switch_B sees the response from Switch_A, it remembers that the response is for Switch_D and retransmits it.
9. Switch_D finally gets its Domain_ID; Domain_ID distribution is complete.

As soon as a switch gets a Domain_ID, it can assign addresses to the attached nodes.

4.6 Routing

Routing is the operation performed by switches to select paths to all reachable destinations in the fabric. Switches exchange information about the fabric topology with other switches and perform calculations to choose the best paths. The selected paths are stored in a “routing table” that the switches use to determine on which port a received frame must be forwarded. As the switches have a consistent view of the fabric, at each hop the frame gets closer to its destination and ultimately reaches it.

The protocol used in a Fibre Channel fabric to distribute routing information is called “Fabric Shortest Path First” (FSPF). In the following subsections, the various components and operations of FSPF are explained.

4.6.1 Topology database

FSPF is a link-state routing protocol, which means that it is based on the following principles:

1. Each switch tells *all* the other switches the connections it has *to its neighbors*.
2. By collecting information from other switches, each switch builds a map of the network.
3. Each switch *independently* performs its calculations on the network map to determine the best paths and updates its routing table.

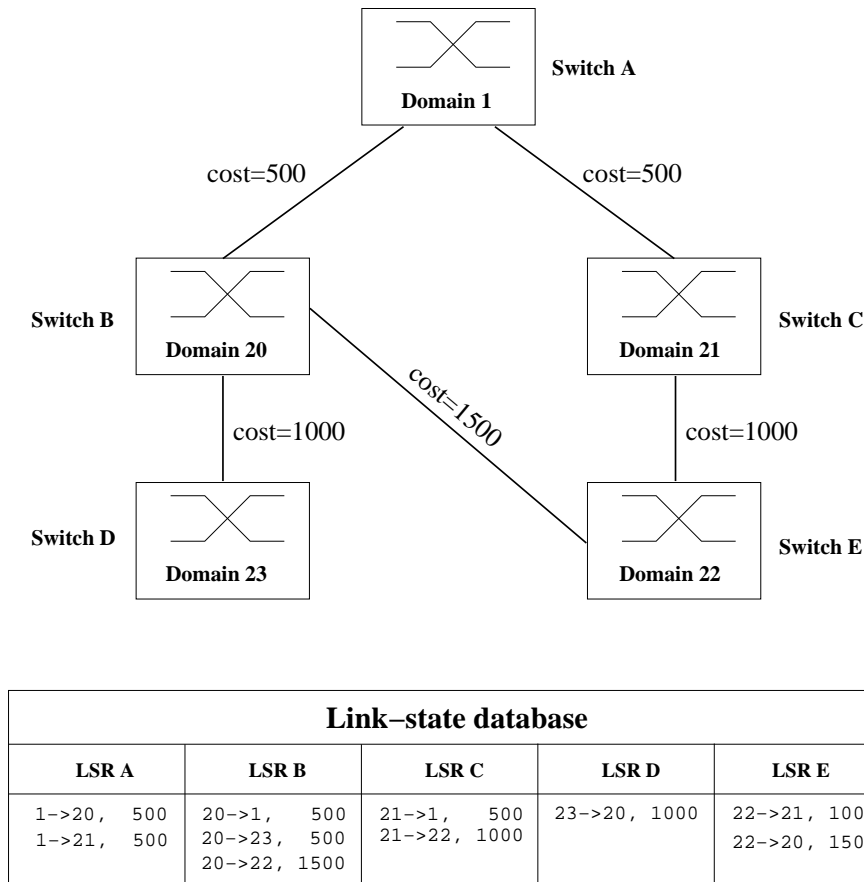


Figure 4.6. Sample topology and corresponding link-state database

The map of the network introduced above is called *topology database* or *link-state database*.

The link-state database contains a data structure called “Link-state Record” (LSR) for each switch in the fabric. The LSR describes the connectivity of the switch by listing all its links to neighbors. Link information is contained in a data structure called “Link-state Descriptor” (LSD) that include the Domain_ID of the switch at the other end of the link and the *cost* of the link. The cost is an administratively-set value that reflects the desirability of using a certain link; the lower the cost, the more convenient the link³. Each LSR also carry a *Incarnation Number* and an *Age* field. These values are used to determine which instance of a given LSR is more recent.

A sample topology and the corresponding link-state database are shown in figure 4.6 (Age and Incarnation Number have been omitted for clarity).

³the default cost of a link is inversely proportional to its bandwidth: for a 1 Gbit/s the default cost is 1000, for a 2 Gbit/s it is 500 and so on.

During normal fabric operations, the topology database is identical for all the switches. Whenever the topology of the fabric changes, update information must be distributed. It is important that all the switches receive information about topology changes so that they can dynamically adapt to them. For example, if a link goes down a path to a certain destination might be invalidated but an alternative one might exist.

When the database must be sent to another switch (completely or in part), the LSRs are packed in one or more SW_ILS called “Link State Update” (LSU). Received LSR must be acknowledged with an SW_ILS called “Link State Acknowledge” (LSA).

4.6.2 Hello protocol

The Hello protocol is a component of FSPF used to establish the identity of a neighbor switch, to exchange basic FSPF parameters and monitor the status of ISLs. When a new ISL comes up (after the address distribution phase), the connected switches transmit “Hello packets” that carry the Domain_ID of the sender and other FSPF parameters, such as timeout values. After the parameters have been mutually accepted, the exchange of Hello packets continues at regular intervals and is used as a keepalive mechanism. If a switch doesn’t receive Hello packets for a certain period of time, it assumes that the ISL is no longer operational (the link might be broken or the neighbor switch might be experiencing a malfunction).

4.6.3 Initial database exchange

After the first successful exchange of Hello packets, the switches connected by the new ISL must exchange their topology databases. The databases might be identical (if both switches were already part of the same fabric) or completely different (if they were part of two previously disjointed fabrics).

Both databases are packed in a number of LSUs and transmitted to the peer. Each LSU received must be acknowledged with an LSA.

When the exchange is complete (both databases have been transmitted and acknowledged), the two switches that previously were only *neighbors*, become *adjacent*. The new ISL can now be used to carry Class N frames and its existence can be notified to other switches by issuing LSRs containing the new adjacency.

The next step is to determine the paths to all the possible destinations in the fabric and update the routing table.

4.6.4 Path selection

The link-state database actually represents a graph, with the switches as vertices and the links as arcs. When the database is complete, the switch runs an algorithm (typically

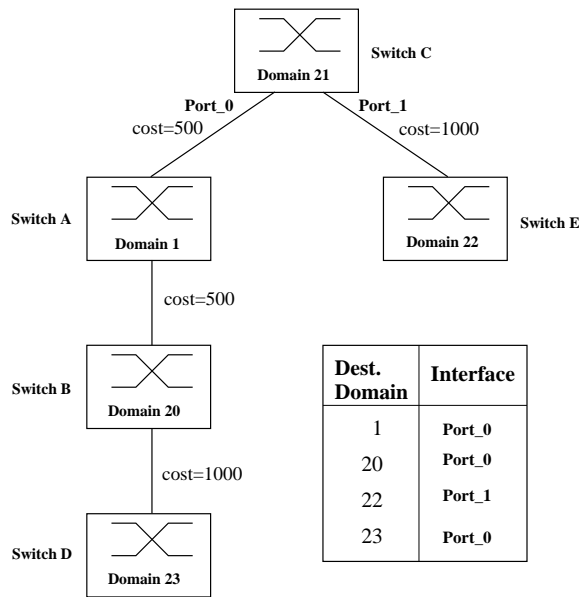


Figure 4.7. Minimum spanning tree and routing table for Switch_C (see figure 4.6)

Dijkstra’s *Shortest Path First* (SPF) [13] or a derived one [14]) that determines the shortest path (the one whose sum of all link costs is minimum) to every switch in the fabric.

The result is always a spanning tree with the switch as the root and the links belonging to minimum paths as arcs. The minimum spanning tree and the corresponding routing table for the previous example, calculated by Switch_C, is shown in figure 4.7.

It is important to note that each switch performs calculations on its own and the results are not distributed to other switches. The structure and management process of the routing table is not specified by the standards as it is strictly implementation-dependent. The routing table consists as a minimum of a list of assigned domains and the E_Port used to forward frames toward that domain.

4.6.5 Database update

When a switch detects a change in the status of its links, it rebuilds its LSR and floods it to all other switches. A switch is authorized to originate only LSRs that describe its connectivity.

As the flooding procedure takes a certain period of time and topology changes can be quite rapid, older and newer instances of the same LSR can be present in the fabric at the same time. Each switch considers only instances of a certain LSR that are newer than the one it owns. To determine if an LSR is newer, the Incarnation number and the Age field are used.

Every time the topology database of a switch is modified, the routes must be recalculated and the routing table updated. When all the switches have received the most recent information about the topology and have recalculated the routes, it is said that “the network has converged”. Before convergence has occurred, there might be a period of time during which some switches do not have the latest available information and make inconsistent decision about frame routing. This can cause the formation of routing loops in the fabric, with frames being passed from one switch to the other without ever reaching their final destination⁴. It is therefore very important that new LSR are redistributed immediately and route recalculation is performed as quickly as possible.

4.7 Distributed Services

Fibre Channel standards define a number of additional services that the network is supposed to provide to nodes. These services are mostly needed in a fabric environment, but are theoretically topology-independent. The providers of these services are abstract entities that behave as nodes, each with a well-known port address. The actual implementation of the services providers is not known to the nodes that simply access them by sending requests to the appropriate addresses.

Due to performance and reliability reasons, the standard choice is to implement these servers directly in the switches. As these services are topology-wide, the implementation is distributed, meaning that each switch runs its own instance of the servers and coordinates with the others. A switch that receives a request considers whether the information needed to satisfy it is locally available; if that is the case, it handles it directly, otherwise it contacts other switches, collect the requested information and responds to the originating host. Information can be locally cached because state-change notification mechanisms are available.

Below is a description of the most common services used in a fabric; complete information can be found in [15].

4.7.1 Directory services

Node ports need directory services in order to discover other nodes and their capabilities. Each switch must implement a “distributed Name Server” (dNS), a server that is responsible for the collection of information regarding the ports connected to the switch and its distribution to the fabric. When a node port is connected to a switch, the switch stores in a database information such as the node and port name, port address, supported classes of service and available FC-4 protocols. Information can be explicitly registered by the attached device or simply added by the switch.

⁴Fibre Channel does not provide for a time-to-live as IP does.

The DNS running on a switch is responsible for the name entries belonging to the domain assigned to the switch.

4.7.2 Management services and zoning

Management services provide mechanisms to control and monitor fabric operations.

Fabric Configuration Server

The Fabric Configuration Server provide to management appliances mechanisms to discover fabric topology and attributes. This server is typically implemented by interconnect devices such as switch and hubs, that provide information about the number and type of ports, physical media, port status, entries in the name database, etc.

Zone Server

Zoning is a mechanism to provide access control in a fabric. It can be used for various purposes, including security, confidentiality and data protection.

A *zone* is a set of ports, called *zone members*, that are allowed to communicate among them. A port cannot access another port unless there is a zone of which both are members. Ports can be members of more than one zone. Zone membership can be based on port names, node names, port addresses or other criteria.

Zoning is especially useful in environments where multiple operating systems and multiple file-systems are used. Accidental access to a storage device using an improper file-system might irremediably damage the data. Another situation where zoning is fundamental is when “storage service providers” build a SAN providing large amount of storage space and lease it to multiple customers.

Zoning information must be administratively set and distributed to all the switches in the fabric. These are the tasks performed by the Fabric Zone Server: it provides an interface for the definition of zones and zone members and distribute this information to newly adjoined switches. When a new ISL comes up, the switches exchange zoning information to see wether they agree on the set of rules that must be enforced; in case of conflict, the ISL is isolated.

Chapter 5

The Large Network Emulator

This chapter describes the “Large Network Emulator” (LNE) a tool that has been developed to study scalability issues in a fabric and test the control plane of a switch.

5.1 Fabric scalability

In the previous chapter a distinction between the data plane and the control plane of a switch was made. That distinction is important to analyze the scalability issues of a fabric.

The flow control mechanisms used by Fibre Channel (section 3.7) ensure that the network does not allow incoming traffic to enter at a rate higher than the maximum it can sustain. This ensure that no frames are dropped due to congestion and that high offered traffic does not cause instability in the network. If the network is not adequately dimensioned, the nodes might not get the bandwidth they need but frame forwarding will not be interrupted. If a switch is not able to provide enough aggregate bandwidth, the devices attached to it will be penalized but the operation of other switches won't be affected.

From these statements we can deduce that data plane of a switch does not need special precautions to operate in large fabrics. Its performance is not dependent on the size of the fabric and does not impact other switches.

The control plane, on the contrary, is sensible to the size of the fabric in which it is placed. Switches exchange information to coordinate their activities and offer services to nodes. As the size of the fabric grows, the amount of information that must be exchanged and processed gets larger.

Besides, the inefficiencies of a single switch can affect the whole fabric. Route recalculation, for example, is a CPU-intensive task: if a switch does not perform it quickly, it delays fabric convergence; if it interrupts Hello packet transmission due to lack of resources, it might break adjacencies, causing more instability in the fabric.

The protocol implementations and the algorithms used in the control plane are therefore the most critical components for fabric scalability and must be adequately tested.

The testing of the control plane is generally quite difficult. The protocols used are complex and rich in features and include Finite State Machines (FSMs) with many states and transitions. Certain tasks require interaction with other switches or specific conditions to be performed. For example, routing operations cannot start until domain assignment has been completed in the whole fabric.

The most straightforward way to observe the behavior of a switch in a large fabric is to implement the fabric itself. Building a fabric containing hundreds of switches is impractical: Fibre Channel switches are costly, cumbersome and difficult to manage, so alternative methods to test the control plane must be found.

5.2 Fabric emulation

An approach that can be used to simplify the testing of the control plane is the *emulation* of a fabric.

A switch collects information about the fabric from other switches: if we connect it to a device capable of interacting with it in the same way that the switches of a particular fabric would, the switch will believe that it is actually connected to that fabric.

The “Large Network Emulator” (LNE) is a tool for fabric emulation. It can be connected to real switches through multiple connections and provides a management interface to program the topology of the emulated fabric and dynamically inject events.

The user describes the topology of the fabric to emulate, specifying the switches that compose it and how they are connected, then the emulation process starts. The LNE transmits on its interfaces the same frames that the fabric specified by the user would.

A sample deployment scenario is shown in figure 5.1. The boxes inside the dashed cloud represent *virtual* switches emulated by the LNE, those on the right are *real* switches under test.

The devices connected to the LNE do not realize that they are connected to an emulation system and react as if they were actually part of the emulated fabric. This way it is possible to observe the behavior of a switch in very large and rapidly-changing fabrics without physically implementing them.

The LNE behaves as a real fabric, so it exchanges link parameters, participate in Principal Switch Selection, provides topological information to other switches and so on. All the components of the control plane can be stressed with properly designed fabrics and sequences of events.

In cases where performance or flexibility reasons suggest it, multiple LNEs can be connected to the same fabric.

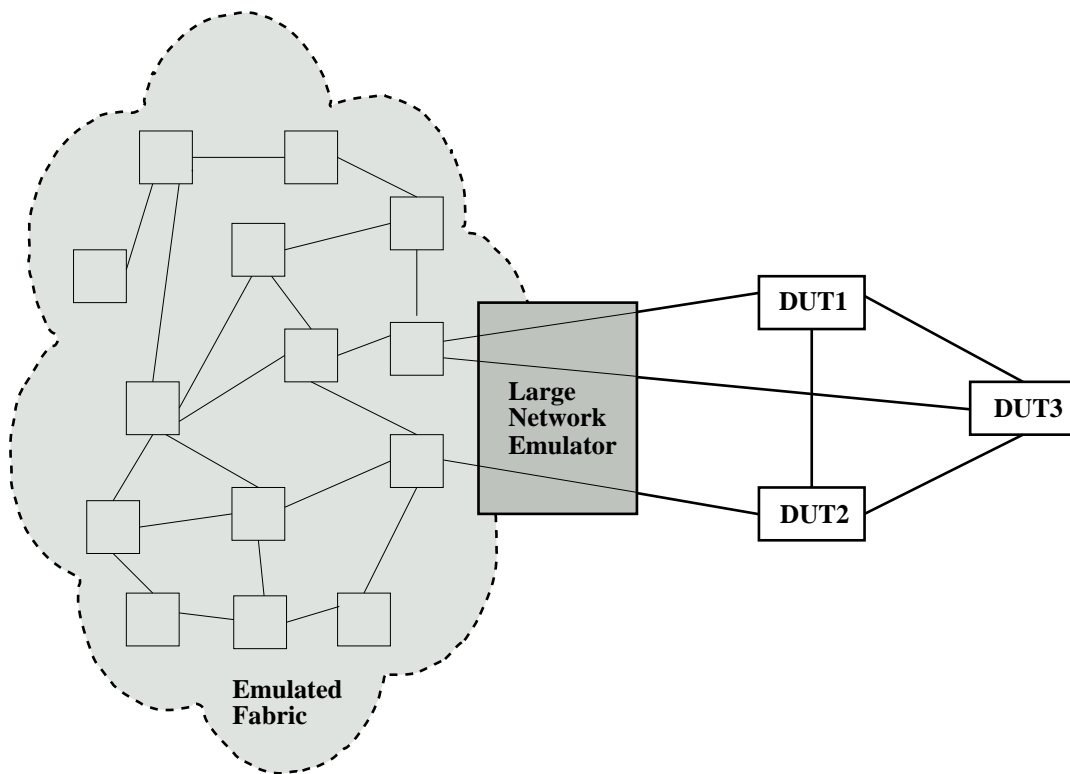


Figure 5.1. Deployment scenario for LNE

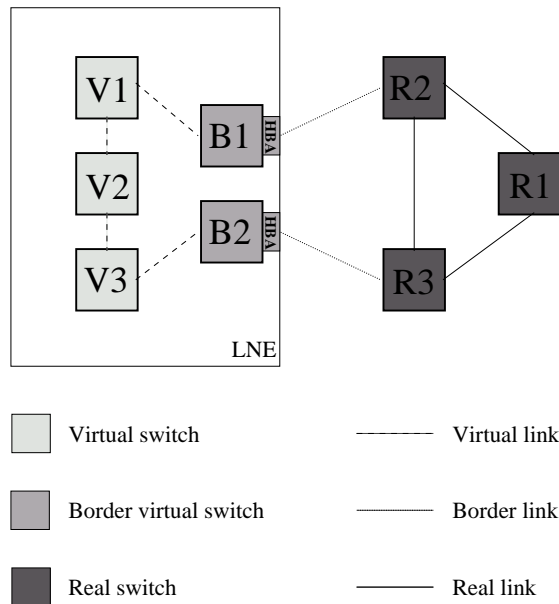


Figure 5.2. Switches and links in an emulated environment

5.2.1 Nomenclature

In a fabric composed by an emulated part and a real one, three types of switches and links can be identified: A *real switch* is a physical device external to the LNE; a *virtual switch* is a switch that is being emulated by the LNE, but it is called a *border virtual switch* (or simply *border switch*) in the case it has one or more connections to the real fabric, that is to say, if any of its ports is mapped to a HBA. A link is a *real link* if it connects two real switches, a *virtual link* if it connects two virtual switches and a *border link* if it connects a border virtual switch to a real one. These concepts are clarified in figure 5.2.

5.3 LNE architecture

5.3.1 Design considerations

The task of emulating a fabric can be approached in two different ways: switch-oriented and fabric-oriented. In the first case, the program is centered around a model of a single switch. By running multiple instances of this object and providing methods for communication among them, a fabric can be emulated. In the other case, the focus is on the fabric, not on the individual switch. The emulation tasks are divided among modules that have different duties but must all be able to act on behalf of the whole fabric. For example, rather than implementing FSPF for a single switch and running a separate instance for all

the switches in the fabric, a single FSPF process generates FSPF frames for the whole fabric on the appropriate HBAs.

The switch-oriented approach is easier to follow at first, because the emulated switch exactly replicates the actions of a real switch and no additional issue arise during the implementation of the protocols. However, it also implies that all the tasks performed by a switch, even those that are not relevant to the fabric emulation, must be implemented. When a link between two virtual switches comes up, for instance, the negotiation of link parameters must be performed, even if it produces no frame flow toward the real switches.

The fabric-oriented approach, on the contrary, requires a greater initial effort to endow each process with the intelligence needed to take into account the whole fabric. This means that the every component, beside implementing a protocol, must be able to foresee which are the results of running that protocol in the fabric. The advantage of this approach is that the component are not forced to replicate all the operations performed by real switches: only the features that are essential for the emulation task must be implemented. They can lack several capabilities, as long as the real devices cannot distinguish the emulated fabric from a real one. By reducing to a minimum the operations, the fabric-oriented approach achieves lower resources utilization. During the design phase of the LNE, this approach has been chosen.

5.3.2 Architectural layers

As in a fabric the control traffic is due to many services and protocols, the architecture of the LNE has been designed to be modular. The LNE is composed of a basic infrastructure and a number of plug-in modules that implement protocols or perform specific tasks. Traffic generation capabilities can be expanded by writing new modules and plugging them in the basic infrastructure.

The functional components of the LNE are organized in a layered framework, shown in figure 5.3.

Layer 1 includes both hardware and software components. Its purpose is to provide to upper layers a hardware-independent interface to send/receive frames. It basically include HBAs, their driver and a software layer to multiplex/demultiplex frames.

Layer 2 provides the basic infrastructure needed to operate. It includes a component for sequence/exchange management, a database of the emulated topology and a port manager for the HBAs.

Layer3 contains the traffic generation components. Each component is responsible for a certain protocol or set of tasks. As a minimum, domain distribution and routing protocols must be active to be able to join a fabric. At this layer is also present a command dispatcher, that receives commands from Layer 4 components and distributes them to the correct component.

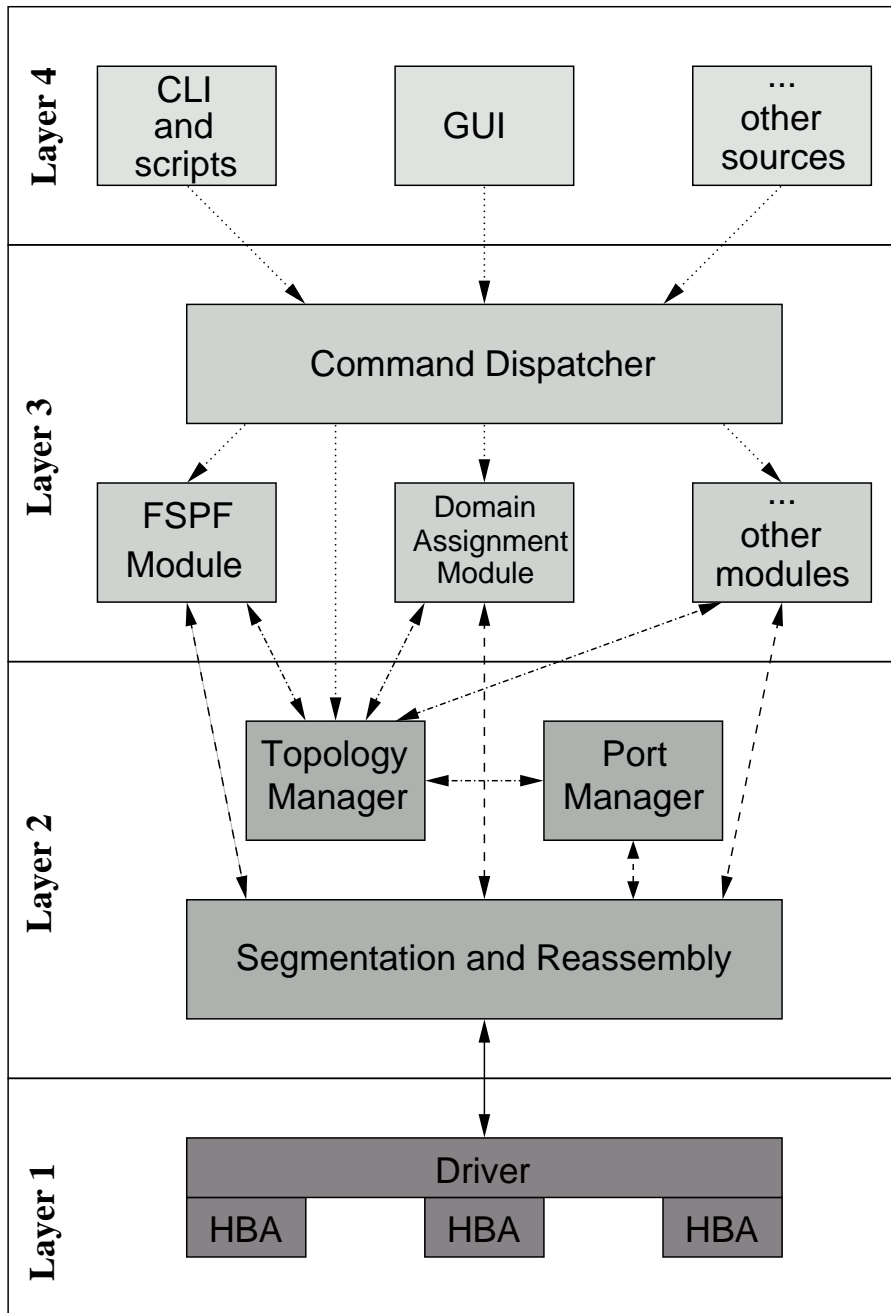


Figure 5.3. Large Network Emulator architecture

Layer 4 includes all the facilities that can be used to inject commands to the LNE. Among these are, a Command-Line Interface (CLI), scripting tools and Graphical User Interfaces (GUIs).

Each of the modules employ one or more threads to provide an adequate level of concurrency. Threads, available on most modern operating systems, allow efficient utilization of the computing resources and eliminate the need for Inter-Process Communication (IPC) mechanisms. A multi-threaded program can seamlessly take advantage of multi-processor platforms because the operating system is able to schedule different threads on different processors.

5.3.3 Hardware and software platform

The LNE has been implemented using a standard PC with commercially-available HBAs. Linux has been chosen as the operating system both for development and field usage. Up to eight HBAs are supported.

5.4 Component description

5.4.1 HBAs and drivers

Most HBAs provide some hardware support for SCSI operations and their driver present them to the operating system as a block-oriented device. As the LNE generates control traffic, a modified version of the HBA driver that allows the transmission and reception of raw frames has been used.

5.4.2 Segmentation and reassembly unit

This unit basically provides some of the services of the FC-2 layer, including Sequence and Exchange management. Modules use its services to transfer information units and register with it to receive them. A module can filter the information units it wish to receive by specifying source address, destination address and protocol type. In case of transmission or reception error, the module is notified and it can choose how to proceed.

5.4.3 Topology Manager

The Topology Manager (TM) is the central repository for the information about the topology that are relevant to all modules. The topology is described in terms of switches and links, with associated status information. If a module deals with data that might be needed by other modules (such as the Domain_ID of a module) it stores them in the TM; if they are specific to that module (for example, the value of a timer) it stores them internally.

The topology can change because of one of these events:

- the user issues a command that directly affects the topology (for example the addition of a new link between virtual switches)
- a module notifies a protocol event, such as the assignment of a Domain_ID to a virtual switch
- one of the links that connect the LNE to the real switches change status (for example, it is physically connected or disconnected or loses synchronization)

As events are drive by topology changes, every time such an event occurs, the TM broadcasts a notification message to all the modules. Some of them will be affected, the others simply ignore it.

5.4.4 Port Manager

The Port Manager (PM) is responsible for the initialization and operation of the HBAs. A HBA becomes an entry point of the virtual fabric when the user specifies a mapping between a port of a virtual switch and the HBA itself.

When the TM receives a request to activate a link that connects a virtual switch to a real one, it requests the PM to perform link initialization and the appropriate E.Port operations. The PM negotiates link parameters and switch capabilities with its peer by exchanging ELP and ESC and reports the results to the TM. If the negotiation is terminated successfully, the TM considers the link up and commits the change to its database; if it fails, the stored topology is not changed and the user is informed of the failure.

The Port Manager is also responsible for link monitoring: if a link connected to a HBA goes down, the PM informs the TM.

5.4.5 Domain Assignment Module

The main responsibilities of this module are participating in Principal Switch Selection, performing EFP exchange when a border link is coming up and requesting Domain_IDs for virtual switches. The module keeps track of the Domain_IDs assigned to the fabric, the identity of the Principal Switch and the Upstream ISLs.

When the module participates in Principal Switch Selection, it uses a value for the Priority that declares its inability to become the Principal Switch. The features needed to operate as the Principal Switch were not implemented because being the Principal Switch doesn't seem to be relevant to study scalability issues.

When the TM notifies the module that a new border-link has successfully terminated PM operations, EFPs are transmitted and received to see wether a fabric reconfiguration must be started. When the virtual fabric has successfully joined the real one and

the Principal Switch has been identified, the module requests addresses for the virtual switches.

When Domain_IDs are obtained, routing operations can start, so the module notifies the TM that, in turn, notifies the FSPF Module.

5.4.6 FSPF Module

The FSPF module is a full implementation of all the components of the routing protocol. It establishes adjacencies with real switches, distribute connectivity information about the virtual fabric and maintain a link-state databases.

Each border switch must transmit and receive Hello packets on its border links and performs initial database exchange if a new border link comes up. When a topology change occurs inside the emulated fabric, the FSPF module is notified by the TM; it determines the appropriate LSU and transmits it out of the proper HBAs. If the topology change occurs outside the virtual fabric, the LNE is informed by a real switch by means of an LSU and floods it appropriately to the other real switches. In both cases, the link-state database is updated to reflect the new status of the network.

All the operations described so far take place in a point-to-point fashion: switches use flooding to distribute information to other switches. For this reason, a routing table is not required and the FSPF module doesn't need to calculate shortest paths. Some distributed services, however, rely on routing functionalities. If the support for such services is to be added, the capability to populate a routing table must be provided.

5.4.7 Command Dispatcher

The Command Dispatcher (CD) receives user commands from external control facilities and directs them to the appropriate module. Each module registers the commands it want to receive by specifying string templates containing keywords and parameters placeholders.

When the LNE is started, the CD listens on a TCP socket. External controllers open a connection to the CD and inject commands. The CD parses the received string, matches it with module-registered templates and dispatches it to the right module. Any output that the command produces is sent back on the TCP connection.

5.4.8 Management facilities

The Command Dispatcher expects commands to be sent through a TCP connection, so it is not aware of what kind of tool is issuing those commands. The facilities that are more adequate for the LNE are:

- a Command-Line Interface (CLI) for interactive usage. The user injects commands one at a time and immediately sees the response of the LNE and its consequences.
- A scripting tool, to perform batch operations. This is especially useful to run unproctored tests or to rapidly inject series of events. An example is a script that, after building a base topology, injects at regular intervals up/down events on randomly chosen links.
- A Graphical User Interface (GUI). This is the most complex and flexible type of interface, but it requires a considerable development effort to be put in place. The GUI allows an instant view of the emulated fabric and its parameters, and provides the ability to deploy the fabric and vary its parameters in an intuitive way, without the need for paper schemes or other drawings. It must be noted however, that a fabric is useful where specific custom topologies are to be studied. The most common topologies, such as fully meshed networks, rings, stars, etc. can be easily produced by scripts accepting few input parameters such as the number of nodes.

Any of the three tools can be used to control multiple LNEs at the same time and coordinate their activities.

5.5 Usage and command set

The usage of the LNE comprise two phases: first the topology is described in the *configuration phase*, then in the *run phase* the emulation task is started and the status of the links and other parameters can be modified. The two phases provide two different command sets described below.

Commands are composed by keywords and parameters. Words included between < > are parameters to be chosen by the user. Some parameters allow only a predetermined set of values; in this case the valid choices are specified between { } and separated by |. Keywords or parameters between [] are optional.

Switches have names represented by alphanumeric strings, port numbers are positive values in the range 0–512.

5.5.1 Configuration phase

```
switch-add <switch>
```

This command adds a switch with a given name. Node and port names of the switch will be determined automatically (see section 3.4).

```
link-add <switch_1> <port_1> <switch_2> <port_2> [cost <cost>]
```

Connects a link between `port_1` of `switch_1` and `port_2` of `switch_2`. If the cost of the link is not specified, it is assumed to be 1000 (see section 4.6.1).

```
hba-add <switch> <port> <hba_num> [cost <cost>]
```

Associates HBA `hba_num` to `port` of `switch`. The switch becomes a border switch and the HBA is now a connection between the real and the emulated part of the fabric. The cost of the border link can be optionally specified.

```
run
```

Terminates the configuration phase and proceeds to the run phase. When this command has been issued, the modules are started and the emulation task begins.

5.5.2 Run phase

```
link-up <switch> <port>  
link-down <switch> <port>
```

Changes the status of the link connected to `port` of `switch`. If the link is a border link, the physical link is brought down.

```
switch-up <switch>  
switch-down <switch>
```

Brings up or down all the links connected to `switch`.

```
fabric-build <switch>
```

Starts a non-disruptive fabric reconfiguration from `switch`, issue a BF on appropriate ports (see section 4.5).

```
fabric-reconfigure <switch>
```

Starts a disruptive fabric reconfiguration from `switch`, issue a RCF on appropriate ports (see section 4.5).

Chapter 6

Conclusions

Protocols used by switches to exchange information about the fabric and their software implementation have been identified as critical components for the scalability of a SAN.

We have developed a tool capable of testing and stressing the control plane of a switch by emulating large fabrics with complex topologies. Such a tool can be used to:

- verify the correct dimensioning of the computational resources of a switch
- identify bottlenecks in the implementation of fabric components and areas for optimization
- verify the capability of a switch to properly handle critical situations
- compare alternative strategies and choices for protocol implementations

The Large Network Emulator has proved to be a useful and effective tool, so it is likely that in the future its development will continue with additional traffic generation capabilities and more flexible and intuitive user interfaces.

Appendix A

LNE emulation session

In this appendix we show a sample session with some switches emulated by the LNE and a real one. First the topology of the emulated fabric and the sequence of commands used to program it are introduced. After that a summary of the frames exchanged between the LNE and the real switch is presented. Some of the frames are expanded and discussed in detail in the last section.

A.1 Scenario

The test fabric is shown in figure A.1. The LNE emulates four switches and has a single connection to the real one. Switch andre is the only border switch. Below is the

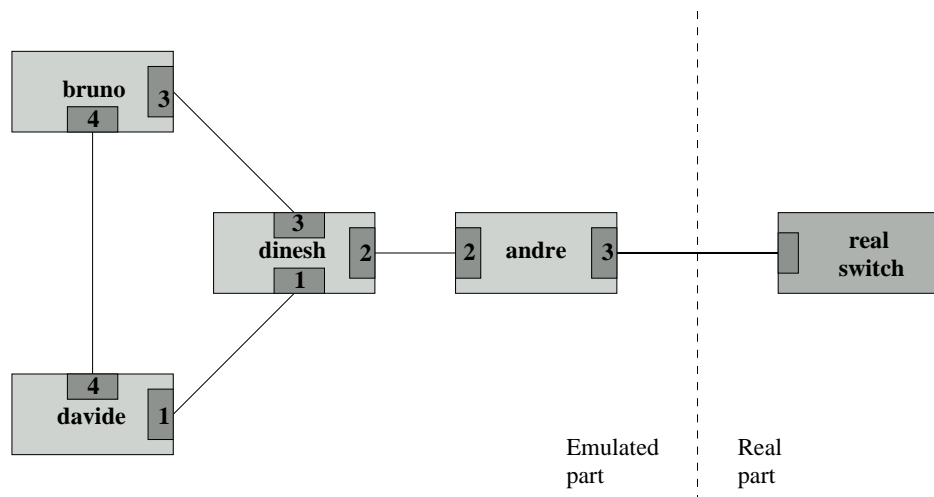


Figure A.1. Emulation session scenario

command sequence to program the topology to be emulated.

The LNE shell starts in configuration mode and `lne-conf>` is the user prompt. Lines that do not begin with the prompt are responses from the LNE.

First we add the four switches:

```
lne-conf>switch-add andre
Switch added, WWN is 20:04:00:05:30:00:02:00.
```

```
lne-conf>switch-add dinesh
Switch added, WWN is 20:04:00:05:30:00:04:00.
```

```
lne-conf>switch-add davide
Switch added, WWN is 20:04:00:05:30:00:06:00.
```

```
lne-conf>switch-add bruno
Switch added, WWN is 20:04:00:05:30:00:08:00.
```

Then we add the links between the switches and we map hba 1 (the first physical interface of the LNE) to andre 3 (the virtual port 3 of switch andre).

```
lne-conf>link-add dinesh 2 andre 2
Link added.
```

```
lne-conf>link-add dinesh 1 davide 1
Link added.
```

```
lne-conf>link-add dinesh 3 bruno 3
Link added.
```

```
lne-conf>link-add bruno 4 davide 4
Link added.
```

```
lne-conf>hba-add andre 3 1
Hba 1 added.
```

At this point, we pass to the run phase and activate the links. The prompt changes to `lne-run>`.

```
lne-conf>run
```

```
lne-run>link-up dinesh 2
Link initializing.
```

```
lne-run>link-up davide 1
Link initializing.
```

```
lne-run>link-up davide 4
Link initializing.
```

```
lne-run>link-up dinesh 3
Link initializing.
```

```
lne-run>link-up andre 3
Link initializing.
```

The final link is a border link, so bringing it up starts traffic generation.

A.2 List of generated frames

The following is a trace (captured with a Fibre Channel network analyzer) of the traffic exchanged between the LNE and the real switch. An arrow going from left to right indicates frames transmitted by the LNE (andre border switch) and received by the real switch, those going the opposite way represent frames flowing in the opposite direction.

All the frames are Class F SW_ILS, source and destination addresses (not shown) are always those of the “fabric controllers” (0xff.f.f.d) of the switches. In this particular case, all the sequences fit in a single frame, so to each frame corresponds a sequence. When a switch initiates an exchange, it does not know yet the RX_ID that will be chosen by the responder, so it simply uses 0xffff. When the responder sends a frame with the RX_ID set, the originator uses the right value. Acknowledge frames have been omitted for clarity.

No.	Time	Dir.	OX_ID	RX_ID	Info
1	0.000000	--->	0x48d6	0xffff	ELP
2	0.009955	<---	0x48d6	0x3655	SW_ACC (ELP)
3	0.019622	--->	0x48d7	0xffff	ESC
4	0.021157	<---	0x48d7	0x3656	SW_ACC (ESC)
5	0.043191	--->	0x2905	0xffff	EFP
6	0.053801	<---	0x29ae	0xffff	EFP
7	0.056766	<---	0x2905	0x29ad	SW_ACC (EFP)
8	0.103481	--->	0x29ae	0x2906	SW_ACC (EFP)
9	9.985092	<---	0x29b1	0xffff	Domain ID Assigned
10	9.988165	--->	0x29b1	0x2909	SW_ACC (Domain ID Assigned)

11	9.990528	--->	0x290a	0xffff	Request Domain ID
12	10.407283	<---	0x29b3	0xffff	Merge Req
13	10.408785	--->	0x29b3	0x290b	SW_ACC (Merge Req)
14	10.414421	<---	0x29b4	0xffff	FSPF: Hello
15	10.567516	<---	0x290a	0x29b2	SW_ACC (Request Domain ID)
16	10.586544	--->	0x290e	0xffff	FSPF: Hello
17	10.588349	--->	0x290f	0xffff	Request Domain ID
18	10.587453	<---	0x29b7	0xffff	FSPF: Hello
19	10.591997	--->	0x2911	0xffff	FSPF: Link State Update
20	10.591145	<---	0x290f	0x29b8	SW_ACC (Request Domain ID)
21	10.593313	<---	0x29b9	0xffff	EFP
22	10.596772	--->	0x29b5	0x290d	SW_ACC (SW_ACC)
23	10.628731	--->	0x2913	0xffff	Request Domain ID
24	10.629622	--->	0x29b9	0x2912	SW_ACC (EFP)
25	10.629471	<---	0x2913	0x29bb	SW_ACC (Request Domain ID)
26	10.631310	<---	0x29bc	0xffff	EFP
27	10.648246	--->	0x2915	0xffff	Request Domain ID
28	10.649237	--->	0x29bc	0x2914	SW_ACC (EFP)
29	10.649093	<---	0x2915	0x29bd	SW_ACC (Request Domain ID)
30	10.651167	<---	0x29be	0xffff	EFP
31	10.665626	--->	0x29be	0x2916	SW_ACC (EFP)
32	15.594472	--->	0x2917	0xffff	FSPF: Link State Update
33	20.602914	--->	0x2918	0xffff	FSPF: Link State Update
34	25.604851	--->	0x2919	0xffff	FSPF: Link State Update
35	30.317549	<---	0x29c2	0xffff	FSPF: Hello
36	30.591113	--->	0x291b	0xffff	FSPF: Hello
37	30.592744	<---	0x29c4	0xffff	FSPF: Link State Update
38	30.612622	--->	0x291d	0xffff	FSPF: Link State Update
39	30.613415	<---	0x29c6	0xffff	FSPF: Link State Ack
40	30.648205	--->	0x291f	0xffff	FSPF: Link State Ack
41	30.649361	--->	0x2920	0xffff	FSPF: Link State Update
42	30.648972	<---	0x29c8	0xffff	FSPF: Link State Update
43	30.651706	<---	0x29ca	0xffff	FSPF: Link State Ack
44	31.601803	--->	0x2923	0xffff	FSPF: Link State Ack
45	31.608453	--->	0x2925	0xffff	FSPF: Link State Update
46	31.611357	<---	0x29ce	0xffff	FSPF: Link State Ack
47	35.616765	--->	0x2928	0xffff	FSPF: Link State Update
48	35.617335	<---	0x29d1	0xffff	FSPF: Link State Ack

A.3 Frame analysis

In this section we analyze some of the frames exchanged between the LNE and the real switch. The payload of these frames is expanded to decode the most relevant information and recall the operations described in the previous chapters.

It is important to remember that both the LNE and the switch used in the lab are experimental devices, so the values of the parameters they use and other characteristics might be different from those specified in the standards ([6], [11] and [16]) and used by commercial products.

A.3.1 Link parameters exchange

Frames 1 to 4 constitute the link initialization procedure describe in section 4.4.2. In this case the LNE is faster than the real switch and sends the ELP first. The ELP contains the port and node name of the switch, the maximum frame size, the supported classes of service and other parameters.

```
Frame 1 SW_ILS
  Cmd Code: ELP (0x10)
  Revision: 2
  Flag: 0000
  R_A_TOV: 10000 msecs
  E_D_TOV: 2000 msecs
  Req Eport Name: 20:04:00:05:30:00:02:03 (00:05:30)
  Req Switch Name: 20:04:00:05:30:00:02:00 (00:05:30)
  Class F Svc Parameters: (Class F Valid | No X_ID Interlk)
  Max Class F Frame Size: 1024
  Class F Max Concurrent Seq: 1
  Class F E2E Credit: 1
  Class F Max Open Seq: 1
  Class 1 Svc Parameters: (Class 1 Invalid)
  Class 2 Svc Parameters: (Class 2 Valid | Seq Delivery)
  Class 2 Frame Size: 1024
  Class 3 Svc Parameters: (Class 3 Valid | Seq Delivery)
  Class 3 Frame Size: 1024
  ISL Flow Ctrl Mode: R_RDY Flow Ctrl
  B2B Credit: 16
```

We can see that the LNE declares that it supports Class 2, Class 3 and Class F traffic, that it initially grants 16 Buffer_to_Buffer credits and has a maximum frame size of 1024 bytes. The limit on the frame size is dictated by the characteristics of the HBA used by the LNE.

Frame 2 is the response of the real switch that accepts the proposed parameters and provides its own. At this point the LNE goes on and transmits an ESC. The ESC is an optional SW_ILS that contains information about protocols supported by the switches. The real switch receives and accept it.

A.3.2 Fabric parameters exchange

After the link initialization is complete, the two switches must discover wether their neighbor is part of an initialized fabric or is unconfigured (see 4.5.1).

In our case, both switches are unconfigured, so they initialize the Principal Switch Parameters to their configured values. The assigned Domain_ID lists are empty.

```
Frame 5 SW_ILS
  Cmd Code: EFP (0x11)
  Payload Len: 16
  Principal Switch Priority: 255
  Principal Switch Name: 20:04:00:05:30:00:02:00 (00:05:30)
```

```
Frame 6 SW_ILS
  Cmd Code: EFP (0x11)
  Payload Len: 16
  Principal Switch Priority: 128
  Principal Switch Name: 20:04:00:05:30:00:43:5f (00:05:30)
```

Note that the LNE has a priority of 255 (meaning that it is not capable of becoming Principal Switch), whereas the real switch has 128.

A.3.3 Domain_ID assignment

After the proper timeout has expired (about ten seconds) the real switch realizes it is the Principal Switch. It assigns itself a Domain_ID and issues Frame 9 (DIA) to notify its peer.

```
Frame 9 SW_ILS
  Cmd Code: Domain ID Assigned (0x12)
  Switch Name: 20:04:00:05:30:00:43:5f (00:05:30)
```

At this point, the real switch should issue an EFP with the updated list of assigned Domain_IDs, but the LNE is faster and immediately sends a RDI for switch andre:

```
Frame 11 SW_ILS
  Cmd Code: Request Domain ID (0x13)
```

```
Payload Len: 16
Req Switch Name: 20:04:00:05:30:00:02:00 (00:05:30)
Requested Domain ID (0): 0x0
```

The Principal Switch postpones the EFP transmission and responds to the RDI granting a Domain_ID:

```
Frame 15 SW_ILS
  Cmd Code: SW_ACC (0x02)
  Payload Len: 16
  Req Switch Name: 20:04:00:05:30:00:02:00 (00:05:30)
  Granted Domain ID (0): 0xa6
```

The LNE goes on requesting new Domain_IDs (Frame 17, 23, 27) and getting them (Frames 20, 25, 20).

The real switch, being the Principal Switch, issues at regular intervals EFPs containing new assigned Domain_IDs (Frames 21, 26, 30). Frame 30 is the last EFP, so it contains Domain_IDs assigned to all the switches in the fabric:

```
Frame 30 SW_ILS
  Cmd Code: EFP (0x11)
  Payload Len: 96
  Principal Switch Priority: 2
  Principal Switch Name: 20:04:00:05:30:00:43:5f (00:05:30)
  Domain ID Record
    Domain ID: 0xa1
    Switch Name: 20:04:00:05:30:00:43:5f (00:05:30)
  Domain ID Record
    Domain ID: 0xa6
    Switch Name: 20:04:00:05:30:00:02:00 (00:05:30)
  Domain ID Record
    Domain ID: 0xa5
    Switch Name: 20:04:00:05:30:00:06:00 (00:05:30)
  Domain ID Record
    Domain ID: 0xa4
    Switch Name: 20:04:00:05:30:00:04:00 (00:05:30)
  Domain ID Record
    Domain ID: 0xa3
    Switch Name: 20:04:00:05:30:00:08:00 (00:05:30)
```

The real switch has Domain_ID 0xa1, iandre 0xa6, dinesh 0xa5, davide 0xa4 and bruno 0xa3.

A.3.4 Hello packets

As soon as the real switch has assigned itself a `Domain_ID`, it starts sending Hello packets (section 4.6.2) to announce its parameters and try to establish the identity of the peer:

```
Frame 14 SW_ILS
  Cmd Code: FSPF: Hello (0x14)
  FSPF Header
    Version: 0x02
    AR Number: 0x00
    Authentication Type: 0x00
    Originating Domain ID: 0xa1
    Authentication: 0000000000000000
  Options: 00000000
  Hello Interval (secs): 20
  Dead Interval (secs): 80
  Recipient Domain ID: 0xffffffff
  Originating Port Idx: 0x001003
```

The Hello Interval is the period between Hello packets and the Dead Interval is the time after which the switch considers the link down if it doesn't receive any Hello packet from its neighbor. The switch announces its `Domain_ID` and notifies the neighbor of the fact that it doesn't know its `Domain_ID` yet by using the value `0xffffffff` in the "Recipient `Domain_ID`" field. This is called a "one-way hello" because only the `Domain_ID` of the sender is known. Switch `andre` sends its one-way hello as soon as he gets his `Domain_ID` (Frame 16).

The next Hello packet is "two-way" because it contains both the `Domain_ID` of the sender and the receiver and both switches agree on the values of the timers, that must be identical:

```
Frame 18 SW_ILS
  Cmd Code: FSPF: Hello (0x14)
  FSPF Header
    Version: 0x02
    AR Number: 0x00
    Authentication Type: 0x00
    Originating Domain ID: 0xa1
    Authentication: 0000000000000000
  Options: 00000000
  Hello Interval (secs): 20
  Dead Interval (secs): 80
  Recipient Domain ID: 0x000000a6
  Originating Port Idx: 0x001003
```

After the first two-way Hello has been received, the switch proceed with database exchange but issue Hello packets at regular intervals (Frames 35, 36).

A.3.5 Initial database exchange

The switches are now supposed to exchange their topology databases, as explained in section 4.6.3. At this time both the real switch and switch *andre* have no adjacencies, so they just tell their peer that the topology database is empty. The LNE and the real switch immediately sends a LSU with no LSRs and the “exchange complete” flag set (Frame 37 and 38).

```
Frame 38 SW_ILS
  Cmd Code: FSPF: Link State Update (0x15)
  FSPF Header
    Version: 0x02
    AR Number: 0x00
    Authentication Type: 0x00
    Originating Domain ID: 0xa6
    Authentication: 0000000000000000
  Flags : LSR is for Initial DB Sync | Last Seq in DB Sync
  Num of LSRs: 0
```

Actually, Frame 38 is a retransmission of Frame 19, the first LSU transmitted by the LNE¹. Frame 19 is retransmitted four times (Frame 32, 33, 34 and 38) until it is acknowledged by the real switch (Frame 39). This is LNE’s fault, as after discovering the identity of its peer (Frame 18), it does not send immediately a two-way Hello, but waits for the Hello timer to expire (20 seconds).

A.3.6 Link state updates

As soon as the initial database exchange is complete, the link is considered operational, so its existence must be notified to all the switches in the fabric. Both the real switch and the virtual one rebuild their LSRs and flood it. The border switch *andre* has one adjacency to the real switch and one to *dinesh* (Frame 41) whereas the real switch has only the adjacency to *andre* (Frame 42).

```
Frame 41 SW_ILS
  Cmd Code: FSPF: Link State Update (0x15)
  FSPF Header
```

¹This is why the adjacency between *andre* and *dinesh* is not yet considered. Domain_ID assignment frame to *dinesh* comes after Frame 19.

```
Version: 0x02
AR Number: 0x00
Authentication Type: 0x00
Originating Domain ID: 0xa6
Authentication: 0000000000000000
Flags : LSR is for a Topology Update
Num of LSRs: 1
Link State Record 0
  Link State Record Header
    LSR Type: Switch Link Record (0x01)
    LSR Age: 0 secs
    Options : 0x40ef1a40
    Link State Id: 00.00.a6
    Advertising Domain Id: 00.00.a6
    LS Incarnation Number: 7
    Checksum: 0x6c1a
    LSR Length: 60
Number of Links: 2
Link Descriptor 0
  Link ID: 00.00.a5
  Output Port Idx: 0x000002
  Neighbor Port Idx: 0x000002
  Link Type: P2P Link (0x01)
  Link Cost: 1000
Link Descriptor 1
  Link ID: 00.00.a1
  Output Port Idx: 0x000003
  Neighbor Port Idx: 0x001003
  Link Type: P2P Link (0x01)
  Link Cost: 1000
```

Frame 42 SW_ILS

```
Cmd Code: FSPF: Link State Update (0x15)
FSPF Header
  Version: 0x02
  AR Number: 0x00
  Authentication Type: 0x00
  Originating Domain ID: 0xa1
  Authentication: 0000000000000000
Flags : LSR is for a Topology Update
Num of LSRs: 1
```

```
Link State Record 0
  Link State Record Header
    LSR Type: Switch Link Record (0x01)
    LSR Age: 1 secs
    Options : 0x0
    Link State Id: 00.00.a1
    Advertising Domain Id: 00.00.a1
    LS Incarnation Number: 2147483783
    Checksum: 0xaf2d
    LSR Length: 44
Number of Links: 1
Link Descriptor 0
  Link ID: 00.00.a6
  Output Port Idx: 0x001003
  Neighbor Port Idx: 0x000003
  Link Type: P2P Link (0x01)
  Link Cost: 1000
```

After that, the LNE floods the LSR that have been generated inside the virtual fabric by dinesh, bruno and davide:

```
Frame 47 SW_ILS
  Cmd Code: FSPF: Link State Update (0x15)
  FSPF Header
    Version: 0x02
    AR Number: 0x00
    Authentication Type: 0x00
    Originating Domain ID: oxa6
    Authentication: 0000000000000000
  Flags : LSR is for a Topology Update
  Num of LSRs: 3
  Link State Record 0
    Link State Record Header
      LSR Type: Switch Link Record (0x01)
      LSR Age: 25 secs
      Options : 0x8ef1a40
      Link State Id: 00.00.a3
      Advertising Domain Id: 00.00.a6
      LS Incarnation Number: 1
      Checksum: 0x7fff
      LSR Length: 60
```

```
Number of Links: 2
Link Descriptor 0
  Link ID: 00.00.a5
  Output Port Idx: 0x000003
  Neighbor Port Idx: 0x000003
  Link Type: P2P Link (0x01)
  Link Cost: 1000
Link Descriptor 1
  Link ID: 00.00.a4
  Output Port Idx: 0x000004
  Neighbor Port Idx: 0x000004
  Link Type: P2P Link (0x01)
  Link Cost: 1000
Link State Record 1
  Link State Record Header
    LSR Type: Switch Link Record (0x01)
    LSR Age: 25 secs
    Options : 0x40ef1a40
    Link State Id: 00.00.a4
    Advertising Domain Id: 00.00.a6
    LS Incarnation Number: 4
    Checksum: 0x7a41
    LSR Length: 60
Number of Links: 2
Link Descriptor 0
  Link ID: 00.00.a5
  Output Port Idx: 0x000001
  Neighbor Port Idx: 0x000001
  Link Type: P2P Link (0x01)
  Link Cost: 1000
Link Descriptor 1
  Link ID: 00.00.a3
  Output Port Idx: 0x000004
  Neighbor Port Idx: 0x000004
  Link Type: P2P Link (0x01)
  Link Cost: 1000
Link State Record 2
  Link State Record Header
    LSR Type: Switch Link Record (0x01)
    LSR Age: 25 secs
    Options : 0x70ef1a40
```

```
Link State Id: 00.00.a5
Advertising Domain Id: 00.00.a6
LS Incarnation Number: 5
Checksum: 0x680b
LSR Length: 76
Number of Links: 3
Link Descriptor 0
  Link ID: 00.00.a4
  Output Port Idx: 0x000001
  Neighbor Port Idx: 0x000001
  Link Type: P2P Link (0x01)
  Link Cost: 1000
Link Descriptor 1
  Link ID: 00.00.a6
  Output Port Idx: 0x000002
  Neighbor Port Idx: 0x000002
  Link Type: P2P Link (0x01)
  Link Cost: 1000
Link Descriptor 2
  Link ID: 00.00.a3
  Output Port Idx: 0x000003
  Neighbor Port Idx: 0x000003
  Link Type: P2P Link (0x01)
  Link Cost: 1000
```

A.3.7 Other frames

During the discussion, we have ignored Frame 12 and Frame 13. These frames are a request from the real switch's zone server to see if its zone parameters are compatible with those of the emulated switch. The LNE at present does not support zoning, so it simply replies that the merge was successful, without considering the actual content of the request frame.

A.4 LNE fabric information

Finally, we can compare the information we have derived from frame analysis with those provided by LNE informative commands:

```
lne-run>show-switch-all
```

```
Switch:          andre
Fabric name:     20:06:00:05:30:00:43:5f
Domain Id:      0xa6
Principal Port: 3
Priority:        255
Generate RCF:   No
```

```
Active Ports:
Port  2    link status:  UP (->dinesh/2), principal downstream
Port  3    link status:  UP (->hba 1), principal upstream
```

```
Switch:          dinesh
Fabric name:     20:06:00:05:30:00:43:5f
Domain Id:      0xa5
Principal Port: 2
Priority:        255
Generate RCF:   No
```

```
Active Ports:
Port  1    link status:  UP (->davide/1), principal downstream
Port  2    link status:  UP (->andre/2), principal upstream
Port  3    link status:  UP (->bruno/3)
```

```
Switch:          davide
Fabric name:     20:06:00:05:30:00:43:5f
Domain Id:      0xa4
Principal Port: 1
Priority:        255
Generate RCF:   No
```

```
Active Ports:
Port  1    link status:  UP (->dinesh/1), principal upstream
```

Port 4 link status: UP (->bruno/4), principal downstream

Switch: bruno
Fabric name: 20:06:00:05:30:00:43:5f
Domain Id: 0xa3
Principal Port: 4
Priority: 255
Generate RCF: No

Active Ports:

Port 3 link status: UP (->dinesh/3)
Port 4 link status: UP (->davide/4), principal upstream

lne-run>

Bibliography

- [1] Lyman P., Varian H. R., *How Much Information ?*, School of Information Management and Systems, University of California at Berkeley, 2000
<http://www.sims.berkeley.edu/how-much-info>.
- [2] Clark Tom, *Designing Storage Area Networks*, Reading, MA, USA, Addison-Wesley Longman Inc., 1999.
- [3] *Fibre Channel Storage Area Networks*, San Francisco, CA, USA, Fibre Channel Industry Association, 2001.
- [4] Krueger M., Haagens R., Sapuntzakis C., Bakke M. *Small Computer Systems Interface protocol over the Internet (iSCSI) Requirements and Design Considerations*, IETF RFC 3347, July 2002.
- [5] Satran J., Meth K., Sapuntzakis C., Chadalapaka M., Zeidner E. *iSCSI, draft-ietf-ips-iscsi-15*, 5 August 2002.
- [6] ANSI NCITS Project 1331-D, *Fibre Channel - Framing and Signaling (FC-FS)*, rev. 1.70, February 8 2002.
- [7] Kembel R. W., *Fibre Channel: A Comprehensive Introduction*, Tucson, AZ, USA, Northwest Learning Associates, 2000.
- [8] ANSI NCITS Project 1235-D, *Fibre Channel - Physical Interface (FC-PI)*, rev. 13, December 9 2001.
- [9] ANSI NCITS Project 1144-D, *Fibre Channel Protocol for SCSI, Second Version (FCP-2)*, rev 7, March 7 2001.
- [10] Rajagopal M., Bhagwat R., Rickard W., *IP and ARP over Fibre Channel*, IETF RFC 2625, June 1999.

- [11] ANSI NCITS Project 1305-D, *Fibre Channel - Switch Fabric - 2 (FC-SW-2)*, rev. 5.4, June 26 2001.
- [12] Kember R. W., *Fibre Channel Switched Fabric*, Tucson, AZ, USA, Northwest Learning Associates, 2001.
- [13] Cormen T. H., Leiserson C. E., Rivest R. L., *Introduction to Algorithms*, 2nd ed., Cambridge, MA, USA, MIT Press, 2001.
- [14] Narváez P., *Routing Reconfiguration in IP Networks*, Ph.D. Thesis, M.I.T., June 2000.
- [15] ANSI NCITS Project 1356-D, *Fibre Channel - Generic Services - 3 (FC-GS-3)*, rev 7.01, November 28 2000.
- [16] ANSI NCITS Project 1377-D, *Fibre Channel - Methodologies for Interconnects (FC-MI)*, rev. 1.92, December 4 2001.