uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Abdelmounaim Dahbi

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Emulation of Limited-perimeter Vector Matching Fault Localization Protocol

TITRE DE LA THÈSE / TITLE OF THESIS

Hussein Mouftah

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

Chung-Horng Lung

Nancy Saaman

Gary W. Slater

Le Doyen de la Faculte des études supéneures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Emulation of Limited-perimeter Vector Matching Fault Localization Protocol

Abdelmounaim Dahbi

A thesis submitted to the Faculty of Graduate Studies and Postdoctoral Studies in partial fulfillment of the requirements for the degree of

Master in Computer Science

Ottawa-Carleton Institute for Computer Science
School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada

July 2010

Your file Votre référence
ISBN: 978-0-494-73803-0
Our file Notre référence
ISBN: 978-0-494-73803-0

To my family...

Abstract

Survivability in all-optical networks is an imperative requirement. There are two types of restoration techniques: link restoration and path restoration. Although link restoration is advantageous in terms of restoration speed and network load balancing, it requires to be preceded by an efficient fault localization procedure. This explains the importance of fault localization in all-optical networks.

Limited-perimeter Vector Matching (LVM) Fault localization Protocol was designed to be fast and efficient. This thesis presents the emulation process for LVM Protocol. The emulation process consists of an all-optical network monitoring multi-layer software implementing the LVM protocol, we called it LVM Explorer. LVM Explorer provides a GUI as the presentation layer. The protocol is launched by switch services mapping the physical switches. The switch services are running in parallel as distributed processes.

The main merit of the LVM Explorer is that it handles fault localization outside the physical network which is called only for alarms.

Acknowledgements

It is a great pleasure to thank the many people who made this thesis possible.

It is difficult to overstate my gratitude to my supervisor, Professor Hussein T. Mouftah, whose distinguished expertise and deep understanding, added considerably to my graduate experience. I am deeply grateful to him for his great guidance, continuous support and considerable patience. Throughout my thesis-writing period, Professor Hussein T. Mouftah incessantly provided instructive comments and helpful advice. My infinite thanks go to him.

Also, I wish to thank all members of my family, my friends and my colleagues for providing a loving environment for me. My parents have been particularly supportive. My wife has been very considerate and encouraging. The unconditional love of my little treasure Maryam has been a source of inspiration for me.

Finally, I would like to thank the many people who have taught me throughout my life.

Contents

List of Figures

Abbreviations

AA	**A**uthentication and **A**uthorization
ALV	**A**ffected **L**ink **V**ector
AON	**A**ll **O**ptical **N**etwork
API	**A**pplication **P**rogramming **I**nterface
BDI	**B**ackward **D**efect **I**ndicator
BER	**B**it **E**rror **R**ate
BPEL	**B**usiness **P**rocess **E**xecution **L**anguage
COBNET	**C**orporate **O**ptical **B**ackbone **NET**work
CRUD	**C**Reate **U**pdate **D**elete
DP	**D**iagnosis **P**hase
EJB	**E**nterprise **J**ava **B**eans
EMS	**E**nterprise **M**anagement **S**ystem
E/O	**E**lectrical-to-**O**ptical
ES	**E**xecutive **S**ink
FDI	**F**orward **D**efect **I**ndicatore
FLA	**F**ault **L**ocation **A**lgorithm
FLSP	**F**ault textbf**L**ocalization textbf**S**ignaling textbf**P**rotocol
GMPLS	**G**eneralized **MPLS**
GUI	**G**raphical **U**ser **I**nterface
HDFS	**H**euristic **D**epth **F**irst **S**earching
HST	**H**euristic **S**panning **T**ree
ILP	**I**nteger **L**inear **P**rogram

IP	Internet Protocol
JMS	Java Message Service
JNDI	Java Naming and Directory Interface
LMP	Link Management Protocol
LRS	Loop-back Restoration Scheme
LVM	Limited-Perimeter Vector Matching
MC	Monitoring Cycle
MP	Monitoring Path
MPC	Monitoring with Paths and Cycles
MPLS	MultiProtocol Label Switching
NE	Network Element
OADM	Optical Add Drop Multiplexer
OCh	Optical Channel
O/E	Optical-to-Electrical
OMS	Optical Multiplex Section
ORM	Object Relational Mapping
OS	Operations System
OTS	Optical Transmission Section
OXC	Optical Cross Connect
PCP	PreComputational Phase
PES	Potential Executive Sink
P-LVM	Parallel Limited-Perimeter Vector Matching
RBA	Rolling Back Alarm
RBSP	Rolling textbfBack textbfSignaling textbfProtocol
SDH	Synchronous Digital Hierarchy
SOA	Service Oriented Architecture
SONET	Synchronous Optical Network
SPE	Synchronous Payload Envelope
SPEM	Shortest Path Eulerian Matching

STS	Synchronous Transport Signal
TCP	Transmission Control Protocol
TI	Trace Identifier
TL1	Transaction Language 1
UFL	Unambiguous Fault Localization
WDM	Wavelength Division Multiplexing

Chapter 1

Introduction

1.1 Motivations

One of the most important characteristics of optical fibers is their huge bandwidth capacity. Theoretically, Optical fibers can support a combined bit rate into the range of terabits per second. Although this feature represents a major merit of optical fibers over other metallic wires, it also introduces a critical issue for optical networks. Because of the enormous transmission bandwidth in optical networks, a single network failure such as a fiber cut could lead to a massive loss of data [1] resulting in network performance degradation or even the network services disruption.

Survivability stands for the capacity of a telecommunication network to recover from network failures and maintain a high level of service availability. It is a crucial feature in all types of telecommunication networks, and especially in wavelength-routed WDM networks in which each optical fiber carry a certain number of optical channels (wavelengths) each of which operates at a very high data rate [1].

In wavelength-routed WDM Optical networks, as in all types of telecommunication networks, node failures and link failures are the two main network failures. Usually, Fiber or cable cuts are the reason behind link failures and hardware/software failures are the cause of node failures. Node failures may occur for different reasons including human errors and hardware degradation. Cable or fiber cuts may also occur either as a result of a human error or a natural phenomenon such as an earthquake. Although node failures are usually more worrying than link failures in terms of the number of network users who might be affected, link failures are still more frequent an more challenging to locate and fix since fiber cables are deployed in harsh environments over very long distances under various geographical conditions [2].

Although the survivability of an optical network can be provided at higher network layers, there are several good reasons why it is advantageous to implement it at the optical layer, namely faster and more efficient service recovery as well as significant cost savings. Network survivability at the optical layer is based on two basic survivability paradigms: static protection and dynamic restoration [1].

In static protection, also known as predetermined protection, spare network resources are reserved during network design or at the time of connection establishment for protection against network failures. Obviously, static protection is fast in service recovery since the reserved network resources are dedicated for network failures. However, it is inefficient in terms of resource utilization because the reserved network resources can not be used for other traffic demands [2].

In dynamic restoration, no spare network resources are reserved during network design or at the time of connection establishment. The network must search dynamically for spare network resources available in the network in order to recover the disrupted services after the network failures occur. This makes dynamic restoration more efficient in resource utilization but slower in service recovery because it takes time to search dynamically for spare network resources. Moreover,

the service recovery can not be guaranteed because the network may not have sufficient spare resources at the time the failure occurs [1].

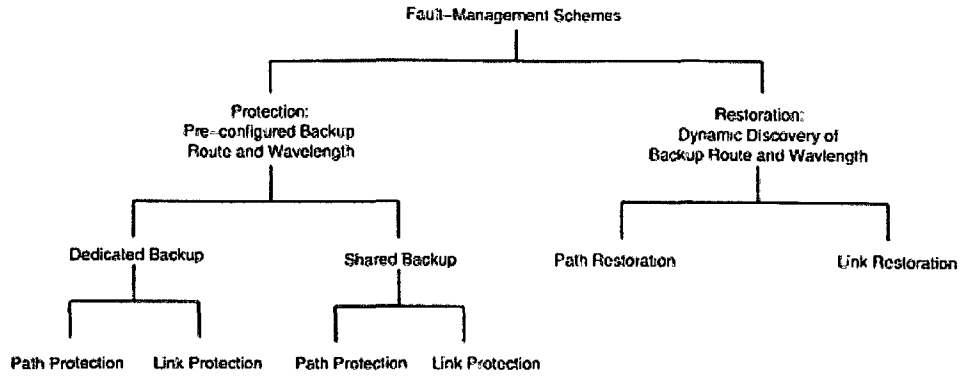


FIGURE 1.1: Basic Protection and Restoration Schemes [1]

In general, two types of restoration techniques (by restoration we mean both static protection and dynamic restoration), namely link restoration and path restoration, are used for recovery from link failures. Figure 1.1 shows both link-based and path-based protection and restoration schemes. Link restoration is a point-to-point recovery technique, while path restoration is an end-to-end recovery technique. In link restoration, the affected path is restored by replacing only the faulty link, while in path restoration, the whole failing path is abandoned and replaced by the backup path which could cause a sudden shock to the load balancing of the system after failure detection. Although, link restoration does not remove the affected path capacity from the network resources, it could create a congestion around the failed link. This phenomenon may only occur in highly loaded networks [3]. The main drawback of using link restoration techniques (Link Shared Static Protection, Link Dedicated Static Protection and Link Dynamic Restoration in Figure 1.1) would be its need of a fault localization procedure beforehand in order to identify the faulty link. This explains the crucial need for efficient and fast fault localization protocols.

The LVM protocol is a novel fault localization protocol that can precede a link restoration procedure in an all-optical network. It has been shown analytically in [3] that LVM protocol is highly efficient in terms of time and space complexities, compared to a similar but boundless centralized protocol. Also, it has been shown through simulation results in [4] that LVM protocol achieve excellent performance in terms of fault localization time while maintaining the maximum-level transparency of all-optical networks. This thesis focuses on its emulation on real optical networks without having to deploy the protocol on the physical switches. The emulation process aims to prove the implementability of the LVM protocol as well as its excellent performance.

By Emulation we mean that the LVM Explorer behavior is exactly the same as that of the LVM protocol implemented directly on the physical switches. In other words, we get the same output for same input under similar conditions, but the mechanism to arrive to that output (from the input) is different. Simulation is however based on models of the LVM protocol. It is carried out through pure software with no involvement of the physical equipment.

1.2 Thesis Objectives

The main objectives of this thesis are the following:

1. To contribute into the design of survivable all-optical networks by contributing into the design and the deployment of efficient and fast fault localization protocols. The main goal of the implementation of the LVM protocol is to confirm the excellent performance of the protocol (shown both analytically [3] and by simulation [4]) on real networks.

2. To design and develop an efficient multi-layer and scalable software which localizes faulty links using LVM protocol in a network without having to

be run on the physical switches. This software can be used eventually to implement other fault localization protocols thanks to its multi-layer feature. It will also be upgraded in order to implement the multi-failure version of LVM protocol.

1.3 Thesis Contributions

The contributions of the present thesis are the following:

1. We developed an efficient and scalable software package capable of running the LVM protocol for fault localization in all-optical networks without having to touch anything on the physical switches. All the work is carried out on a virtual network mapping the physical network. The physical network is required only to trigger the process by notifying the alarms when a failure occurs.

2. We proved, through a series of tests, the advantage of using a limited-perimeter area for both the multicasting and determining phase in the LVM protocol. We demonstrated the important gain in performance when the protocol operates within a limited-perimeter-area instead of the whole network.

3. We implemented the extension of the limited perimeter of the LVM protocol by one hop at a time. This ensures that the LVM Explorer keeps launching the LVM protocol as long as there is room for the extension of the limited-perimeter area. As a result of this contribution, the answer "Not enough traffic to localize the failure" is returned by the LVM explorer only when the LVM protocol was tried over the whole network in vain.

1.4 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 describes the different protection and restoration schemes and specifies where the fault localization procedure is needed, and then it presents a survey on the existing fault localization techniques with their merits and drawbacks. Chapter 3 presents the emulation process with a focus on the architecture design of the LVM Explorer and its three layers. Chapter 4 describes the communication process taking place between the layers of the LVM Explorer on one hand, and on the other hand between the physical equipment and the LVM Explorer. Chapter 5 is dedicated for performance evaluation of both the software and the LVM protocol based on a series of tests carried out on different network topologies. Finally, Chapter 6 concludes this thesis and briefly provides highlights for future work perspectives.

Chapter 2

Existing Fault Localization Techniques

2.1 Introduction

This chapter is mainly dedicated to a survey of some recently proposed fault localization techniques for all-optical networks. It also contains discussions about the merits and the drawbacks of those techniques. These discussions provide a frame of reference to which the performance of the LVM fault localization protocol can be compared.

Right before the literature review regarding the fault localization techniques, this chapter presents some background material about protection and restoration schemes for optical layer survivability. The goal being to specify where the contributions of the present thesis fit into the big picture.

2.2 Background

Although survivability can be provided in several network layers through different protection and restoration schemes, it is still appealing to implement it on the optical layer when dealing with WDM networks.

2.2.1 Optical Layer Survivability

Although Optical Layer survivability has some limitations such as its inability to detect increased bit error rates of traffic and also its limitation regarding handling traffic at a granularity finer than one lightpath [2], it also has a number of merits over survivability provided on other network layers. Among those merits are the following:

- A faster service recovery can be provided through the optical layer than through the higher layers. It is in terms of milliseconds while it is in terms of seconds for higher layers.
- The optical layer protection and restoration functions can be cost-effective. The conversion from the optical domain into the electronic domain is not necessary.
- Isolating failure recovery within the network optical domain prevents alarm storms that would affect all the higher layers. This would reduce the management cost of the control plane.

There are also fault localization techniques that operate on higher layers such as GMPLS [5, 6] and LMP [7, 8] fault localization techniques. These approaches are not part of the scope of this thesis since their fault localization time is in seconds whereas the typical time constraint for fault recovery in optical networks is 50

milliseconds. So the focus in this thesis will be on fault localization protocol in the Optical Layer.

Many other works used spectrum analyzers and photo-diodes to detect and localize failures. These techniques are not cost-effective. They can be found in [9], [10], [11] and [12].

2.2.2 Protection and Restoration for Optical Layer Survivability

An optical layer failure might be either a channel failure, a node failure or a link failure. Link failures are more frequent than node and channel failures. They are also more challenging to handle since their localization is a complex issue. Fiber cuts are the most common link failures.

Protection can be defined as the action performed in advance of a failure intending to defend the network from any possible interruption and restoration can be defined as the action conducted to restore the network availability after a failure occurs [13].

Figure 2.1 illustrates where the fault localization procedure fits into the big picture of protection and restoration schemes. All link-based protection and restoration schemes in Figure 2.1 have to be preceded by a fault localization procedure in order to identify the faulty link. Link-based protection and restoration schemes require that the faulty link is known first. This makes the fault localization a crucial step in the link-based protection and restoration schemes.

Two cases were studied in the literature: single failures and multiple simultaneous failures. Although multiple failures do occur, they still are very rare since the probability that two links go down simultaneously is very small. We focus in the present thesis on single failures rather than multiple simultaneous failures

assuming it is much less likely that a new failure occurs before the previous one is processed.

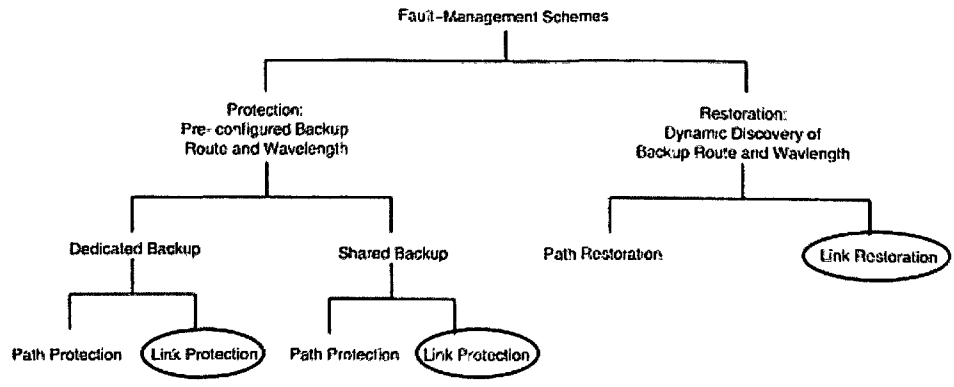


FIGURE 2.1: Basic Protection and Restoration Schemes [1]

2.3 Fault Localization in SONET

SONET is based on the Synchronous Transport Signal STS-1 frame consisting of 9 rows and 90 columns with each cell holding an 8-bit byte [14]. The first three columns are dedicated for section and line header information and is called Transport Overhead (Figure 2.2 illustrates the line and section layers within the optical layer). The remaining columns carry the data and is called the Synchronous Payload Envelope(SPE) [15]. The role of the overhead is to ensure that the connection is well managed before or after any disruption. Figure 2.3 illustrates the STS-1 frame structure.

Fault localization in SONET requires examining overhead at each node, and especially of certain specific bytes dedicated for fault detection.

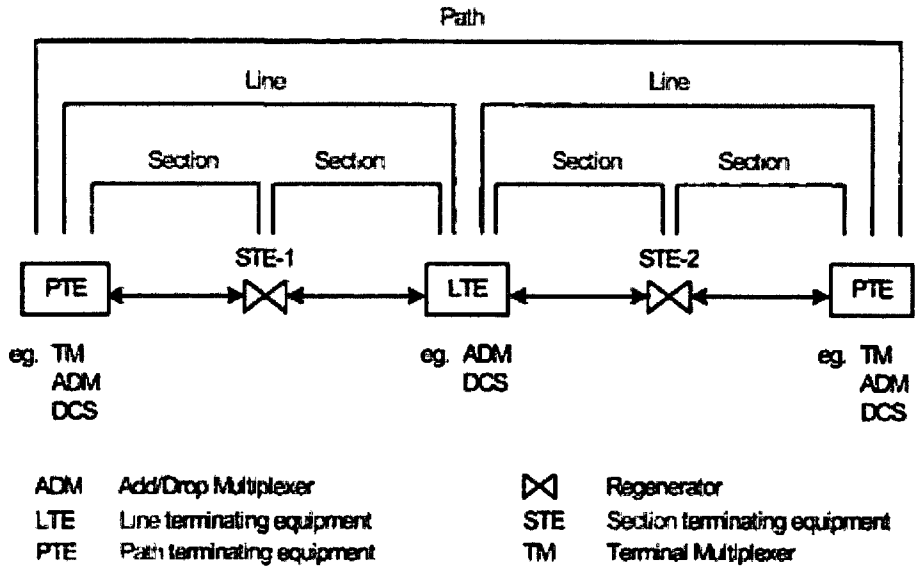


FIGURE 2.2: SONET Section, Line and Path Illustration [16]

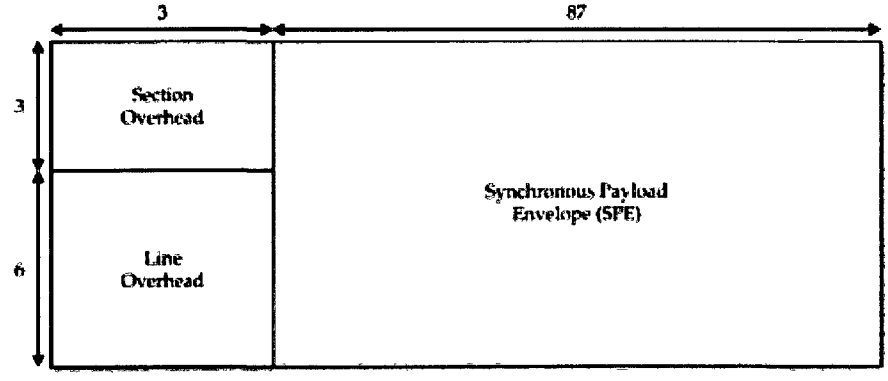


FIGURE 2.3: SONET STS-1 Frame Structure [17]

- Trace Identifier (TI): The Trace Identifier is used to check whether the signal is carried towards the correct destination. The Trace Identifier has a particular importance in WDM Networks where a wavelength might be converted to another.

- Forward Defect Indicator (FDI): As a single failure may create an alarm storm by causing multiple alarms to be generated and consequently several improper actions to be taken, it is important to identify the source of the alarm and then to remove/ignore all the remaining alarms. The Forward Defect Indicator is used mainly to suppress false alarms produced by downstream nodes.

- Backward Defect Indicator (BDI): Following a failure notification in the network, the end terminals acknowledge the notification via the Backward Defect Indicator which is used as a response.

Although SONET/SDH detection techniques are significantly developed and widely used, they are unfortunately not applicable to WDM networks with current features. Reference [10] has shown that some typical schemes for SONET/SDH could not be applied to All-Optical Networks.

2.4 Fault Localization in WDM Mesh Networks

Unfortunately, existing fault localization methods designed for conventional electrical networks are not applicable to all-optical networks. This is mainly due to the lack of electrical terminations in such networks [18]. It has even been shown in [19] that some typical fault detection mechanisms designed for SONET/SDH could not be used for all-optical networks.

It is well-known that fault detection and localization can be performed in higher network layers such as the IP layer. Most routing protocols in the IP layer enclose

the fault detection functionality [20]. However, the fault detection process takes a long time in such protocols which makes them not fast enough in terms of service recovery. Therefore, some efficient fault localization mechanisms running at the optical layer are needed.

2.4.1 Fault Localization Signaling Protocol (FLSP)

Fault Localization Signaling Protocol is a fault localization protocol capable of identifying the location of a failure in a distributed way [21]. It operates in the optical layer and functions under the assumption that the failure occurs during the data transfer. This implies obviously that the destination node is aware of both the established path and the source node [22]. FLSP is similar to Pingger-Pongger Protocol [23].

The connection disruption activates the protocol which operates as follows:

- A fault localization signal, which consists of a short packet, is broadcasted by the destination node in parallel to all the nodes belonging to the failed path. Obviously the number of the fault localization signals to be sent by the destination node is equal to the number of hops in the established route. The fault localization signals travel via the control network, which is assumed to be secured.
- Each recipient of the fault localization signal sends back a small acknowledgement to the destination node through the faulty lightpath.
- The use of the faulty lightpath to send the acknowledgement reveals to the destination node the number of nodes whose the acknowledgement is not received. For example, if only one acknowledgement is missing, this implies that the link adjacent to the source node is the faulty one.

- The identified faulty location is broadcasted to the network and a fast link restoration protocol is launched.

Figure 2.4 illustrates the required steps to restore a failed connection by the Fault Localization Signaling Protocol.

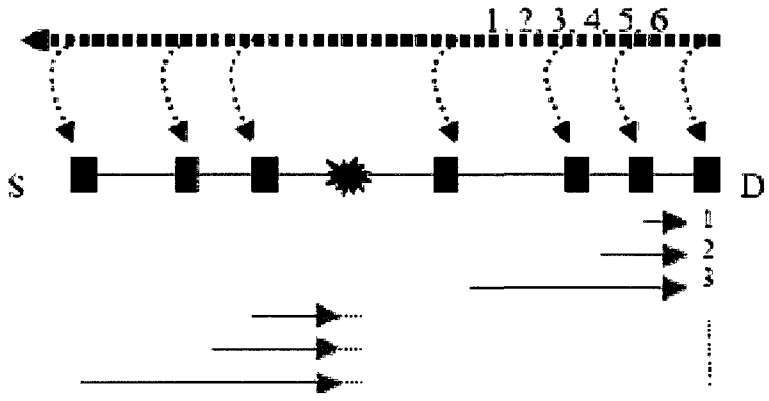


FIGURE 2.4: The illustration of the required steps to restore the connection by FLSP [22]

2.4.2 Rolling Back Signaling Protocol (RBSP)

The Rolling Back Signaling Protocol is fault localization protocol that operates in the WDM layer. It is capable of identifying the location of the failure in a failed lightpath [24]. The protocol operates under the following assumptions:

- The failure occurs during the data transfer. This implies obviously that the destination node is aware of the source node and the set up path before any failure.
- Intermediate nodes are not capable of power monitoring. As a result, they are not able to detect and report high Bit Error Rate (BER) events.

The protocol is activated when the destination node does not receive the expected data stream. It is the destination node that triggers the protocol as follows:

- An alarming signal, which consists of a short packet and is called Rolling Back Alarm (RBA), is sent by the destination node to its upstream neighbor.
- The recipient of the RBA alarm checks whether data has been received at this level or not. In the case the data has not been received, the RBA alarm recipient node ignores the alarm and rolls it back to its upstream neighbor.
- In the case the data has been received successfully at the RBA alarm recipient, an acknowledgement signal is sent back to the downstream sender.
- Once the acknowledgement signal is received, it will activate a restoration protocol to reroute the data over the localized failed link. Loop-back Restoration Scheme (LRS) [25] can be used for that matter.
- the identified location of the failed link is then flooded in the whole network in order to maintain the routing tables accordingly.

Figure 2.5 illustrates the required steps to restore a failed connection by The Rolling Back Signaling Protocol.

2.4.3 Island-Based Restoration Technique

In Island-Based architecture, the network is split into a number of sub-networks called "Islands". Both node and link faults are handled by this techniques. The way the islands are constructed makes it possible to localize a node failure within at most one island and a link failure within at most two islands [26]. The islands are predefined during the initial network design and obviously should be updated in the case the topology of the network changed. [26] suggests three partitioning methods, namely minimal island, shortest path island and 2-stage island. Minimal

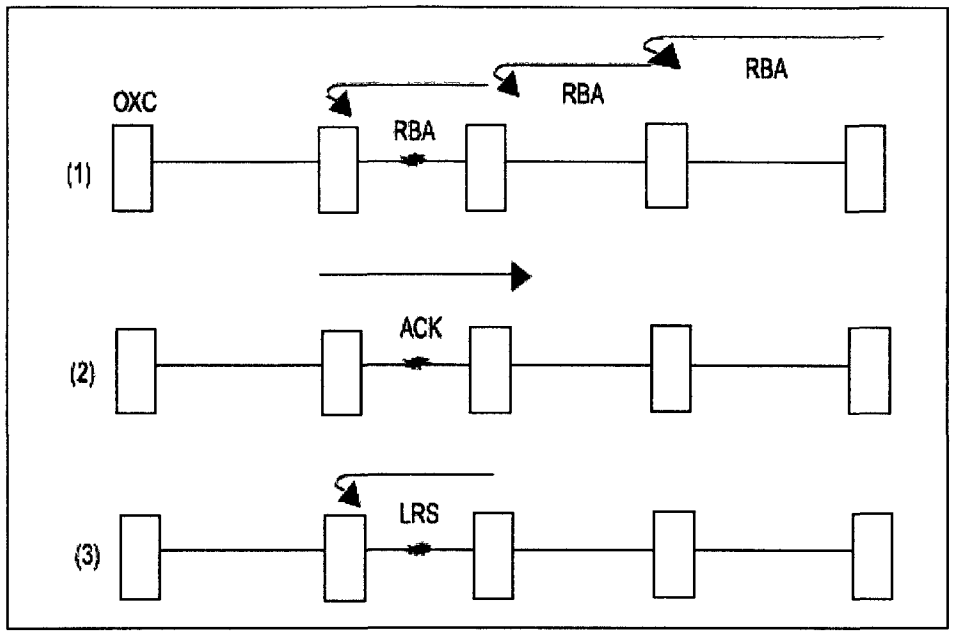


FIGURE 2.5: The illustration of the required steps to restore the connection by RBSP [24]

islands have the advantage of rapidly reacting to failures; while larger islands have the merit to be more flexible and to have better capacity utilization for restoration.

Although this technique looks efficient since it operates in small areas of the network called islands (One islands in the case of a node failure and at most two islands in the case of a link failure), it has the inconvenience of predefined islands during the initial networks design, and also it lacks capability of explicitly locating a failure.

It has been claimed in [26] that Island-Based Restoration technique outperforms segmented restoration protocol [27] in terms of the time delay, overhead and complexity.

2.4.4 Distributed Hierarchical Monitoring Technique

An approach for efficient centralized optimization of activated optical-power monitors has been presented in [28] and [29]. The objective of this approach is to minimize the number of reported alarms following a failure. Obviously, the smaller the number of alarms is, the faster is the fault localization process. It has also been proved in [30] that the monitor activation problem in Transparent Optical Networks is an NP-hard problem. In [30], the idea proposed in [28, 29] has been expanded by developing a scheme for optimization of hierarchically distributed monitor-activation and fault-localization.

The Distributed Hierarchical Monitoring Technique consists of partitioning the network into a number of independent monitoring domains each of which is handled by a local fault-manager. This would enable parallel optimization of the alarm vector size (monitor-activation) and the fault-localization processing by the local fault-managers. This also presents a high scalability scheme capable of localizing failures at multiple granularity levels [30].

The main merit of this technique is that it reduces the number of exchanged messages since each fault-manager operates in its own domain.

A heuristic is proposed in [30] in order to minimize the number of activated monitors while maintaining full fault-coverage. The minimization of the number of activated monitors is important since the fault-manager can be flooded with a large number of redundant alarms by the fault-propagation in transparent optical networks. It is known that redundant alarms complicate rapid fault localization.

2.4.5 Soft and Hard Fault Localization Algorithm Operating in the Physical Layer

This algorithm, called Fault Location Algorithm (FLA), is capable of locating multiple failures at the physical layer of a WDM network. it improves the previous version implemented in the ring network [31].

FLA has the ability to detect both hard and soft failures. The algorithm is non polynomial in terms of its computational complexity. This is the reason why it has been split into two operation phases in order to speed up the fault localization process. The first step, called PreComputational Phase (PCP), is carried out off-line when the optical channels are set up or cleared down and before any alarm is received [32]. A binary tree is generated at the end of the PCP phase based on the established paths in the network.

During the second phase, which is called the Diagnosis Phase (DP), the binary tree generated in the PCP phase is traversed based on the received alarms. The traversal of the tree is performed in order to localize the failures upon receiving alarms from active components [32].

Most of the processing time in FLA is spent during the PCP phase which makes it particularly appropriate for large meshed networks. Another important feature of the FLA algorithm is that it combines information on hard failures with information on soft failures. Information on hard failures is gathered at the WDM layer while information on soft failure is gathered at WDM, SDH, or IP layers.

Although the protocol is capable of detecting multiple failures, it is still unable to detect failures associated with empty leaves in the tree. Moreover, it is only useful for opaque networks and not for transparent optical networks.

2.4.6 A Finite State Machine based technique

This technique investigates fault localization mechanisms in an all-optical network, and requires no optical-to-electrical (O/E) nor electrical-to-optical (E/O) conversion [33]. Fault detection and isolation mechanisms have been proposed in [33] for the following types of networks:

- Broadcast-and-select networks in which nodes are interconnected to each other via a star coupler. An optical fiber link, called the uplink, transports signals from each node to the star while another optical fiber link, called downlink, carries the signal from an output of the star to each node. Figure 2.6 illustrates an example of a Broadcast-and-select network.
- Wavelength-routed networks which consist basically of static or reconfigurable wavelength routers. Those routers, which provide static or reconfigurable lightpaths between end-nodes, are interconnected by fiber links. Figure 2.7 illustrates an example of a Wavelength-routed network.

Figure 2.8 illustrates a fault detection and recovery finite-state machine used to keep track of the current state of the link at the end node. The finite-state machine is driven by the output of a photo-diode that detects the presence or absence of light on the link.

The main drawback of this approach is that its deployment becomes very complex when applied to large and dynamic networks.

2.4.7 Monitoring-Cycles based Fault Localization Technique

This technique consists of decomposing the all-optical network into a set of cycles in such a way that all the nodes and the links of the network appear in at least one of the cycles. These cycles are called "monitoring cycles" (m-cycle). The

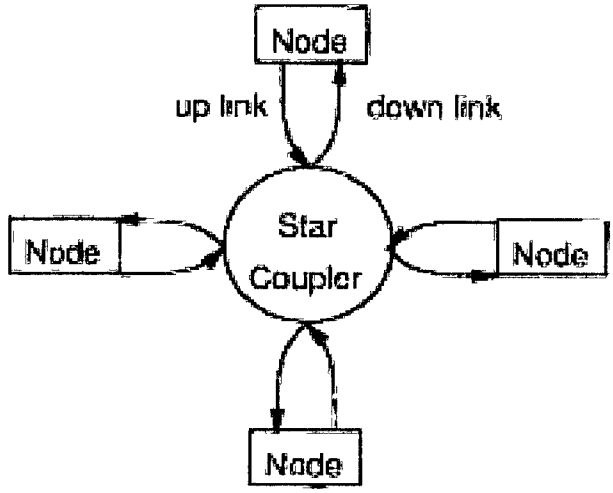


FIGURE 2.6: A broadcast-and-select network [33]

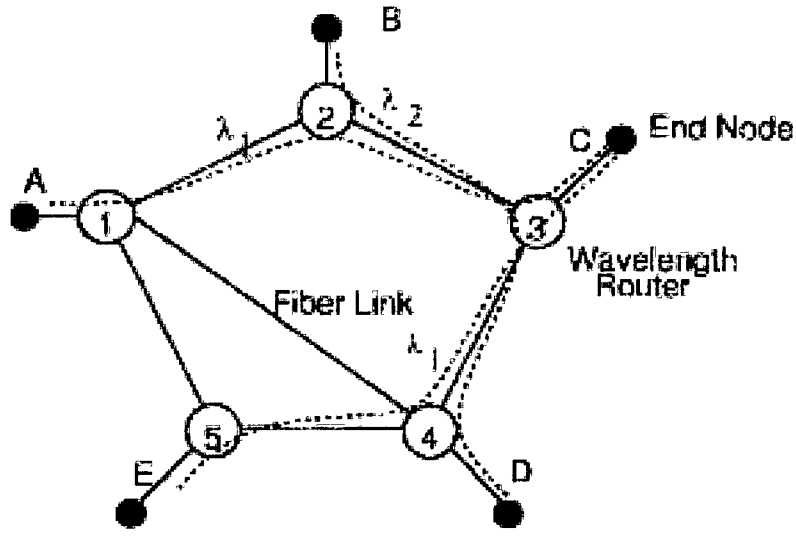


FIGURE 2.7: A wavelength-routed network [33]

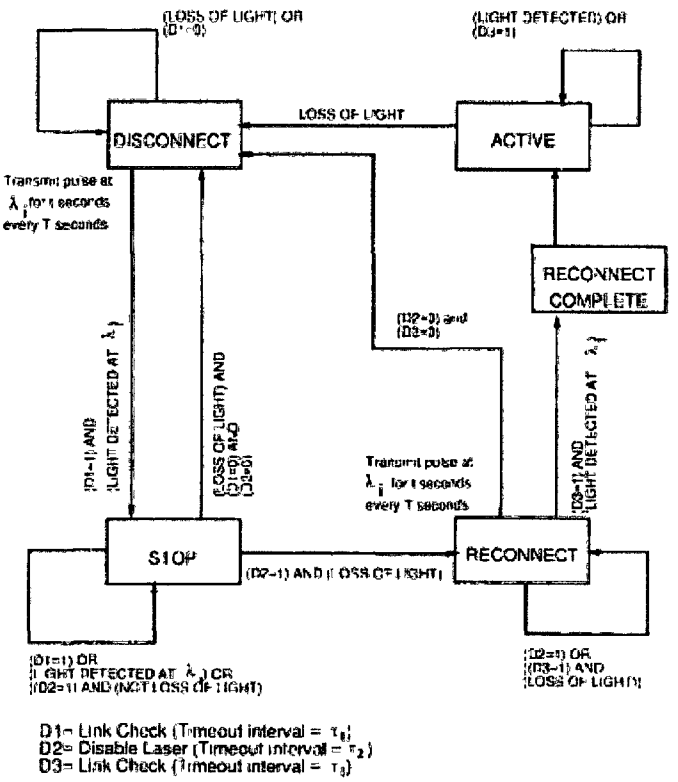


FIGURE 2.8: A Fault detection and recovery finite-state machine [33]

monitoring cycles form a cycle cover of the network. The capacity of monitoring in those cycles comes from two assignments:

- The assignment of a monitor in one node of each cycle.
- The assignment of a loop-back supervisory channel on each cycle.

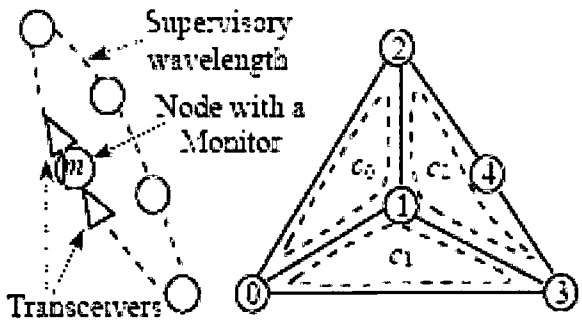
Once a fault occurs in any node or link belonging to a given monitoring cycle, the optical signal in this cycle will be disrupted. Therefore an alarm is triggered within that cycle [18]. Following the alarming of the monitors of all the affected cycles, an alarm code is generated and an alarm code table is used to identify the faulty link. The alarm code is in the form of a vector where each element corresponds to

one of the cycles. If a cycle is affected by the failure, it has the value 1, otherwise 0.

While only simple cycles are considered in [18], reference [34] tries a more flexible approach by considering non-simple cycles as well. A simple cycle can not cross a node more than once, while a non-simple cycle can cross a node several times. Reference [34] proposed to minimize the network cost for link failure detection by considering the trade-off between the monitor cost and the wavelength cost.

M2-CYCLE is another cycle-based technique for fault localization, It consists of preferring m-cycles with the minimum length. It has been claimed [35] that M2-CYCLE uses the least amount of network resources compared with other cycle-based fault localization techniques. The drawback of the M2-CYCLE approach is that it requires more monitors since it prefers minimum length m-cycles. It is however doing better than the spanning tree-based m-cycle approach as it has been proven in [35].

Figure 2.9 shows an example of monitoring cycle fault localization technique. Based on the constructed alarm code which depends on the generated alarms, the alarm code table in Figure 2.10c is used to identify the corresponding faulty link.



Link	c_2	c_1	c_0	Decimal
(0,1)	0	1	1	3
(0,2)	0	0	1	1
(0,3)	0	1	0	2
(1,2)	1	0	1	5
(1,3)	1	1	0	6
(2,4)	1	0	0	4
(3,4)	1	0	0	4

(a) m-cycle (b) An m-cycle solution (c) Alarm code table

FIGURE 2.9: m-cycle based fast link failure localization [36]

A cycle cover for the network has to be found in order to localize faults for all nodes and links in a given network. The cycle cover has to minimize the occurrence of each node and each link in all monitoring cycles so as to minimize the number of wavelengths occupied by monitoring channels in all nodes and links.

Figure 2.10 gives an example of a network with 10 nodes and an instance of its cycle covers. This cover consists of 4 cycles. Some nodes and links appear only in one cycle of the cover. For example node c, link bc and cg, are covered by cycle (4) only. But some others appear in multiple cycles, e.g. edge bd is covered by both cycle (1) and (4), node f by cycle (2), (3) and (4) [18].

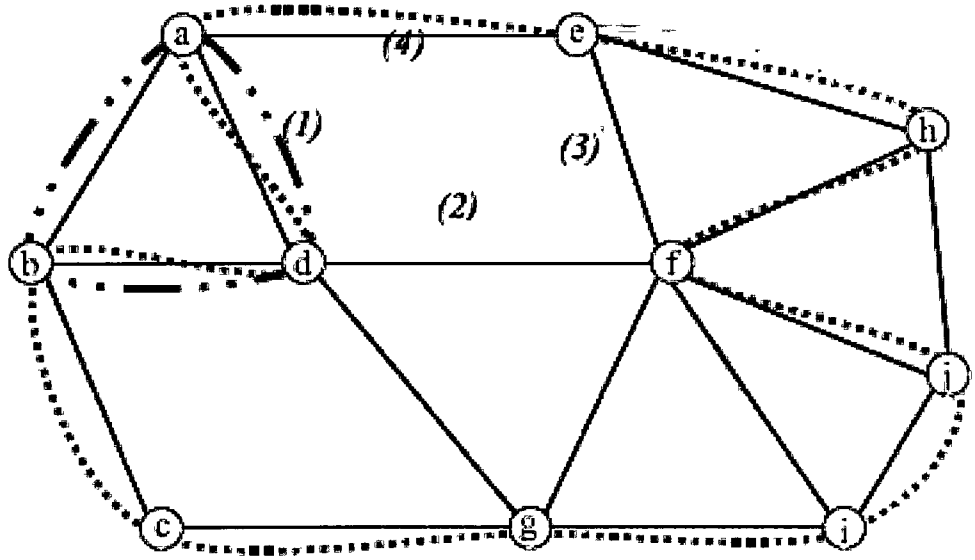


FIGURE 2.10: A graph example and a cycle cover instance [18]

There are many algorithms to form cycle covers on a network. Heuristic depth first searching (HDFS) and Shortest path Eulerian matching (SPEM) were used in [37] while Heuristic spanning-tree (HST) based cycle construction was adopted in [18] for finding a cycle cover in given graphs. Although, the HST algorithm requires more monitors than in HDFS and SPEM, it allows better cost gains when

it comes to complete localization [18]. Complete localization requires more link-based monitors for the HDFS and SPEM algorithms. The cost gain in [18] taking into account both the cycle-based monitors and the link-based monitors for the three algorithms (HDFS, SPEM and HST) shows the HST stays in the worst case comparable to the other two algorithms.

The M2-cycle algorithm, which searches cycle covers with minimum length cycles does better than any spanning-tree based algorithm including HST algorithm. It saves in terms of both cover length and monitoring wavelength requirement.

Compared to a channel-based monitoring scheme requiring one monitor per channel and to a link-based monitoring scheme necessitating one monitor per link, the m-cycle technique has the merit to reduce the number of monitors needed for the fault localization process. This cuts down the hardware cost of additional monitors and also minimizes the network management effort.

On the other hand, a cycle-based monitoring scheme is not able to localize the failure if two links form a cut in the network topology. This is to say that when a segment is part of a network (e.g. link (2, 4) and link (3, 4) in Figures 2.9b and 2.9c), where a segment is a path consisting of at least two links with a degree of two at any intermediate node (such as 2-4-3 in Figure 2.9b), more link-based monitors are required to be able to achieve unambiguous link failure localization on the network. This has two drawbacks, first it has more monitors to manage on the network, which leads to more network management effort, and second the alarm code is longer.

The need for link-based monitors in some special cases led to think about new paradigms other than the cycle. In [38] a combination of the monitoring cycle concept and the monitoring path concept is introduced for unique identification of single-link failures. In [36], a new concept monitoring trail (m-trail) is proposed.

The LVM protocol has the merit of not requiring the assignment of a loopback supervisory channels. It makes use of the channels of the network to localize the failure in an unambiguous way.

2.4.8 Monitoring-Cycles/Monitoring-Paths based Fault Localization Technique

This technique states that the monitoring cycles (MCs) and the monitoring paths (MPs) are constructed in such a way that any single failure would have a unique identifier of the monitoring locations failures. It has been proven that three edge connectivity on each node is a necessary and sufficient condition in the case we want to use only MCs with only one monitoring location for unambiguous failure localization.

The assignment of a supervisory channel for each cycle or path is a requirement for monitoring-cycles/monitoring-paths based fault localization technique. This requirement, not needed for the LVM protocol, presents an advantage of the LVM protocol.

In the case of a three edge connectivity network, the MCs construction was formulated as an integer linear program (ILP). In other cases where the network might have nodes with less than three-edge connectivity, a linear-time algorithm to compute the minimum number of needed MCs and MPs was presented [38].

It has been shown in [38] that the monitoring cycle problem with one monitoring location (MC-1) has certainly a solution for three edge-connectivity networks. It also has been proven that a network with L links requires at least $\lceil log_2(|L+1|) \rceil$ cycles for unambiguous fault localization with an arbitrary monitoring location. The worst case however involves $O(L^2)$.

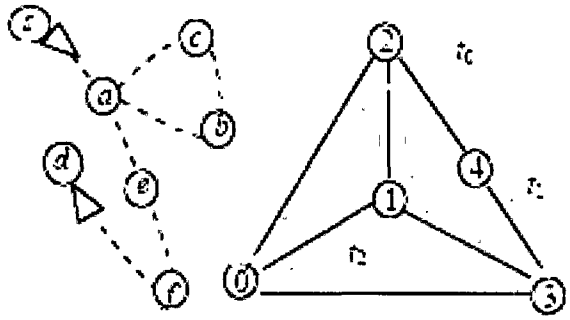
Monitoring with Paths and Cycles (MPC) in a given network with a predetermined number of monitoring locations consists of finding a set of monitoring cycles and a set of monitoring paths such that any single failure can be identified with a unique combination of failed cycles and paths. The procedure starts by performing a network transformation that would convert the MPC problem in the original network with a number of monitoring locations into an MC-1 problem in the transformed network with only one monitoring location [38]. The major drawback of this technique is that the transformation is a complex process.

A fault detection scheme able to localize single failures in an arbitrary network topology has also been proposed [38]. An $O(L)$ (where L is the number of links) algorithm has been developed to calculate the minimum number of monitoring locations.

Although the cycle and path based technique is better than the cycle-based approach, it still has the weakness of requiring supervisory channels assignment when compared with the LVM protocol.

2.4.9 Monitoring-Trail based Fault Localization Technique

The monitoring-Trail (m-trail) technique differs from the m-cycle method by removing the cycle constraint, aiming to minimize the total monitoring cost. An m-trail can cross a node multiple times but no link more than once. Figure 2.11 shows an m-trail based solution for link failure localization in a 4 nodes network. Only three m-trails are needed to detect any faulty link in the network. In other words, only three monitors are needed while four are required in the case of an m-cycle based solution in Figure 2.9. This reduction of the monitoring cost is important in terms of both the hardware cost and the bandwidth cost since only three supervisory wavelengths are to be set up.



Link	t_2	t_1	t_0	Decimal
(0,1)	1	0	1	5
(0,2)	1	1	1	7
(0,3)	1	0	0	4
(1,2)	0	1	1	3
(1,3)	1	1	0	6
(2,4)	0	0	1	1
(3,4)	0	1	0	2

(a) m-trail (b) An m-trail solution (c) Alarm code table

FIGURE 2.11: m-trail based fast link failure localization [36]

An Integer Linear Program (ILP) for m-trail design was formulated in [36]. The objective being to minimize the monitoring cost. The monitoring cost involves both the monitor cost and the bandwidth cost (i.e. supervisory wavelengths). The ILP consists of 19 constraints. It did good when tested with SmallNet topology (10 nodes and 22 links) with the following results as stated in [36]:

- Solution time=761.84 sec
- Gap to optimality=1.43%
- Number of m-trails=6
- Monitoring cost=70

Nevertheless, It took more than three hours for ARPA2 topology (21 nodes and 25 links) with the following results as stated in [36]:

- Solution time=11234.86 sec
- Gap to optimality=21.21%

- Number of m-trails=8
- Monitoring cost=99

Although, an m-trail solution has the capacity to localize any faulty link even in the presence of segments in the network topology (This was a constraint on the m-cycle approach where additional link-based monitors might be necessary), its performance might be a problem when it comes to big networks with general topologies.

A novel algorithm is developed in [39] for m-trail design in general topologies claimed to be much butter in terms of performance than the ILP solution proposed in [36]. The computation time of this new algorithm is supposed to be much less than the ILP solution.

It has been proven in [39] that a ring network of more than 4 nodes needs $\lceil |E|/2 \rceil$ m-trails for unambiguous fault localization where $|E|$ is the number of links in the ring network. Also, it has been proven that in any two-connected network, the UFL requirement can always be achieved with no more than $\lceil |E|/2 \rceil$ m-trails.

2.4.10 Limited-Perimeter Vector Matching Fault Localization Protocol

The Limited-Perimeter Vector Matching (LVM) Protocol is a fault localization protocol that was constructed based on the principles of distributed control and management mechanisms [3]. It is assumed that each fiber is wavelength multiplexed in the network. As a result, the number of lightpaths traversing a link could be as large as the multiplexing degree. It is also assumed that no optical power monitoring or spectrum analysis is supported at the intermediate nodes of established lightpaths. This provides the maximum level of transparency [22]. LVM

protocol can only localize single failures or multiple failures with two conditions [3]:

- The limited-perimeters of the multiple failures are completely non-overlapped (no node or links is shared)
- The failures occur consecutively not simultaneously in a way that the limited-perimeter of the first failure is identified before the second failure occurs

One important feature of the LVM protocol is the fact that it restricts the fault localization area to a small area called limited-perimeter. This feature narrows down the time and space complexities. It has been shown in [3] through analytical performance analysis that the time and space complexities are less than those of the counterpart protocols.

Through a simulation process detailed in [4, 40], the high performance of the LVM protocol has been confirmed. It has been shown that the fault localization time is less sensitive to the change in the diameter of the network as the increase in fault localization time was only 6.7 ms when the distance was increased 500 times. Traffic load has also no or very small effect on the LVM protocol performance [40]. The reader may refer to Appendix A for more details about the LVM protocol.

A new distributed fault localization protocol called Parallel LVM (P-LVM) protocol has been designed to localize multiple simultaneous failures with no constraints [41, 42]. The P-LVM protocol is based on the LVM protocol. In order to handle multi-link failures, P-LVM tries first to separate each failure in a separate limited-perimeter area and then operates in parallel to localize the failures simultaneously in a distributed manner [42].

2.5 Summary

In this chapter we first presented some background about survivability in optical networks. Then we talked briefly about fault detection in SONET, and we went through a number of recently proposed fault localization techniques. The first part presenting the background illustrated the big picture about survivability in optical networks and then pointed out the area of focus of the present thesis.

Fault localization in SONET requires examining overhead at each node, which slows down the fault-localization procedure.

We showed that the LVM protocol has two attractive features that make it among the best proposed protocols for fault localization. The first feature is the fact that it does not require any monitors and the second feature is that the number of exchanged messages is reduced since the LVM protocol operates in a limited-perimeter area rather than the whole network. Another advantage of the LVM protocol over some existing fault localization techniques such as cycle-based, path-based or trail-based approaches is that it does not require the assignment of loop-back supervisory channels. It makes use of the channels of the network to localize the failure in an unambiguous way.

RBSP is an improvement of FLSP. They both require that the destination node is aware of the setup route and the source node. This is not needed for the LVM protocol.

The cycle-based, the path-based and the trail-based techniques require monitors. Compared to a channel-based monitoring scheme requiring one monitor per channel or to a link-based monitoring scheme necessitating one monitor per link, the proposed approaches are a lot better.

Although the island-based technique looks efficient since it operates in small areas of the network called islands, it still has the inconvenience of predefined islands

during the initial network design, and it lacks capability of explicitly locating a failure.

Although the FLA algorithm is capable of detecting both single and multiple failures efficiently, it is unable to detect failures associated with empty leaves in the tree. Moreover, it is only useful for opaque networks and not for transparent optical networks.

The main drawback of the finite state machine based technique is that its deployment becomes very complex when applied to large and dynamic networks.

Chapter 3

LVM Fault Localization Protocol Emulation (LVM Explorer)

3.1 Introduction

The LVM Explorer is a 3-Tier application that was developed to emulate the LVM protocol in real all-optical networks. The Front-end layer is a GUI interface which represents the entry point of the system to the user. All the software components (Networks, Equipments, Physical Links, Lightpaths, etc) are displayed to the user at this level.

The Back-end layer defines the system business rules and objects described in section 3.2.2. It controls the application's functionality by performing the processing of each function initiated by the clients: The GUI and the Protocol processes.

The data layer consists of storing and retrieving information from a database server. This layer keeps data neutral and independent from application servers or business logic.

In this Chapter we will describe in detail the Front-end Layer and the Back-end layer. The data layer is described in Chapter 4.

3.2 Overall Architecture

3.2.1 Multi-tiered Architecture

The LVM Explorer application is composed of three layers, as shown in Figure 3.1:

1. Font-end layer: It represents the presentation layer of the LVM Explorer. This layer contains the software components that are displayed to the user, and represents the entry point of the system to the user. For the LVM Explorer, the front-end layer is a Java based GUI. The GUI is described in detail in Section 3.3.

2. Back-end layer: This is the layer that holds the system business rules and objects. It is composed of different controller components, each of which has one specific role, and delivers one defined service:

    - LVM Controller/AA Controller: The Controller (both LVM and Authentication and Authorization) is the interface that is provided for the front-end layer. All the clients' requests will be handled by this controller which can be considered as the client's requests dispatcher. It implements all the business logic related to interaction with the database described in 4.2. It also contains all the authentication and authorization business rules that enable it to accept or deny a client access to the system. It is also responsible of building the user profile object when the user is successfully authenticated. The user profile defines what are the functions and resources that should be displayed in the GUI based

on the user's role. The user profile also lists the networks to which the authenticated user has access. The controller is described in detail in Section 3.4.

- Switch service: The switch service is the core of the LVM Explorer system. It implements the LVM protocol algorithm and business rules. It is responsible of communicating with the network elements, namely the switches, through an I/O module and listen to the entire alarms. Once an alarm is notified, the LVM protocol is launched. Each physical switch in the real network is mapped to a switch service which runs an instance of the LVM protocol. The Switch Service is described in detail in Section 4.3.
- Registry Service: This module provides an interface for the system modules, and can only be used internally to access to the LVM Explorer registry. All the services (switch services instances) are saved in the LVM Explorer registry. This interface provides a set of functionalities that enable other modules to add, search, update and remove services from the registry.

3. Data layer: The data layer illustrates the source of the data handled by the system. In our case we have 2 source of data:

   - The Database: It contains the system data information, namely the network topology (nodes and links) and the lightpaths. The data maps the physical network into a virtual network. The traffic data is also set up based on the situation on the real physical network. This mapping is very important since all the process of fault localization is handled in the switch services within the LVM Explorer and not on the real physical switches. The database is described in detail in Section 4.2.
   - The physical network: It contains the physical switches, the lightpaths, the physical links and the alarms. The only need for data coming from

the physical network consists of the alarm notifications following a physical link failure. Once link goes down, all the lightpaths traversing that link are affected and therefore their destination nodes notify alarms specifying the type of alarm and which shelf/slot/subslot/port (destination port) is concerned. Each switch service keeps listening for alarms from its corresponding physical switch. Therefore, all the alarms notified by a physical switch are received by the its corresponding switch service. Once a switch service receives an alarm, the LVM protocol is launched.

The LVM Explorer is a Service-Oriented Architecture (SOA) application where each physical switch in the physical network is mapped into a web service. These web services are called switch services, each of which run an instance of the LVM protocol. Several types of messages are exchanged between switch services during different phases of the protocol, namely the flooding phase, the multicasting phase and the determining phase.

Figure 3.2 shows how the GUI interacts with the controller component, built on an Oracle Application Server, through the JNDI (Java Naming and Directory Interface) technology. The controller interacts with the database through the entity beans using ORM (Object Relational Mapping) technology. The GUI gets all the information needed to draw the networks in a profile object as a result of a successful authentication.

The controller is also used to make the user capable of deploying/undeploying and invoking BPEL processes in the Oracle BPEL process manager. There is a BPEL process for each and every physical switch. When the BPEL process is invoked, it launches the protocol application through the Java Binding Service. Once the protocol is launched, it communicates with the controller through JNDI to get all the information needed to run the protocol, and then listens to alarms coming from the physical network. The listening process makes a switch service (running

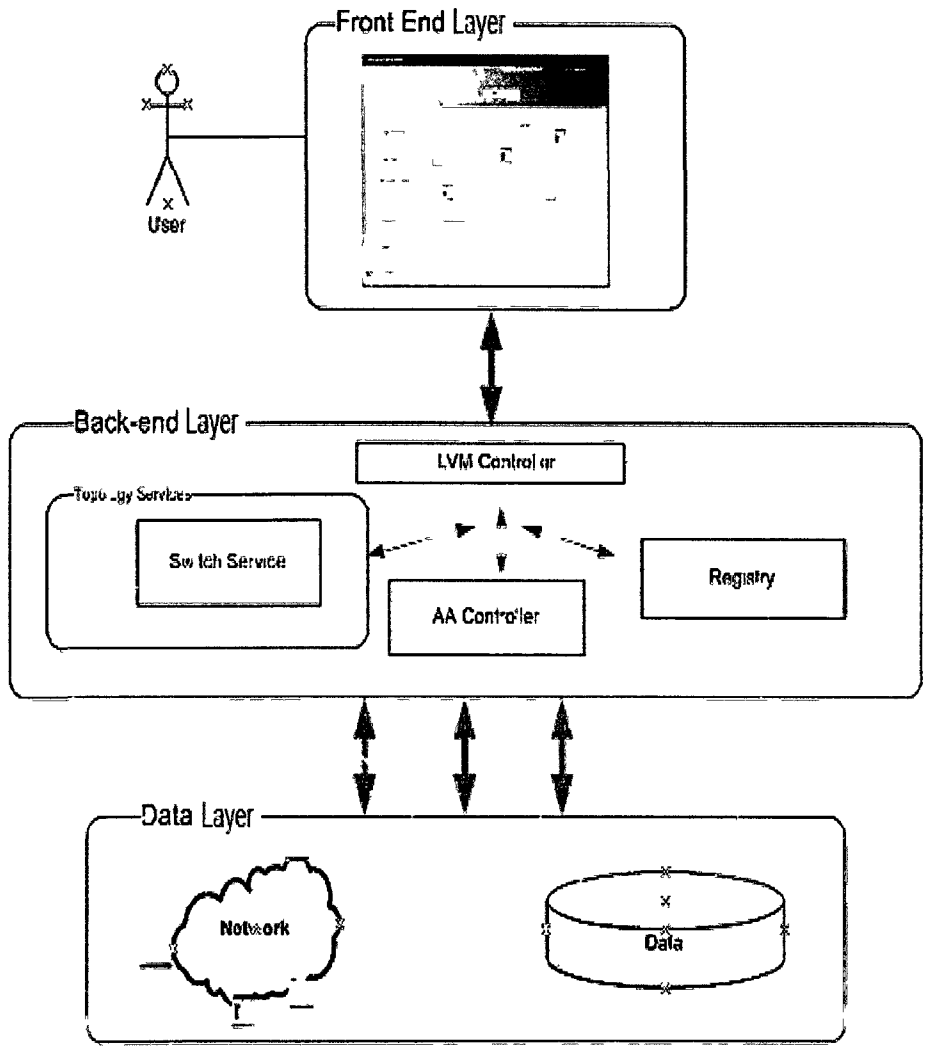


FIGURE 3.1: LVM Explorer High Level Architecture

the protocol) receive an alarm from its corresponding physical switch once a link failure occurs and affects that physical switch.

Each switch service is performing the protocol on behalf of its corresponding physical switch. The message exchange between nodes is performed through a JMS (Java Messaging Service) topic, to which all the switch services are subscribed and

to which they are publishing their messages to be flooded, multicasted or unicasted. The GUI is also a subscriber to the topic in order to read the failure message calculated by the protocol. The GUI updates the network topology appearance following the reception of a failure message in order to distinguish the identified faulty link using the red color.

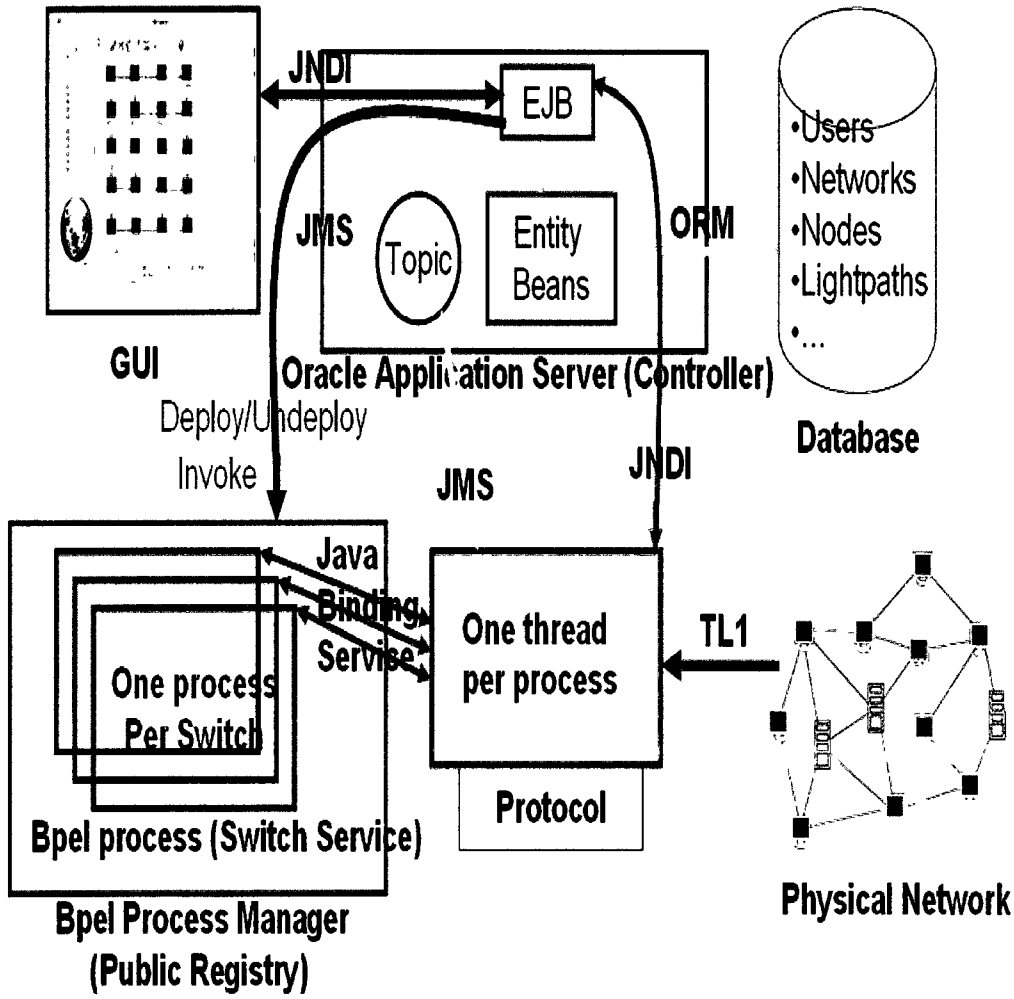


FIGURE 3.2: LVM Explorer SOA Architecture

Figure 3.3 illustrates the block diagram of the LVM Explorer. It shows how the components of the LVM Explorer are linked together. It shows that the controller is the only module interacting with both the database and the registry. It also

demonstrates the correspondence between each switch within the physical network and its switch service within the virtual network. Alarms are sent by the physical switches to their corresponding switch services and the protocol message exchange is performed between the switch services.

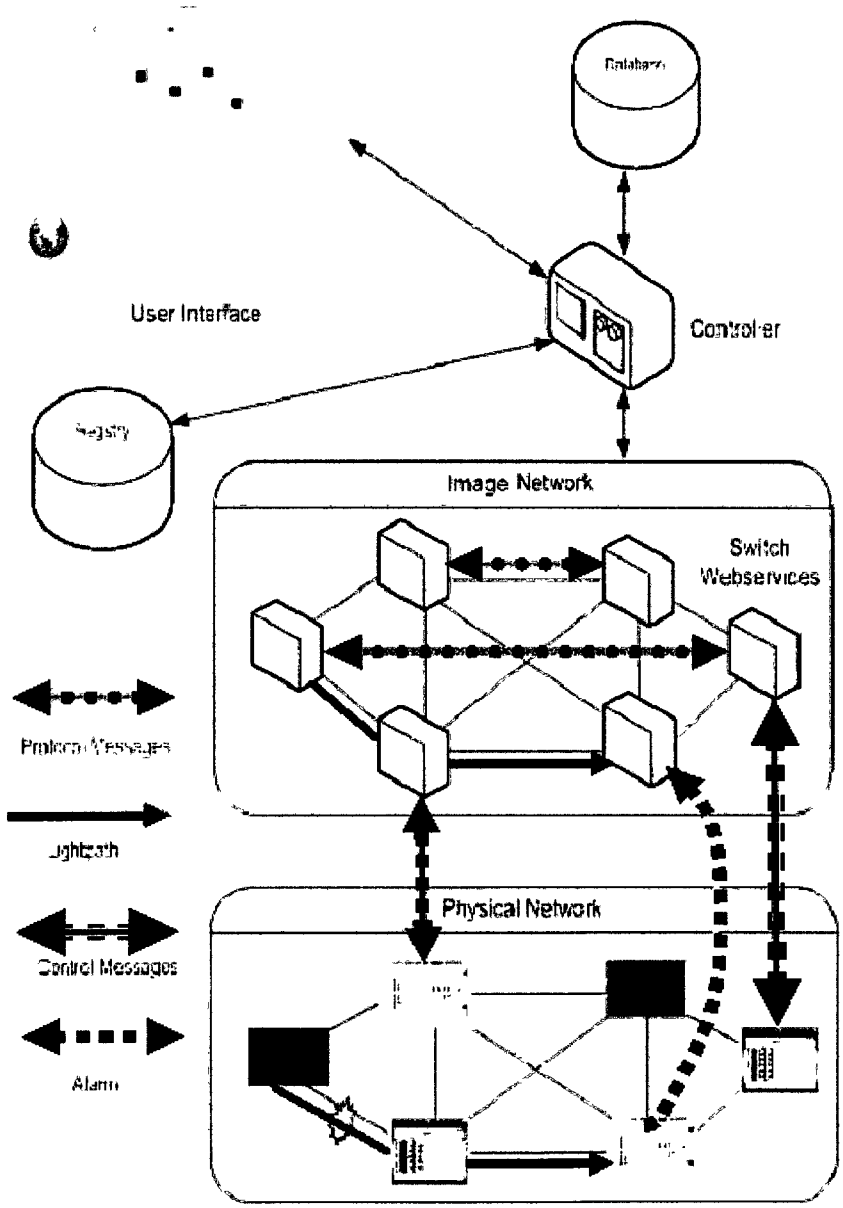


FIGURE 3.3: Block Diagram of LVM Explorer

3.2.2 Business Rules

The business rules related to the implementation of the five phases of the LVM protocol are the following:

1. The protocol is launched when an alarm is received.
2. The receiving node initiates the flooding phase if it is the destination for a lighpath.
3. The sinks (destination nodes for lightpaths) broadcast a flooding message to the whole network.
4. Upon receiving all the flooding messages broadcasted in the network, the sinks elect an executive sink by comparing their own flooding message with the ones they have received.
5. Only Executive sink will start the multicasting step. It first defines a limited-perimeter area and then sends a message to the nodes in that limited-perimeter area.
6. Upon receiving the multicasting message, all the sinks within the limited-perimeter area compute a binary vector based on the received message and their own information. Then, these sinks send their binary vectors to the executive sink.
7. The executive sink calculates the faulty link based on the received binary vectors.
8. If the executive sink is not able to clearly identify the faulty link, it restarts the process from 5 with an extended limited-perimeter area.

3.2.3 Service-Oriented Architecture (SOA)

One can define a service-oriented architecture (SOA) as a group of services (Switch services in our case) that communicate with each other. The process of communication involves either simple data-passing or two or more services coordinating some activities. Intercommunication implies the need for some means of connecting two or more services to each other.

SOA aims to allow users to string together fairly large chunks of functionality to form ad-hoc applications that are built almost entirely from existing software services. As an architecture, it relies on service-orientation as its fundamental design-principle. If a service presents a simple interface that abstracts away its underlying complexity, users can access independent services without knowledge of the service's platform implementation.

The SOA principle is used in our project at the switch service level. Each switch service is implemented as a BPEL process that can be deployed or undeployed on Oracle BPEL Process Manager. The BPEL process implements a business process which occurs to be the LVM protocol launching in our case. The BPEL processes are invoked as web services to start an instance of the protocol code for each switch. The process running the protocol is called a switch service. All the switch services are running exactly the same way making use of the same LVM protocol code through a Java Binding Service. Obviously, these BPEL processes can be deployed in a distributed way making use of the url specified in the database for each node in the network.

3.3 GUI Interface

The GUI interface represents the entry point to the LVM Explorer. It is composed basically of a number of screens helping the user to manage the network data and

to visualize the faulty links detected by the LVM protocol.

When the GUI starts running, an authentication screen is displayed to the user. It asks for a login and a password to be checked against the database content before opening the main GUI screen based on the user profile built upon the database information.

The GUI interface is interacting with the controller in order to get the information needed from the database. This interaction starts by doing a lookup for an EJB (Enterprise Java Bean) in the Oracle Application Server. The EJB contains all the functions needed by the GUI.

In the case the Oracle Application Server is down or the EJB is not deployed, an error is displayed to the user to inform him/her about the reason for the authentication process failure.

3.3.1 User Profile

The user profile is a Java class returned by the authentication function called once the lookup goes well. It contains all the information needed by the GUI to display the network topology.

3.3.1.1 Profile Object

The user profile object contains the user login and the network list to which the connected user has access. A network object contains its name, the list of its nodes and the lists of its lightpaths. A node object defines a number of attributes such as the node name, its IP address, its equipment type and its neighbors. A lightpath object defines the lightpath information such as the source node/port, the destination node/port and the intermediate nodes. The profile object has the following structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://profile.xml.lvm.uottawa.ca"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lvm="http://profile.xml.lvm.uottawa.ca">
<xsd:element name="User" type="lvm:User"/>
<xsd:complexType name="User">
<xsd:attribute name="name" type="xsd:string" use="required"/>
use="required"/>
</xsd:complexType>

<xsd:element name="NetworkList" type="lvm:NetworkList"/>
<xsd:complexType name="NetworkList">
<xsd:sequence>
<xsd:element ref="lvm:Network" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>

<xsd:element name="Profile" type="lvm:Profile"/>
<xsd:complexType name="Profile">
<xsd:sequence>
<xsd:element ref="lvm:User"/>
<xsd:element ref="lvm:NetworkList"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

3.3.1.2 Network Object

A network object specifies the network's name, defines the list of its nodes and the lists of its lightpaths. A network object has the following XML structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://profile.xml.lvm.uottawa.ca"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lvm="http://profile.xml.lvm.uottawa.ca">
<xsd:element name="EquipmentList" type="lvm:EquipmentList"/>
<xsd:complexType name="EquipmentList">
<xsd:sequence>
<xsd:element ref="lvm:Equipment" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="LightpathList" type="lvm:LightpathList"/>
<xsd:complexType name="LightpathList">
<xsd:sequence>
<xsd:element ref="lvm:Lightpath" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="Network" type="lvm:Network"/>
<xsd:complexType name="Network">
<xsd:sequence>
<xsd:element ref="lvm:EquipmentList"/>
<xsd:element ref="lvm:LightpathList"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>
</xsd:schema>
```

3.3.1.3 Node Object

A Node object is identified by a set of attributes such as the id, the name and the IP address. It also contains a list of its neighbors which are the nodes to which

it is connected through a physical link. A Node object has the following XML structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://profile.xml.lvm.uottawa.ca"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lvm="http://profile.xml.lvm.uottawa.ca">
<xsd:element name="Neighbor" type="lvm:Neighbor"/>
<xsd:complexType name="Neighbor">
<xsd:attribute name="id" type="xsd:string" use="required"/>
<xsd:attribute name="link-status" type="xsd:boolean" use="required"/>
</xsd:complexType>
<xsd:element name="NeighborsList" type="lvm:NeighborsList"/>
<xsd:complexType name="NeighborsList">
<xsd:sequence>
<xsd:element ref="lvm:Neighbor" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="Equipment" type="lvm:Equipment"/>
<xsd:complexType name="Equipment">
<xsd:sequence>
<xsd:element ref="lvm:NeighborsList"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string" use="required"/>
<xsd:attribute name="ipaddress" type="xsd:string" use="required"/>
<xsd:attribute name="physical_status" type="xsd:boolean" use="required"/>
<xsd:attribute name="lvm_status" type="xsd:boolean" use="required"/>
<xsd:attribute name="url" type="xsd:anyURI" use="required"/>
<xsd:attribute name="type" type="xsd:string" use="required"/>
<xsd:attribute name="connMode" type="xsd:string" use="required"/>
```

```
<xsd:attribute name="connModeVer" type="xsd:string" use="required"/>
<xsd:attribute name="telnet_port" type="xsd:string" use="required"/>
<xsd:attribute name="login" type="xsd:string" use="required"/>
<xsd:attribute name="password" type="xsd:string" use="required"/>
<xsd:attribute name="posX" type="xsd:int" use="required"/>
<xsd:attribute name="posY" type="xsd:int" use="required"/>
<xsd:attribute name="realNode" type="xsd:boolean" use="required"/>
<xsd:attribute name="lvmProcessStatus" type="xsd:int" use="required"/>
</xsd:complexType>
</xsd:schema>
```

3.3.1.4 Lightpath Object

A lightpath object is identified by a number of attributes such as the id, the source port and the destination port. It also lists all the physical links that define the path taken by the light from the source node to the destination node. A lightpath object has the following XML structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://profile.xml.lvm.uottawa.ca"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lvm="http://profile.xml.lvm.uottawa.ca">
<xsd:element name="PhysicalLink" type="lvm:PhysicalLink"/>
<xsd:complexType name="PhysicalLink">
<xsd:attribute name="id" type="xsd:string"/>
<xsd:attribute name="Node1Name" type="xsd:string"/>
<xsd:attribute name="Interface1Number" type="xsd:string"/>
<xsd:attribute name="Node2Name" type="xsd:string"/>
<xsd:attribute name="Interface2Number" type="xsd:string"/>
<xsd:attribute name="physicalStatus" type="xsd:boolean"/>
```

```
<xsd:attribute name="order" type="xsd:string"/>
</xsd:complexType>
<xsd:element name="Lightpath" type="lvm:Lightpath"/>
<xsd:complexType name="Lightpath">
<xsd:sequence>
<xsd:element ref="lvm:PhysicalLink" minOccurs="1" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string"/>
<xsd:attribute name="srcNodeName" type="xsd:string"/>
<xsd:attribute name="srcSlotNumber" type="xsd:string"/>
<xsd:attribute name="srcPortNumber" type="xsd:string"/>
<xsd:attribute name="destNodeName" type="xsd:string"/>
<xsd:attribute name="destSlotNumber" type="xsd:string"/>
<xsd:attribute name="destPortNumber" type="xsd:string"/>
<xsd:attribute name="hop" type="xsd:string"/>
<xsd:attribute name="wavelength" type="xsd:string"/>
<xsd:attribute name="physicalStatus" type="xsd:boolean"/>
</xsd:complexType>
</xsd:schema>
```

3.3.2 Graphical mapping

Graphical mapping stands for mapping the profile object's networks into graphical interfaces displaying the network visually. All the network components including nodes, physical links and lightpaths are presented visually mapping the physical network topology into a virtual network shown on the screen. The LVM protocol runs on this virtual network rather than on the physical network through the corresponding switch web services. Figure 3.4 shows an example of a 20 nodes mesh network ($4X5$) mapped graphically:

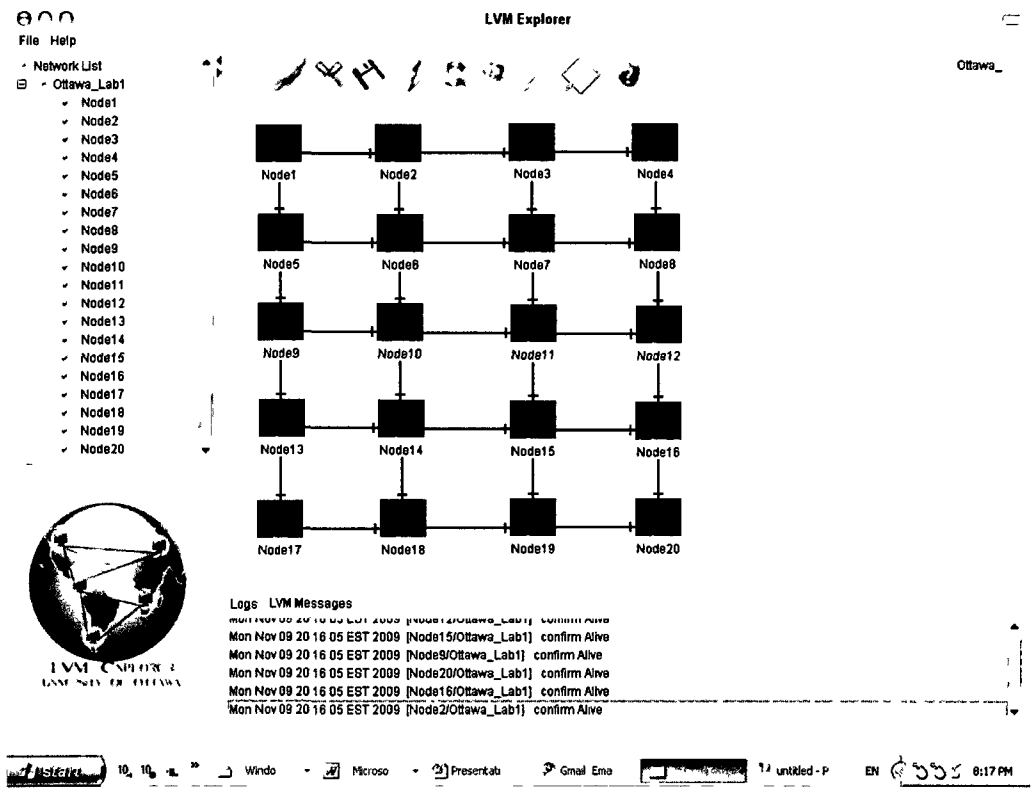


FIGURE 3.4: GUI graphical Mapping of a network

3.3.3 Authentication

The authentication process is triggered by a request of a user to connect to the LVM Explorer. The request is then sent to the controller to check the credentials before giving access to the application. If the authentication fails, the user access is denied and an error message is displayed. Otherwise, the main screen of the LVM Explorer is displayed containing the network list to which the authenticated user has access. Figure 3.5 illustrates the authentication process.

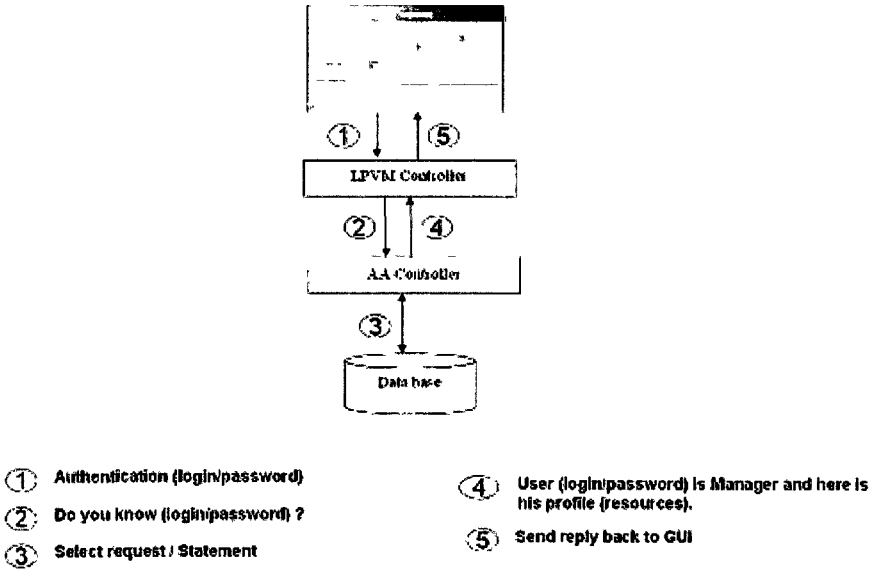


FIGURE 3.5: Authentication of a user

3.3.4 Switch Publishing

When a user creates a new switch, the request is sent to the controller which takes care of persisting data into the database as well as publishing the corresponding BPEL process. The controller creates a new switch in the database and deploys a BPEL process in the Oracle BPEL Process Manager within a transaction context. This concept of transaction insures coherent data in the LVM Explorer. If a failure occurs during the operation of the switch creation, all the transaction is rolled back. Figure 3.6 shows the process of switch publishing.

The switch is published following the url data that is chosen by the user. This feature makes it possible for the user to distribute the switch services among several different machines. Figure 3.7 illustrates how a user can select the url where he/she wishes to deploy the corresponding BPEL process. The BPEL process is defined in 3.4.4.

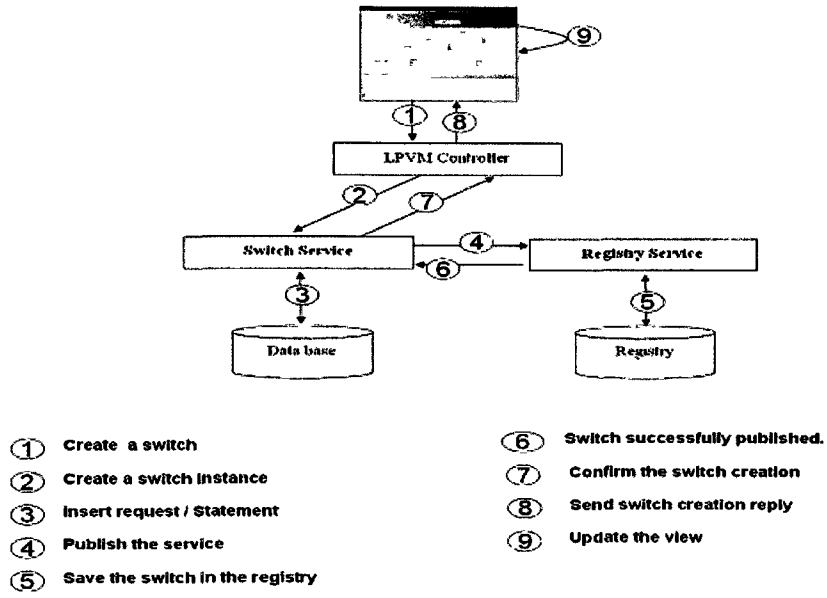


FIGURE 3.6: Publish a switch in the registry

Figure 3.3 illustrates the correspondence between each physical switch within the physical network and its corresponding switch service within the virtual network. The LVM protocol is launched in the virtual network rather than in the physical network, which presents an important feature of the LVM Explorer.

The publishing of a switch requires five files that are archived into a jar file. The jar file is then deployed into the Oracle BPEL Process Manager.

3.3.5 Other Functionalities

The GUI also provides a number of other functionalities such as handling networks, managing users and managing lightpaths. A user can create his/her own networks, create their elements including the switches, the links and the lightpaths. Access to those networks may be provided to other users. The reader may refer to Appendix

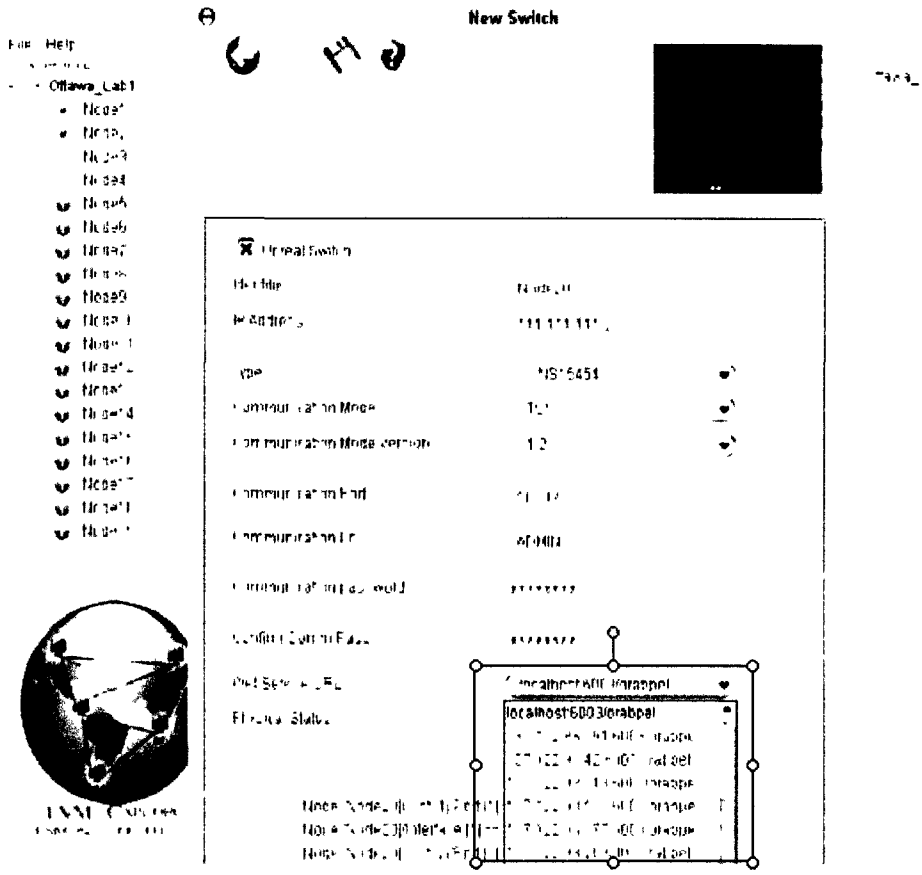


FIGURE 3.7: Distributed Switch Publishing

A for more details about how the networks and their elements are managed in the LVM Explorer.

3.4 Controller

The controller is the module responsible for interaction with the database and the public registry. It represents an interface between the GUI and the database, between the protocol and the database and also between the GUI and the public registry. It implements two EJB (Enterprise Java Bean) classes and an entity bean

for each table or view in the database. One EJB bean contains all the functions needed by the GUI and the other one contains those needed by the protocol.

3.4.1 J2EE Architecture

J2EE multi-tiered applications are generally considered to be three-tiered applications because they are distributed over three different locations:

- Client machines
- The J2EE server machine
- The database or legacy machines at the back end

Figure 3.8 illustrates a general J2EE architecture:

3.4.2 Enterprise Java Beans (EJB)

The controller contains two EJBs, the GUI EJB and the protocol EJB. The GUI EJB implements the functions needed by the GUI to operate, while the protocol EJB implements the functions needed to run the protocol.

The functions required by the GUI are the following:

- Database CRUD functions: Create, Update and Delete (CRUD) functions for each component in the network such as Network, Node, Physical Link and Lightpath.
- Retrieve data from the database. This is the case during the authentication process where the controller returns the profile object of the authenticated user. The profile object is built following a list of queries in the database.

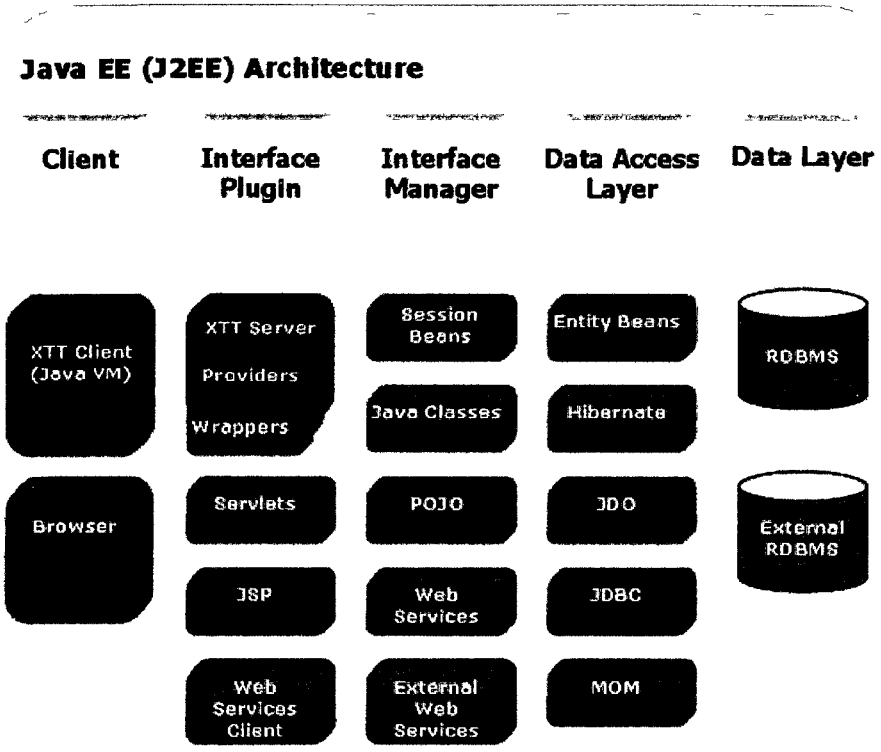


FIGURE 3.8: J2EE Architecture [43]

- Deploy, Undeploy and Invoke BPEL processes. When a new switch is being created, the controller insures the corresponding BPEL process is deployed successfully before committing the changes into the database. In the case a switch is being deleted, the corresponding BPEL process must be undeployed before committing the changes into the database. In order to start the LVM protocol, all the BPEL processes should be invoked in order to make the switch services listening for alarms from the physical network.

The functions needed by the protocol are all related to retrieving data from the database. When started, each switch service running the protocol code has to first get all the information required by the protocol from the database. The

protocol needs all the information about the node, the network and all its nodes and lightpaths. Each node in the network object should also comprise its neighbors list since this information is needed to build the limited-perimeter area at the beginning of the multicasting phase. A list of queries are sent to the database through the controller to get all the needed information.

3.4.3 EJB Entity Beans

An entity bean is an EJB enterprise bean component that manages persistent data, performs complex business logic, potentially uses several dependent Java objects, and can be uniquely identified by a primary key.

In our project, the defined Entity beans implement the interface between the Controller and the database. Each table or view in the database is mapped in the controller by an entity bean implementing all its CRUD functions. Persistence is handled through a persistence framework called TopLink.

3.4.3.1 Data Source

Connections to the database are handled by a unique data source defined once for all the entity beans. The management of the connections to the database are handled by this data source. The connection management is very important since the LVM Explorer is a multi-user application.

3.4.3.2 Caching

For performance reasons, TopLink handles caching within the Entity Beans in order to make the interactions with the database optimized. This insures that data is cached in the memory for future use. Access to data in the memory is

of course a lot faster than in the database. Through a small configuration, the cached data is refreshed whenever data is updated in the database.

Caching is performed in an incremental fashion. That is to say that the application retrieves from the database only data that is inquired and then keeps it in the memory for later use.

As seen before, the LVM Explorer starts by an authentication operation. In the case the authentication is successful, a series of queries are sent to the database to get all the data related to the authenticated user in the form of a profile object built to be returned to the GUI. The construction of the profile object allows to load practically all the needed data from the database into the memory which makes the next data retrievals faster.

3.4.4 BPEL Process Handling

A BPEL process is a collection of Web services whose interactions are choreographed in a defined manner. Each service is a participant that performs some type of processing. Services can be highly granular (e.g., calculate a rate) or very large in scope (e.g., process an order).

The controller is also responsible for handling the BPEL processes which start the corresponding switch services. A BPEL process has to be deployed before it can be invoked in order to run the switch service. It is deployed when a switch is created and undeployed when it is deleted. An attribute (URL) in the node table specifies the location where the BPEL process has to be deployed. This makes the application very dynamic and distributed since we can deploy the processes in different machines. The need for the distributed aspect might be sensed for performance reason since each switch service is handling a message listener listening asynchronously for messages.

Figure 3.9 shows the interaction that takes place between the controller and the public registry to deploy, undeploy or invoke a BPEL process.

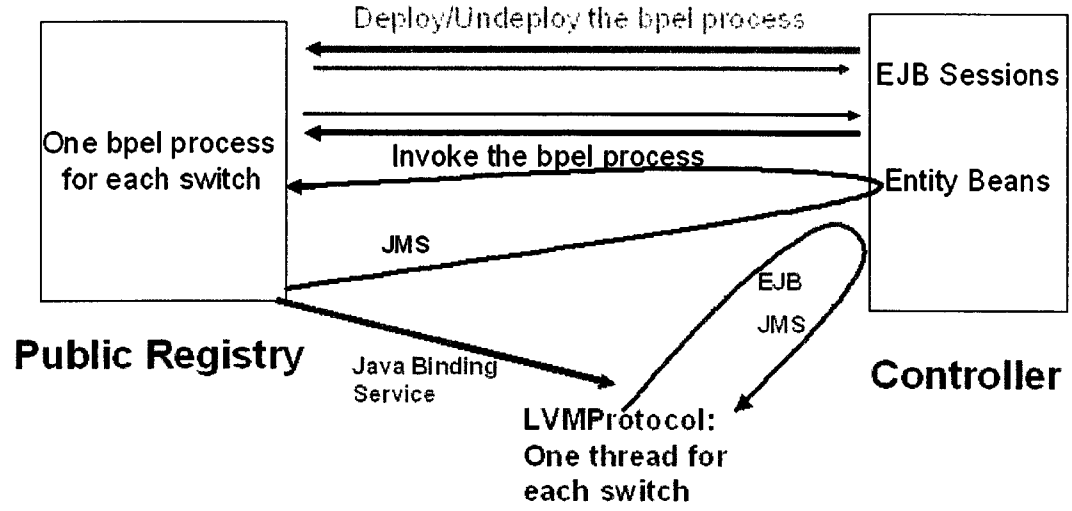


FIGURE 3.9: Deploy, undeploy and invoke a BPEL process

3.5 Summary

In this chapter we presented the overall architecture of the LVM Explorer, we went through the LVM Explorer presentation layer and business layer. The presentation layer is presented by the GUI interface and the business layer consists of the controller module. We also provided the reasons why we made certain technical and technological choices.

The use of the SOA architecture is highlighted since it presents an attractive and important feature in the whole system. This architecture enabled us to create a virtual network mapped on the physical network and then to run the protocol on the virtual network rather than on the physical network. This is an important feature of the LVM Explorer which talks to the physical switches only to get the alarms. All the work is done on the software itself.

Chapter 4

Data and Data Exchange in LVM Explorer

4.1 Introduction

This chapter describes in detail the data layer of the LVM Explorer. This layer keeps data neutral and independent from application servers or business logic. It describes the data sources of the LVM Explorer including the database and the physical network. The interaction with the database consists of storing and retrieving information from a database server, while the interaction with the physical network is required to retrieve the alarm notifications in the case of a link failure.

Also, this Chapter explains how the other layers interacts with these data sources and how the message exchange process is handled.

4.2 Database

The database has been designed in order to take into account different equipment that might be part of an all-optical mesh network. Sometimes, equipment vendors do not use the same naming for some specific equipment components. This is the reason why we have decided to define those components in a precise way for the database model to be understood correctly.

The database contains the following tables:

- Network: This table describe the id and the name of a network.
- LVM_User: This table defines the user credentials to access the LVM Explorer.
- LVM_User_in_Network: This table defines a relationship between a network and an LVM user. Each LVM user can have access to many networks and a network might be assigned by many users.
- Equipment: This definition table lists the different equipment types of a node.
- Connection_mode: This definition table lists all the supported connection modes.
- Node: A record in the Node table refers to a physical switch. It lists all its attributes (name, IP address, equipment type, etc). A node belongs to one and only one network. A network can have many nodes.
- Node_in_Connection_Mode: This table defines a relationship between a node and a connection mode. A node has at least one connection mode. A connection can be assigned to many nodes. The connection specific data , namely the login and the password, is part of this relationship table.

- WDM_Interface: This table defines the interfaces of each node. A node should have at least one interface while an interface belongs to one and only one node.

- Shelf: This table defines the shelves of a node if there are any. A node can have 0 to many shelves while a shelf belongs to one and only one node.

- Slot: This table defines the slots of a node or of a shelf. A node should have at least one slot. A slot belongs to a node either directly or indirectly through a shelf. A shelf, if any, has zero to many slots. A node has zero to many slots.

- Sub_Slot: This table defines the subslots of a slot. A slot can have zero to many subslots. A subslot belongs to one and only one slot.

- Port: This table defines the ports of a slot or of a subslot. A slot should have at least one port. A port belongs to a slot either directly or indirectly through a subslot. A subslot, if any, has zero to many ports. A slot has zero to many ports. Moreover, an interface has many ports and a port belongs to one and only one interface.

- Physical_Link: This table defines the physical connections between two nodes through two of their interfaces. An interface could be the extremity of one and only one physical link. A physical link can not be defined between two interfaces of the same node.

- Lightpath: This table defines the lightpaths of a network. Each lightpath has a source port and a destination port. No port can be referenced more than one in lightpath table, either as a source port or as a destination port.

- Physical_Link_in_Lightpath: This table defines the association between the lightpath table and the physical link table. Each lightpath passes across a certain number of physical links. A physical link can hold more than one lightpath.

4.2.1 Definitions

A node is defined as an entity holding a number of shelves, slots and interfaces. Each shelf contains a number of slots and each slot is a set of subslots and ports. Each subslot has a number of ports and each port is linked to an interface.

The link between each port and its interface insures the WDM (Wavelength Division Multiplexing) concept in the database model.

Figure 4.1 illustrates the following concepts:

- Interface
- Slot
- Port

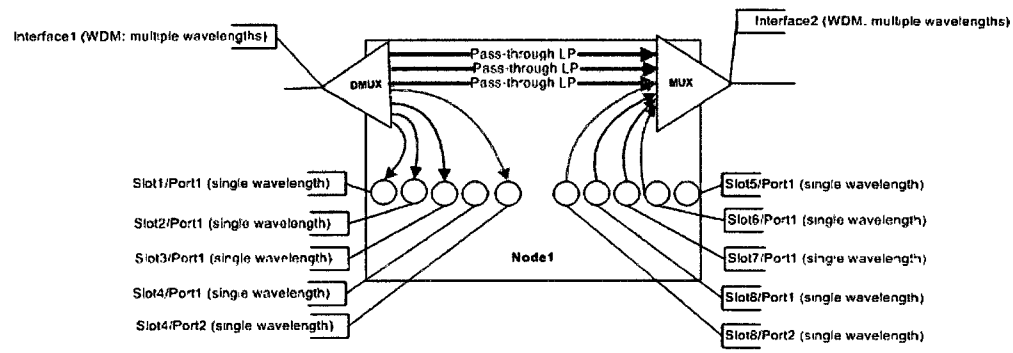


FIGURE 4.1: Illustration of the Interface, Slot and Port Concepts

4.2.1.1 Interface

An interface is a multiple wavelength socket into which a connector can be plugged connecting two switch in the network by means of a fiber. Within the switch, it is directly linked to Multiplexer/Demultiplexer which makes it possible for several

lightpaths to share one fiber. A fiber is linking two switches through two of their interfaces.

4.2.1.2 Shelf

A shelf is a support that consists of a horizontal surface for holding slots. A switch can contain a number of shelves. Each shelf has a unique identifier within each switch. We designed our database model making a zero-to-many relationship between the entity "Node" and the entity "Shelf".

4.2.1.3 Slot

A slot is a piece holding either subslots or sockets for ports. It can be part of either the node directly or a shelf within the node.

4.2.1.4 Subslot

A subslot is a smaller slot which is useful in case of need of more ports. Instead of having ports directly placed in a slot, we can partition the slot into a number of subslots each of which will hold a number of ports.

4.2.1.5 Port

A port is a single wavelength socket into which a connector can be plugged connecting an end user to the network. It is the input/output point for a single wavelength signal that is linked to an end user. A lightpath has a source port and a destination port. The source port is the port receiving the signal from the sender while the destination port is the port transmitting the signal to the receiver. A port is not able to manage more than one lightpath.

4.2.1.6 Physical Link

A physical link represents the fiber linking a node A to a node B. The fiber is connected to node A and node B through one of their interfaces. Figure 4.2 illustrates the concept in a small mesh all-optical network example.

4.2.1.7 Lightpath

A lightpath is a logical link between two nodes. It starts from a source port (port belonging to the source node) and ends at a destination port (port belonging to the destination node), passing through a set of interfaces belonging to the intermediate nodes it crosses. Figure 4.2 illustrates the concept in a small mesh all-optical network example.

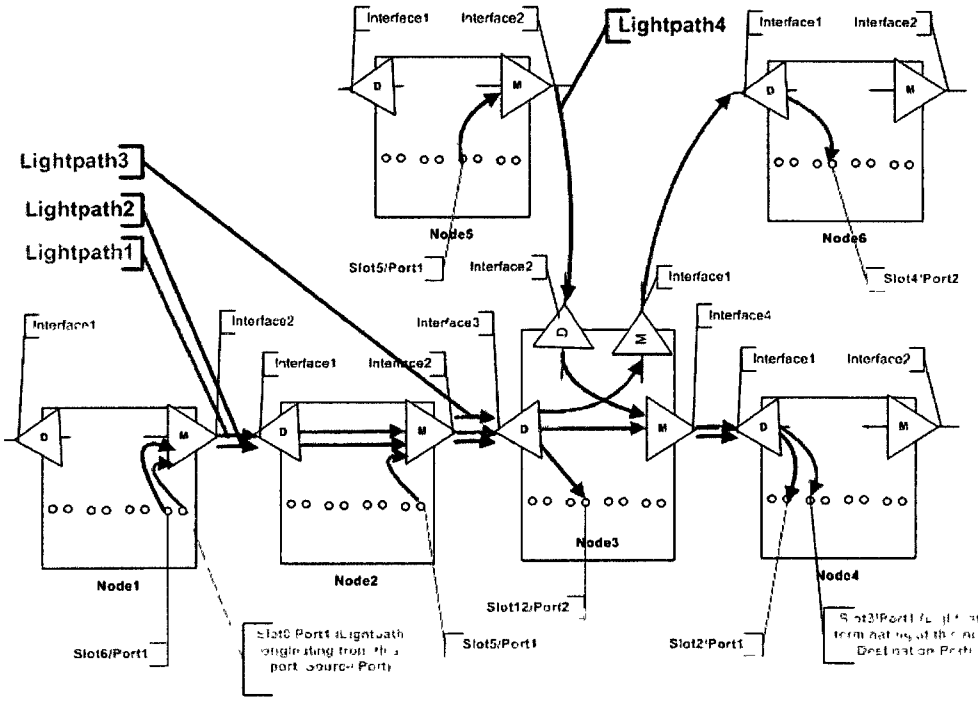


FIGURE 4.2: Illustration of the Lightpath and the Physical Link Concepts

4.2.2 Database Model

Figure 4.3 presents the database model adopted in the LVM Explorer.

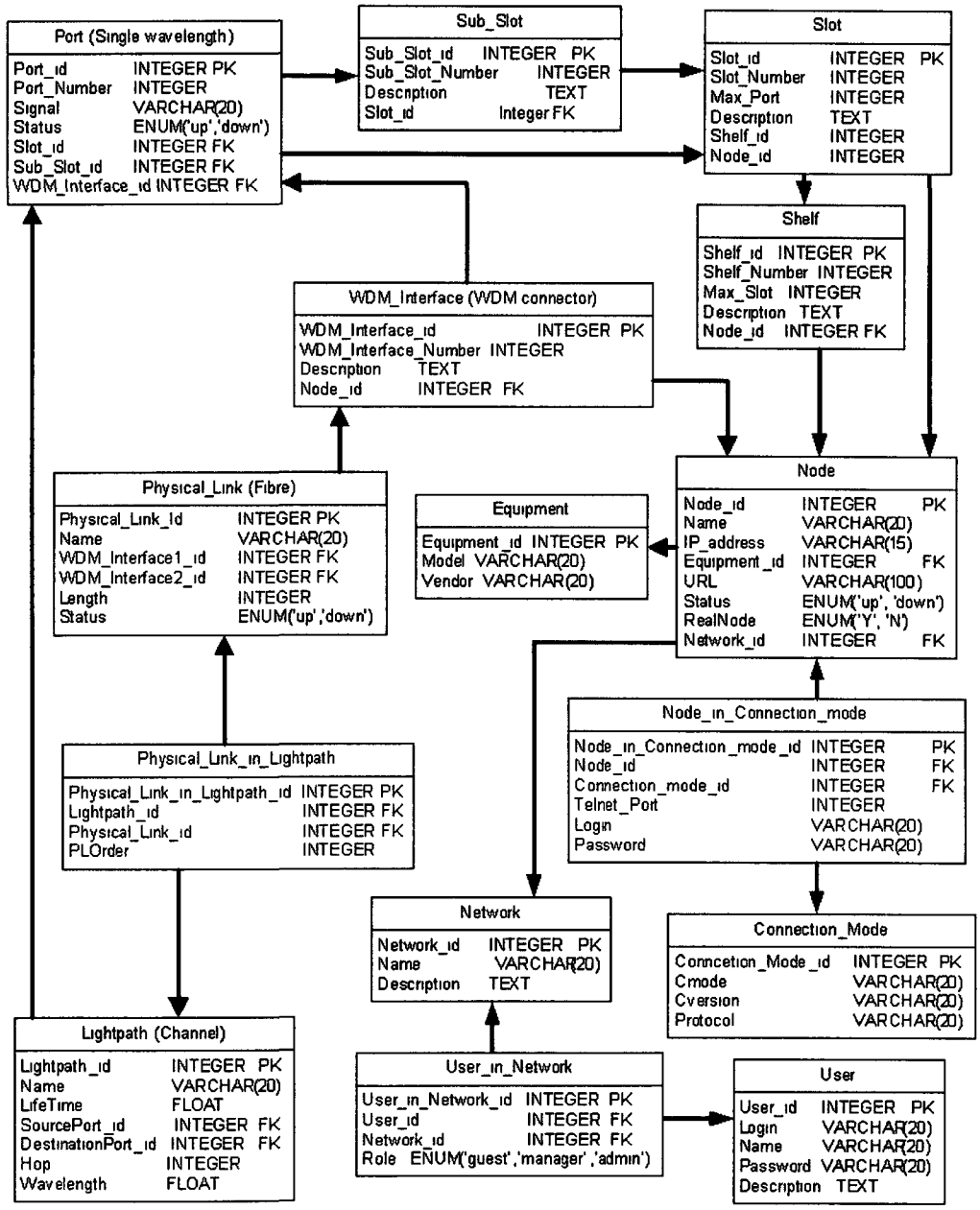


FIGURE 4.3: Database Model Diagram

4.3 Switch Service

When the user creates a node in the GUI, a request is sent to the controller in order to persist the node data in the database and also deploy the BPEL process for that specific node.

Each physical switch is mapped by a switch service process which runs the protocol. The switch service is a thread launched when the corresponding BPEL process is invoked. The thread keep running non-stop listening for alarms through Meta-TL1 from the physical switch or listening for messages (flooding messages for instance) from other nodes through the message listener.

The switch services run the LVM protocol based on a number of business rules mapping the protocol specifications for each phase.

Figure 4.4 illustrates the mapping between physical switches and switch services within the LVM Explorer.

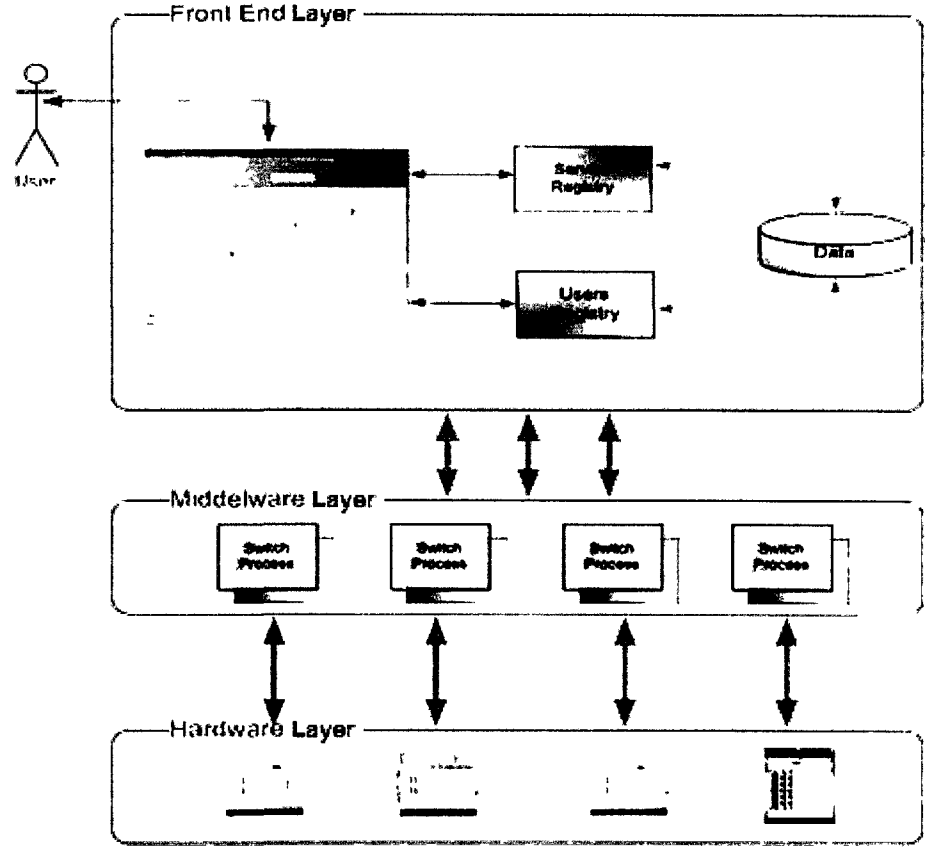


FIGURE 4.4: Switch Services vs Physical Switches

4.3.1 Protocol launching

Figure 4.5 illustrates how the protocol is launched. It shows how the alarms are notified to the switch service listening for alarms. Before starting to listen for alarms, the switch service call the protocol EJB beans through the controller module in order to get all the data it needs from the database; namely, the network data, the lightpath list data and the node data.

The lightpath data object structure is the same as the one used at the GUI level.

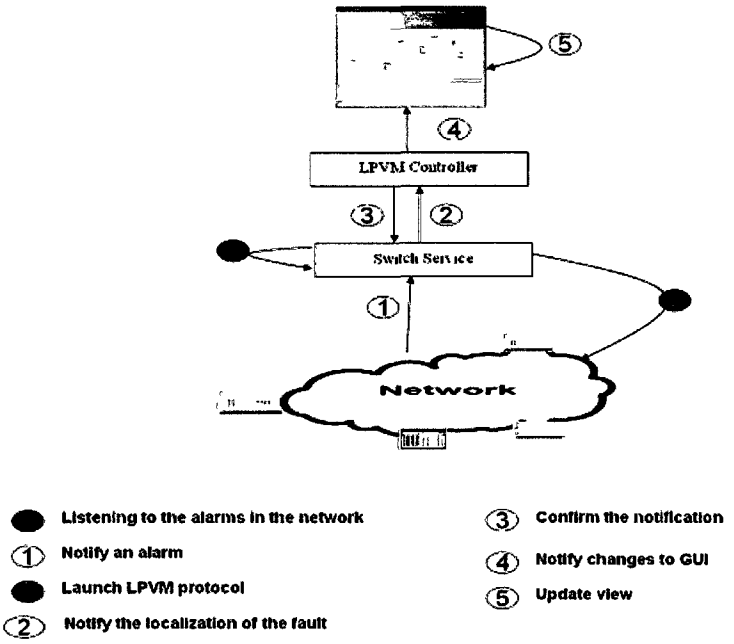


FIGURE 4.5: Launch LVM Protocol

However, the node data object is slightly different from the one defined for the GUI, it has the following XML structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://profile.xml.Meta.uottawa.ca"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:lvm="http://profile.xml.lvm.uottawa.ca">
<xsd:element name="CommunicationProtocol"
       type="lvm:CommunicationProtocol"/>
<xsd:complexType name="Neighbor">
<xsd:attribute name="protocolName" type="xsd:string" use="required"/>
<xsd:attribute name="protocolVersion" type="xsd:string" use="required"/>
<xsd:attribute name="protocolPort" type="xsd:string" use="required"/>
<xsd:attribute name="login" type="xsd:string" use="required"/>
```

```
<xsd:attribute name="password" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element name="CommunicationProtocolsList"
                        type="lvm:CommunicationProtocolsList"/>
<xsd:complexType name="CommunicationProtocolsList">
<xsd:sequence>
<xsd:element ref="lvm:CommunicationProtocol"
                    minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="Host" type="lvm:Host"/>
<xsd:complexType name="Host">
<xsd:attribute name="id" type="xsd:string" use="required"/>
<xsd:attribute name="url" type="xsd:string" use="required"/>
<xsd:attribute name="status" type="xsd:int" use="required"/>
<xsd:attribute name="login" type="xsd:string" use="required"/>
<xsd:attribute name="password" type="xsd:string" use="required"/>
<xsd:attribute name="domain" type="xsd:string" use="required"/>
<xsd:attribute name="domainPassword" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element name="Neighbor" type="lvm:Neighbor"/>
<xsd:complexType name="Neighbor">
<xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element name="NeighborsList" type="lvm:NeighborsList"/>
<xsd:complexType name="NeighborsList">
<xsd:sequence>
<xsd:element ref="lvm:Neighbor" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
```

```
</xsd:complexType>
<xsd:element name="Sub_Slot" type="lvm:Sub_Slot"/>
<xsd:complexType name="Sub_Slot">
<xsd:attribute name="sub_slot_number" type="xsd:string" use="required"/>
<xsd:element ref="lvm:Slot"/>
</xsd:complexType>

<xsd:element name="Slot" type="lvm:Slot"/>
<xsd:complexType name="Slot">
<xsd:attribute name="slot_number" type="xsd:string" use="required"/>
<xsd:element ref="lvm:Shelf"/>
</xsd:complexType>
<xsd:element name="Shelf" type="lvm:Shelf"/>
<xsd:complexType name="Shelf">
<xsd:attribute name="shlef_number" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:element name="Port" type="lvm:Port"/>
<xsd:complexType name="Port">
<xsd:attribute name="port_number" type="xsd:string" use="required"/>
<xsd:attribute name="status" type="xsd:boolean" use="required"/>
<xsd:attribute name="interface_number" type="xsd:string" use="required"/>
<xsd:element ref="lvm:Sub_Slot"/>
<xsd:element ref="lvm:Slot"/>
</xsd:complexType>
<xsd:element name="PortsList" type="lvm:PortsList"/>
<xsd:complexType name="PortsList">
<xsd:sequence>
<xsd:element ref="lvm:Port" minOccurs="0" maxOccurs="unbounded"/>
```

```
</xsd:sequence>
</xsd:complexType>
<xsd:element name="Interface" type="lvm:Interface"/>
<xsd:complexType name="Interface">
<xsd:sequence>
<xsd:element ref="lvm:PortsList"/>
</xsd:sequence>
<xsd:attribute name="interface_number" type="xsd:string" use="required"/>
</xsd:complexType>
<xsd:element name="InterfacesList" type="lvm:InterfacesList"/>
<xsd:complexType name="InterfacesList">
<xsd:sequence>
<xsd:element ref="lvm:Interface" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="LVMNode" type="lvm:LVMNode"/>
<xsd:complexType name="LVMNode">
<xsd:sequence>
<xsd:element ref="lvm:NeighborsList"/>
<xsd:element ref="lvm:CommunicationProtocolsList"/>
<xsd:element ref="lvm:PortsList"/>
<xsd:element ref="lvm:InterfacesList"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:string" use="required"/>
<xsd:attribute name="ipaddress" type="xsd:string" use="required"/>
<xsd:attribute name="status" type="xsd:boolean" use="required"/>
<xsd:attribute name="network_id" type="xsd:int" use="required"/>
<xsd:attribute name="posY" type="xsd:int" use="required"/>
<xsd:attribute name="equipmentName" type="xsd:string" use="required"/>
```

```
<xsd:attribute name="realNode" type="xsd:boolean" use="required"/>
<xsd:element ref="lvm:Host"/>
</xsd:complexType>
</xsd:schema>
```

4.3.2 Meta-TL1

TL1 is an industry recognized protocol used to exchange messages between Network Elements (Switches, Routers) and Operation Systems (OSs) [44]. It provides the network elements with the capability of sending events to Operation Systems (OSs) or Enterprise Management Systems (EMSs). The communication is performed through TCP/IP connections. Although TL1 has been designed as a standard protocol defining the structure of the TL1 messages, it is extensible to incorporate vendor specific commands. This feature explains the variation in the implementation of TL1 by different vendors, which affects the interoperability between their equipment.

Meta-TL1 is a new concept that has been developed in order to solve the interoprability problem [45]. It provides a vendor-independent TL1 Application Programming Interface (API). It implements the protocol TL1 in a way that handles different TL1 formats adopted by distinct equipment vendors. Meta-TL1 consists of three layers:

- The first layer contains a set of configuration files in the XML format. These files handle the equipment and commands configuration;
- The second layer consists of a Java API that offers a set of public methods to execute TL1 commands and get their responses.

- The third Layer consists of a Meta-TL1 Editor, a GUI that uses both the XML configuration files and the MetaTL1 API to manage the equipment and the commands.

The only interaction between the LVM Explorer and the physical network is through Meta-TL1. This interaction takes place so that the switch service knows about the alarms notified by the corresponding physical switch following a link failure. Figure 4.6 illustrates the commands/answers exchanged between the physical switch and the corresponding switch service.

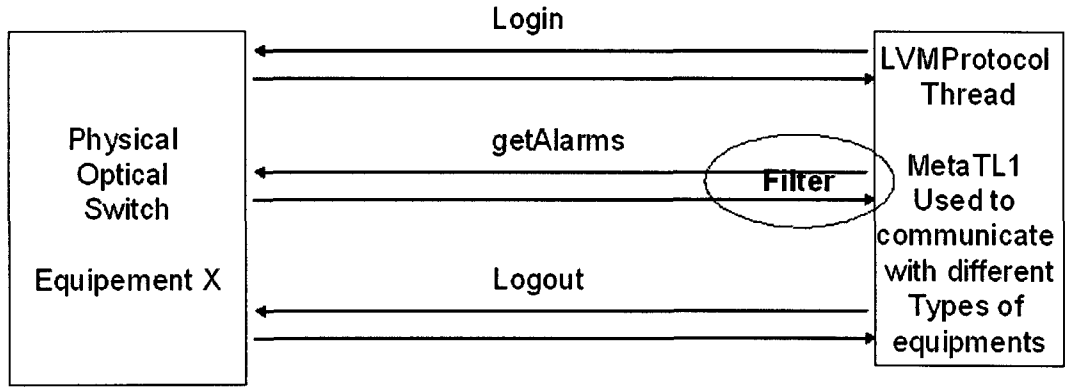


FIGURE 4.6: Communication with physical switches to get alarms

In the case one of the switch services is not able to connect to the corresponding physical switch through TL1, this node send a message to all the other switch services in order to stop the protocol. This would change the status of the nodes on the GUI as described in Appendix B. This tells the user that something is wrong with the system.

4.3.3 Delta times configuration

The LVM protocol is composed of five phases. Each switch in the network is running the protocol. As a result, synchronization between all the nodes is an

important requirement in order for the protocol to give the expected results. That is to say that the nodes need a way to synchronize between them the start of each phase. No node starts the next phase unless all the other nodes are done with current phase.

The synchronization process is performed through a waiting time to be configured for the following phases:

- detecting
- flooding
- determining

Each node receiving an alarm waits for a delta time in order to receive more alarms if there are any and also to wait for the other nodes to receive their corresponding alarms.

Once the receiving alarm node ends the detecting phase, it starts the flooding phase and then waits for a delta time in order to make sure all the flooding messages are exchanged.

During the determining phase, the executive sink waits for another delta time to make sure all the binary determining messages are received before starting to calculate the faulty link.

Those waiting times are crucial to the protocol since they are responsible for the synchronization between nodes. Without synchronization, the results of the protocol might be biased.

The configuration of delta times is done in a property file that should be present in each machine that will be used to run the protocol. It should be placed into the "j2ee" folder within the oracle application server installation.

The property file consists of a line for each phase in terms of milliseconds as follows:

- *lvm.protocol.detecting.delta.time* = 3000: This means that when a node receives an alarm, it waits 3*seconds* before starting the flooding phase. This is to make sure all the notified alarms reach the network.

- *lvm.protocol.flooding.delta.time* = 4000: This means that once the first flooding massage received, each node should wait 4*seconds* for other flooding messages that might be published by other nodes receiving the alarms a little bit later than the first alarm.

- *lvm.protocol.determining.delta.time* = 3000: This waiting time insures that the determining phase, during which the faulty link is identified, does not start unless all the determining messages are received.

The above delta times values have not been chosen in an optimal way. This is because the focus for the performance evaluation is to see the effect of the network size and the network traffic on the performance of both the LVM protocol and the LVM Explorer. For that matter, these values are kept unchanged for all our tests.

4.4 Communication Process

The communication between switch services is insured through a JMS (Java Message Service) topic to which switch services are subscribed as consumers and/or producers. Each producer will publish a message meant to be sent to specific destinations. Each destination, being a consumer, gets a copy of the message from the topic in an asynchronous way. Once all the consumers get their of the message, it is removed from the JMS topic.

The Java Message Service (JMS) has been designed by Sun Microsystems and several other companies under the Java Community Process. It is the first enterprise messaging API that has received wide industry support. It has been designed

mainly to make it easy to develop business applications that asynchronously send and receive business data and events. JMS supports both messaging models: point-to-point (queuing) and publish-subscribe [46]. The publish-subscribe model is the one adopted in our case.

The communication process is a crucial process in the LVM Explorer application. It is controlling the messages exchanged between the switch services. The communication between the switch services is one of the most important aspect in the LVM protocol. It is required at the beginning to launch the protocol, then through its different phases and finally to broadcast the localized faulty link message.

Figure 4.7 illustrates the concept of producer/consumer (known also as publisher/subscriber) and how they exchange messages through a JMS topic.

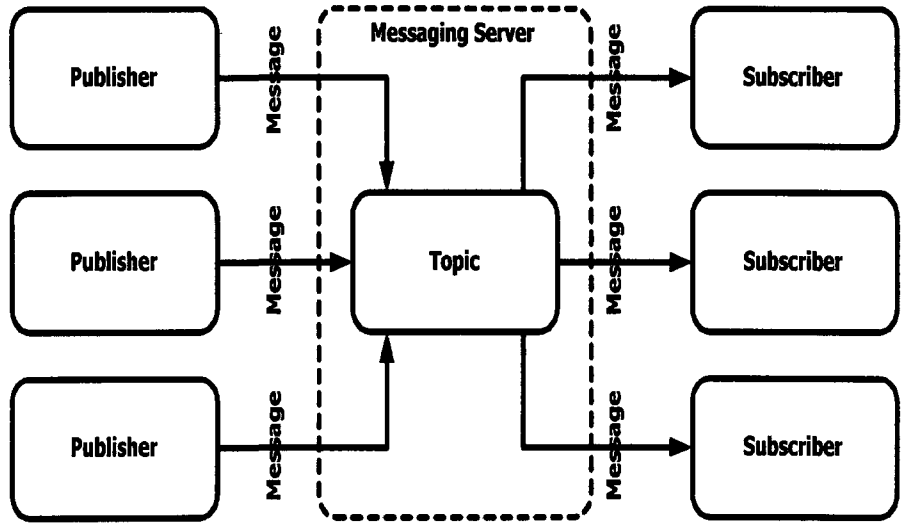


FIGURE 4.7: Publish-Subscribe Based Messaging System [47]

4.5 Summary

In this chapter we talked about the data sources and the data exchange process of the LVM Explorer. The database, the physical network and the configuration files are the data sources for the LVM Explorer.

The database model is designed in a way to be generic. This is important because usually vendors do not use the same technical words when describing their equipment. Therefore, in order to talk the same language, a number of definitions are presented before the database model description.

Also, we have illustrated the communication process that has been adopted for the message exchange during the different phases of the LVM protocol. We have shown that the proposed communication process presents an attractive feature compared to the traditional message exchange process on graphs which is based on the neighboring concept.

Chapter 5

Performance Evaluation

5.1 Introduction

This chapter is dedicated to the provision of the results of a series of tests performed on LVM Explorer using different network topologies with different traffic loads and different network sizes. These results will help in the evaluation of the performance of both the LVM Explorer and the LVM protocol based on a number of metrics related to time needed either by the LVM protocol or by the LVM Explorer to achieve a specific task.

All the tests have been performed on simulated mesh networks with predetermined delta times for the flooding, multicasting and determining phases . We used the following delta time values:

- 3 seconds for the detecting phase
- 4 seconds for the flooding phase
- 3 seconds for the determining phase

Those delta times are used for synchronization purposes. Synchronization in such a distributed system is an important issue. It insures the following phase does not start unless all the nodes are done with the current phase. Whenever the LVM protocol is launched, each switch service that is triggered to start a certain phase would wait the corresponding delta time before starting the next phase. Therefore, the message exchange of each message type (Alarm Message, Flooding Message, Determining Message) is processed synchronously by all the switch services.

Although LVM Explorer takes into account different types of equipment that might be part of the same network through Meta-TL1 module, as seen in detail in the previous chapter, the tests focus on the performance evaluation of the LVM protocol and the LVM Explorer. The alarms are triggered via a simulation system that works from the GUI. Once a link is simulated to be faulty, the GUI identifies the related alarms that would be triggered as a consequence of this failure, based on the network information. The identified alarms are then published to be read by the corresponding switch service.

It is important to mention that the sum of the above delta times ($10 seconds$) would be a minimum time for any test for both the LVM Explorer and for the LVM protocol.

5.2 Metric Measures

The performance evaluation of both the LVM Explorer and the LVM protocol will be based on the following metrics:

- Time needed for the flooding phase to be completed.
- Time needed for the multicasting phase to be completed.
- Time needed for the determining phase to be completed.

- Time required by the LVM protocol code to localize the faulty link.
- Time required by the LVM Explorer to localize the faulty link, to communicate the failure message to the GUI.
- Time required to broadcast the failure message in the network.

For each experiment, we run five tests for which we calculate the metrics above. Then we use the average time of the five experiences along with a 95% confidence interval. We illustrate the 95% confidence interval in the form of $[\mu - d, \mu + d]$ where d represents the distance within which the measured time would fall with a probability of 95%. We use the largest distance between all our tests. μ represents the mean of the five experiments run for each test case.

5.3 Network Size Effect

In order to understand the effect of the network size on the performance of both the LVM protocol and the LVM Explorer, we have performed a number of tests on networks with different sizes and for which we have maintained exactly the same load traffic. We have carried out our tests on 7 mesh networks in the form of grids: $4X3$, $5X4$, $6X5$, $7X6$, $9X7$, $9X9$ and $10X10$ mesh networks.

For all the above mesh network we have maintained a traffic load of 4 lightpaths defined exactly the same way, with the same number of hops and the same path (using same source source port, same destination port and same intermediate interfaces). We have defined the one lightpath with 3 hops and the other three with two hops.

For the tests to bring out the effect of the network size, we have simulated the failure in a similar location for each network. This is to insure the LVM protocol will be running exactly the same way with the same number of notified alarms and

the same number of processed alarms. The same number of processed alarms is guaranteed by having the same number of notified alarms and the same lightpath topology.

All the tests are based on the best case for which the LVM protocol determines the faulty link within the smallest limited-perimeter area, surrounding the executive sink associated lightpath.

5.3.1 Time required by the LVM Explorer

The chart in Figure 5.1 illustrates the time (in ms) required by the LVM Explorer to localize and display the faulty link in the GUI interface. The chart shows that the LVM Explorer takes between 11 and 11.25 seconds to localize and display the failure with a 95% confidence interval $[\mu - 437ms, \mu + 437ms]$. This means that if we run a large number of tests with different sizes between 12 and 100, then 95% of the times, the time needed by the LVM Explorer will be between $\mu - 437ms$ and $\mu + 437ms$.

Figure 5.1 also illustrates that the bigger the network size is, the more time the LVM Explorer requires to localize and display the failure. The difference is however so small that we can even claim that the network size does not affect both the LVM Explorer and the LVM protocol performance. The LVM Explorer requires only $0.2s$ more time for a 100 nodes network over a 12 nodes network.

Due to the values defined for the delta times, no test of the LVM protocol within the LVM Explorer would take less than $10seconds$. Figure 5.1 proves this by showing the chart line above the $10seconds$ line.

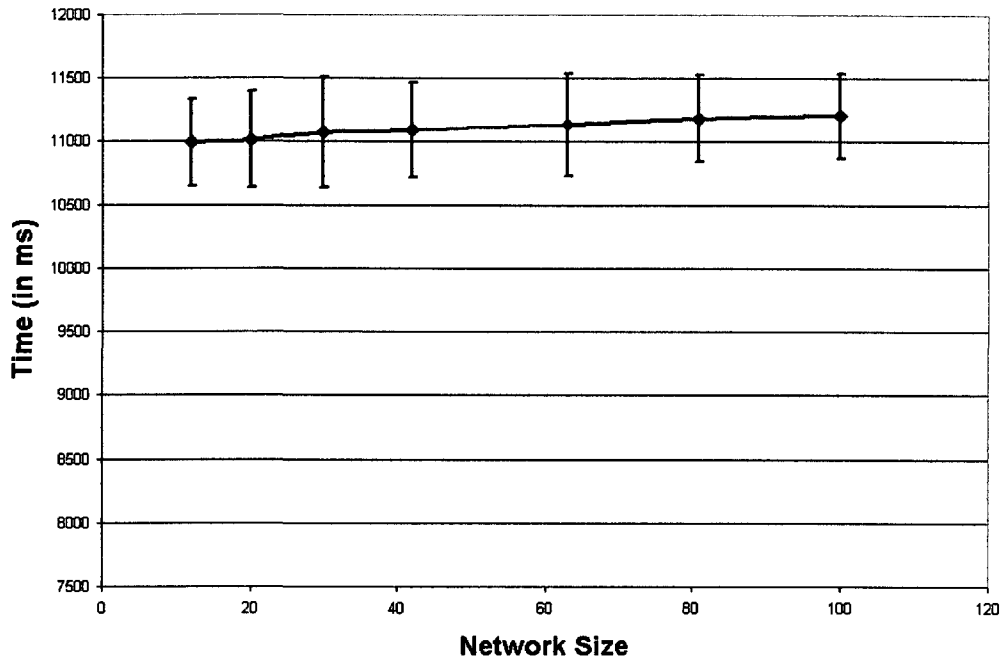


FIGURE 5.1: Time required by the LVM Explorer for networks with different sizes

5.3.2 Time needed by the LVM protocol code

The chart in Figure 5.2 shows the time required by the LVM protocol to localize a faulty link. To measure that time, we record the timestamp at which the first alarm message is received by a switch service on one hand, and the timestamp at which the determining phase ends by detecting the faulty link, and we take the difference between the two timestamps.

We can read from the chart in Figure 5.2 that for all the networks (From the one with 12 nodes up to the one with 100 nodes), the time needed by the LVM protocol to localize the fault is between $10.2 seconds$ and $10.3 seconds$. This means a difference of $0.1 second$. This proves that even though there is a small effect of the network size on the performance of the LVM protocol, it is still very insignificant.

Using the values of the delta times defined above, the time needed by the LVM protocol code would never be less than $10seconds$. The chart line in Figure 5.2 illustrates this remark graphically.

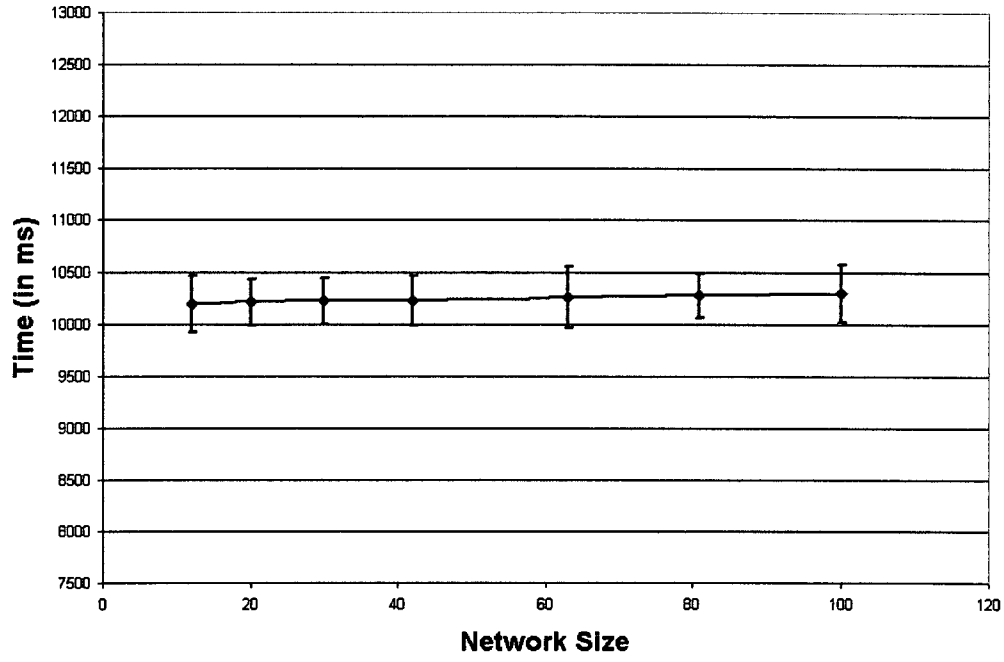


FIGURE 5.2: Time required by the LVM Protocol for networks with different sizes

5.3.3 Time needed by the flooding phase

Figure 5.3 illustrates the time that has been effectively required by the flooding phase for each experience. It does not include the delta time defined above for the flooding phase. It represents the difference between the time when the last flooding message has been received and the time when the first flooding message has been published.

The flooding time varies between $2.2seconds$ and $2.4seconds$. The difference of time needed by the flooding phase for all the performed experiences is about

$0.2seconds$. Although $0.2seconds$ is a small difference compared to the variation of the network size, it still indicates a slight effect of the network size over the flooding time through an increasing trend in Figure 5.3. The 95% confidence interval for the flooding phase is $[\mu - 117ms, \mu + 117ms]$.

Needless to mention that the delta time defined for the flooding phase can not be less than $2.4seconds$ as this is the maximum time needed for the flooding phase to be achieved for a network where the number of nodes does not exceed 100, as illustrated in Figure 5.3. This would suggest a convenient and practical way to define optimal delta time for the flooding phase.

The synchronization performed during the flooding phase helps to identify the right executive sink. All the flooding messages should be exchanged before starting the computation of the executive sink.

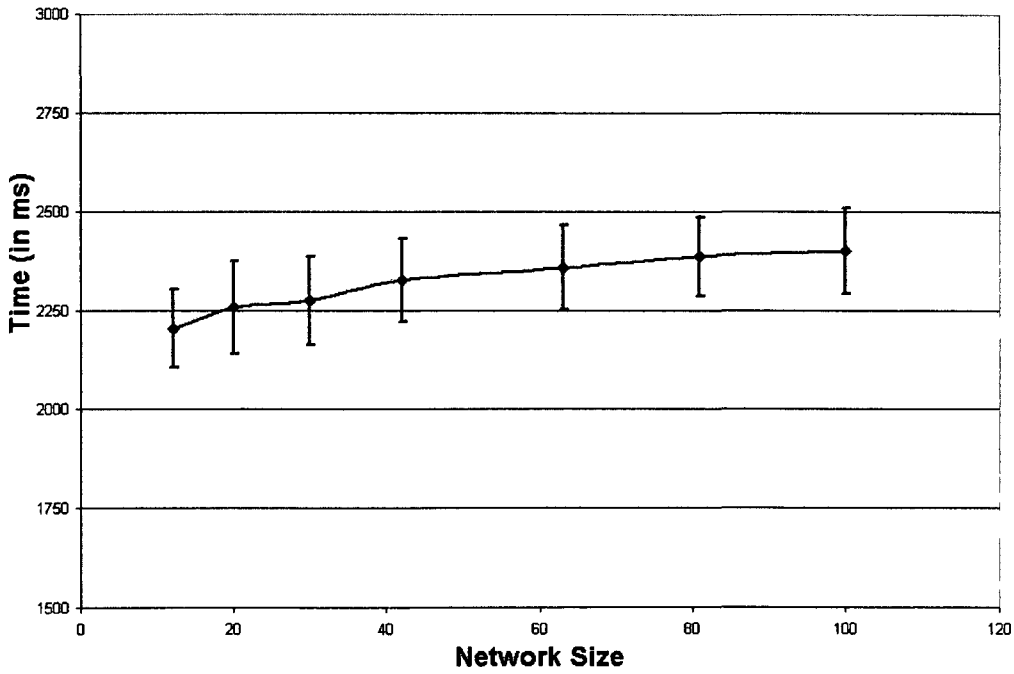


FIGURE 5.3: Time required by the flooding phase for networks with different sizes

5.3.4 Time needed by the multicasting phase

The multicasting phase does not need to be synchronized since it is launched immediately by the node that recognizes itself to be the executive sink. There is only one and only one executive sink for each run of the LVM protocol. Therefore, no need for synchronization at this stage of the protocol. This is why no delta time has been defined for the multicasting phase.

The time in Figure 5.4 represents the difference between the time when the last multicasting message has been received and the time when the multicasting message has been published by the executive sink.

The chart in Figure 5.4 shows that the time needed by the multicasting phase is about $0.2 second$ in all cases. We can see that this time is much smaller compared to the time needed by the flooding phase. This observation finds a logical explanation in the fact that the flooding phase involves all the nodes of the network while the multicasting phase operates on the limited-perimeter area instead of the whole network. Through this comparison between the multicasting phase and the flooding phase, we see clearly the importance of Limited-Perimeter Area feature in the LVM protocol in terms of performance gain. It is worth mentioning that the flooding phase needed more than $10 times$ the time required for the multicasting phase.

Figure 5.4 shows that the network size has practically no effect on the multicasting phase. It also shows that the 95% confidence interval for the multicasting time is $[\mu - 79ms, \mu + 79ms]$.

5.3.5 Time needed by the determining phase

The determining phase corresponds to the answers of the multicasting messages sent from the executive sink to the nodes of a limited-perimeter area. Upon

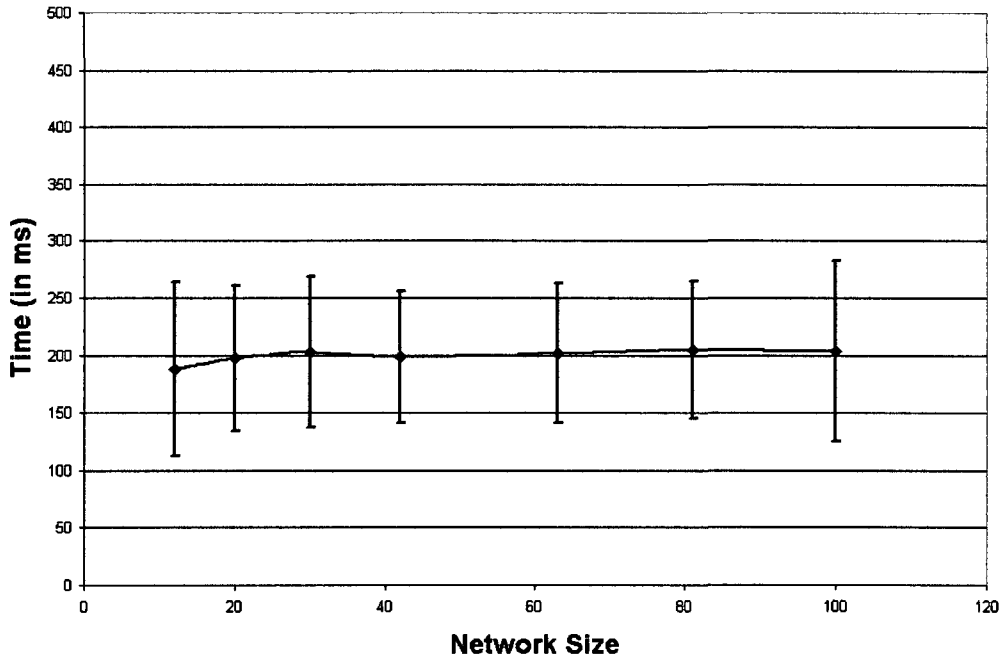


FIGURE 5.4: Time required by the multicasting phase for networks with different sizes

receiving the multicasting message, each node that is a sink computes an Affected Link Vector (ALV) that is part of the determining message to be sent back to the executive sink. Like the multicasting phase, the determining phase is processed within a limited-perimeter area. Therefore, the time required is a lot less than the flooding phase which involves all the nodes of the network.

The time in Figure 5.5 represents the difference between the time when the last determining message has been received by the executive sink and the time when the first determining message has been published by a sink within the limited perimeter area.

The chart in Figure 5.5 shows that the time required for the determining phase is between $50ms$ and $75ms$ in all the experiences. The Limited-Perimeter Area advantage is one more time highlighted through the gain in time compared to

the flooding phase. The 95% confidence interval for the determining phase is $[\mu - 58ms, \mu + 58ms]$.

The delta time defined for the determining phase is obviously not included in the chart. Synchronization in the determining phase is extremely important since it is important for the protocol to wait for all the ALV vectors to be received before starting the computation of the faulty link. Otherwise the protocol may fail in detecting the faulty link because of lack of information seen usually as lack of traffic.

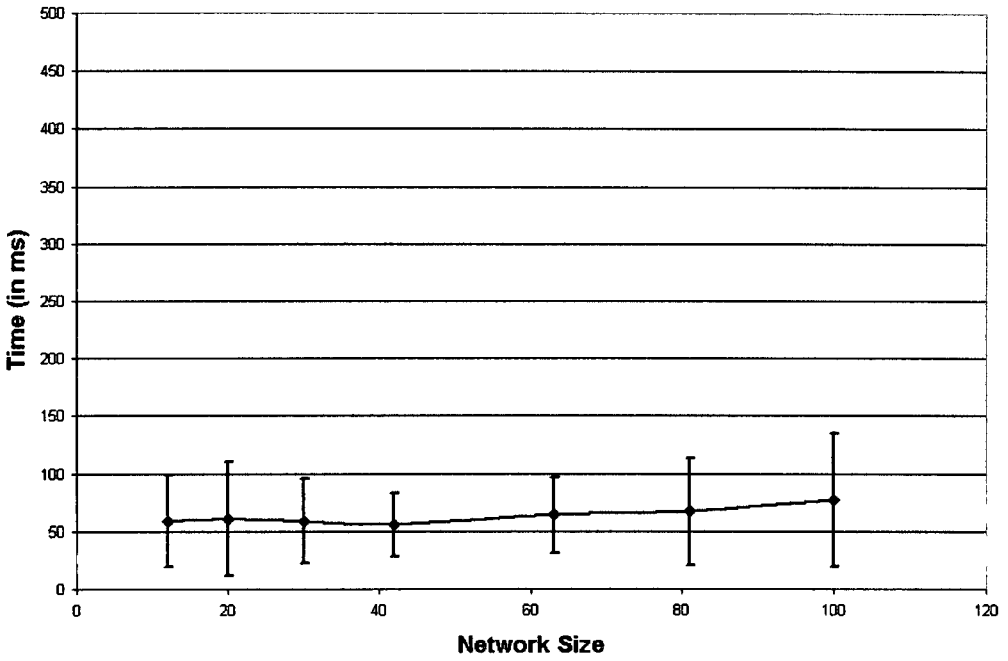


FIGURE 5.5: Time required by the determining phase for networks with different sizes

5.3.6 Time required to broadcast the failure message

The chart in Figure 5.6 shows that the time required for the failure broadcast once the faulty link is localized by the executive sink. It is that same executive sink

which broadcasts the information to all the nodes in the network in order for the restoration process to be started.

The time in Figure 5.6 represents the difference between the time when the failure message has been received by the last node in the network and the time when it has been published by the executive sink. It is less than $0.25second$ in all the experiences. Although all the nodes of the networks are destinations for the failure message, the time required for the broadcast is still small. This is because there is only one published message as it is the case for the multicasting phase, while during the flooding phase many messages are supposed to be exchanged based on the number of alarms that have been triggered. The 95% confidence interval for the failure broadcast is $[\mu - 74ms, \mu + 74ms]$.

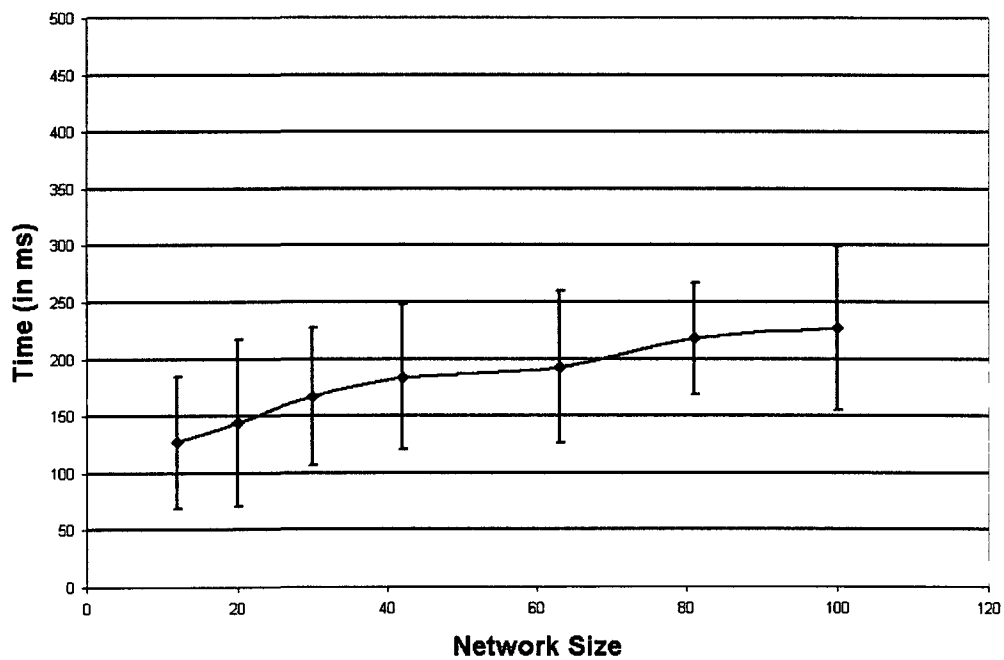


FIGURE 5.6: Time required by the failure message broadcast

5.4 Network Traffic Load Effect

It is clear that a small traffic decreases the probability that the LVM protocol detects the fault. This is due to the fact that LVM protocol makes use of the network lightpaths and their status in order to localize the failure.

The LVM protocol remedy to this issue is to try the protocol with wider perimeters hoping to include more traffic during the new run of the protocol. As a result, the protocol will take more time since it relaunches both multicasting and determining phases on the new perimeter area. However, it is more probable for the protocol to detect the failure during the second run than in the first one.

Another way to handle this issue is to generate more traffic in the network and then run the protocol. The challenge in this solution is the criteria to generate more traffic in the network. This will be a point of future research since it falls outside the scope of our project.

To get an idea on the effect of the network load traffic on the performance of both the LVM protocol and the LVM Explorer, we have performed a number of experiences of five tests each on networks with different traffic loads and for which we have maintained exactly the same size. We have carried out our tests on a $7X6$ mesh network with different numbers of lightpaths: 2, 4, 8, 12, 16, 20, 24 and 28 lightpaths.

For the tests to bring out the effect of the traffic load, we have simulated the failure in exactly the same location for each test. The outcome of the test performed with 2 lightpaths is different from the outcome of the others. With 1 lightpath, the LVM protocol did not have enough information to point out the faulty link. All the other tests have been successful in identifying the faulty link. The successful tests are used to understand the effect of the traffic load on the performance of the LVM protocol and the LVM Explorer.

All the tests are based on the best case for which the LVM protocol determines the faulty link within the smallest limited-perimeter area, surrounding the executive sink associated lightpath.

5.4.1 Not enough traffic

"Not enough traffic" is the answer of the LVM Explorer in the case it finds out that the traffic is not sufficient to identify the faulty link. This is the case when we have tested with 1 lightpath in the network. Although the LVM protocol has been tried with wider limited-perimeter areas up to the whole network, it did not get enough information to point out the faulty link.

5.4.2 Time required by the LVM Explorer

The chart in Figure 5.7 illustrates the time required for the LVM Explorer to localize a faulty link and to display it using a different color in the GUI interface. Obviously, it involves the delta times defined above. Therefore it can never be less than $10 seconds$. The time required by the LVM Explorer when changing the traffic load is about $11 seconds$.

From Figure 5.7, we can say the traffic load has no effect on the performance of the LVM Explorer. The trend is steady and the 95% confidence interval is $[\mu - 506ms, \mu + 506ms]$.

5.4.3 Time needed by the LVM protocol code

The chart in Figure 5.8 shows that traffic load has no effect on the performance of the LVM protocol. This coincides with the conclusion that has been drawn from the chart in Figure 5.7.

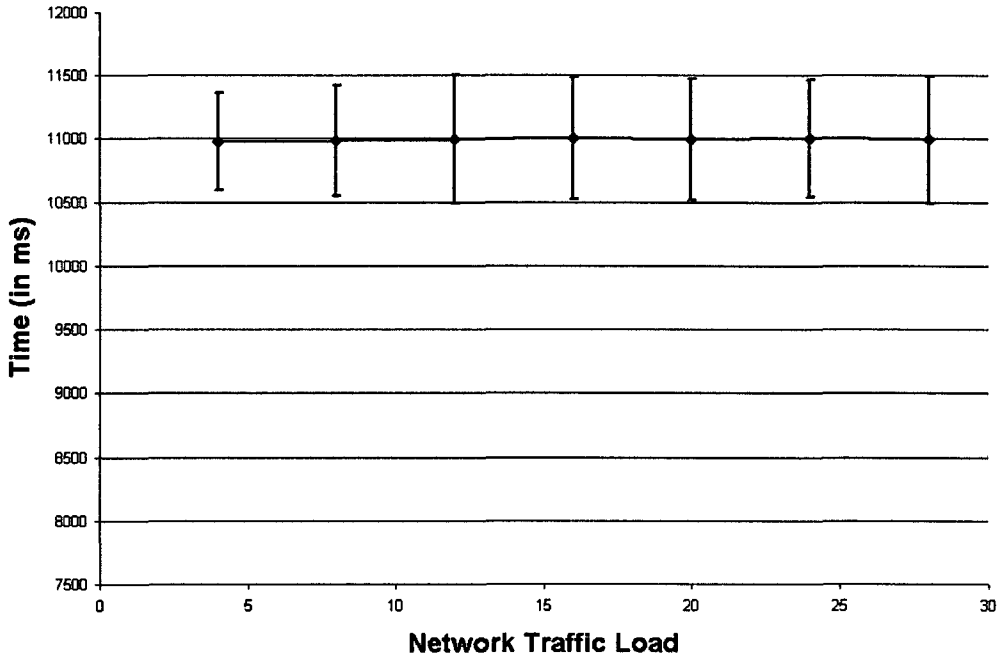


FIGURE 5.7: Time required by the LVM Explorer for networks with different traffic loads

Obviously, the time needed by the LVM protocol code involves the delta times and therefore can not be less than $10seconds$. The chart shows that the time required by the LVM protocol is about $10.25seconds$. The 95% confidence interval for the time needed by the LVM protocol is $[\mu - 202ms, \mu + 202ms]$.

Theoretically, the LVM Protocol performs a number of phases with a number of message exchanging tasks, no matter how much traffic the network is handling. The very small difference when more traffic is generated in the network might be during the multicasting and determining phase. This difference may consist a few more messages to be exchanged, making the effect on the performance of the whole process insignificant.

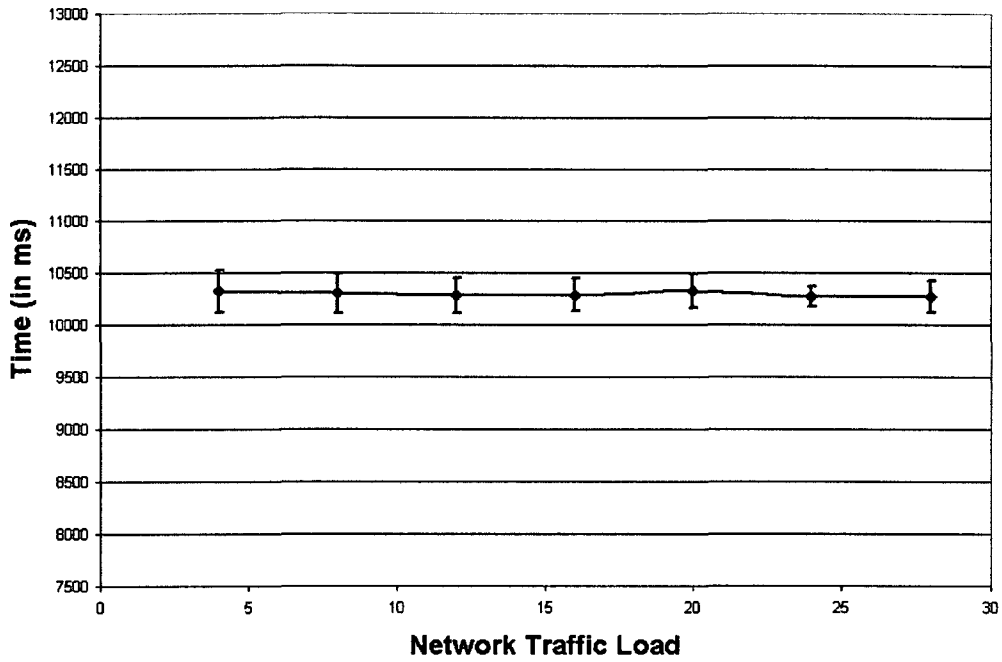


FIGURE 5.8: Time required by the LVM Protocol for networks with different traffic loads

5.4.4 Time needed by the flooding phase

The time in Figure 5.9 represents the difference between the time when the last flooding message has been received and the time when the first flooding message has been published. It does not include the delta time defined for the flooding phase.

Theoretically, the traffic load should has absolutely no effect on the performance of the flooding phase. No matter how much traffic is handled by the network, exactly the same operations and the same number of message exchanges are performed during the flooding phase.

The claim above has been strengthened by the chart in Figure 5.9 which illustrates a steady trend of the time required by the flooding phase for the same network with

different traffic loads going from 4 lightpaths up to 28 lightpaths. The flooding phase has taken about $2.3second$ for all the experiences. The 95% confidence interval for the flooding phase is $[\mu - 151ms, \mu + 151ms]$.

The tests carried out to measure the time needed by the flooding phase are a suitable tool to tune up the delta time for this phase. From Figure 5.9, we can say that $2.5seconds$ might be an optimal value for the flooding delta time instead of the value used for the tests which is $4seconds$.

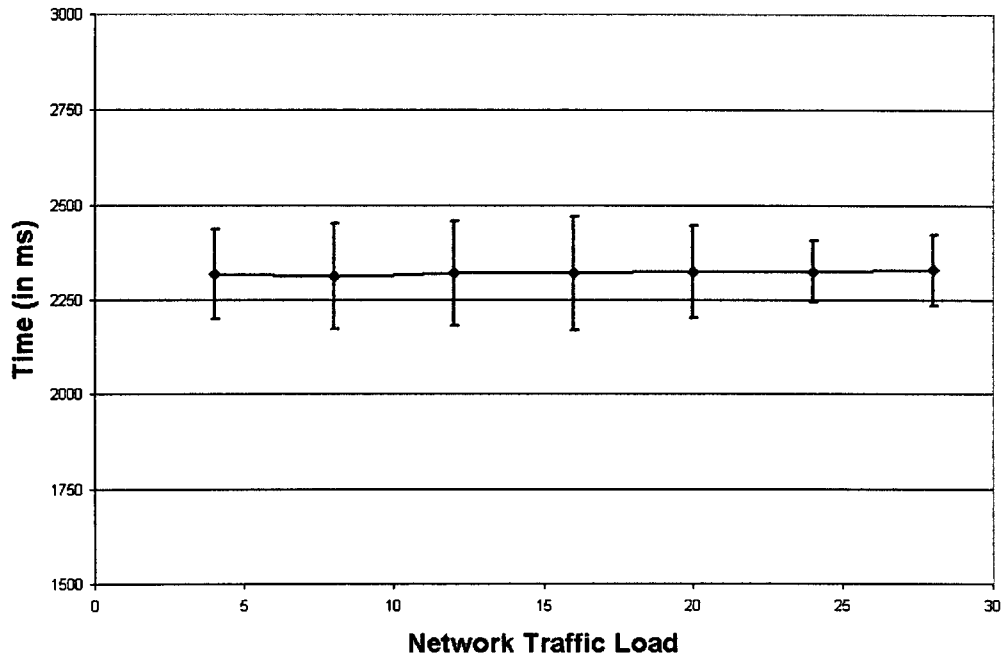


FIGURE 5.9: Time required by the flooding phase for networks with different traffic loads

5.4.5 Time needed by the multicasting phase

The time in Figure 5.10 represents the difference between the time when the last multicasting message has been received and the time when the multicasting message has been published by the executive sink.

In the case more traffic is generated in a network, the multicasting and determining phase may have few more message exchanges to handle, depending on the number of lightpaths that are fully comprised within the limited perimeter area. Theoretically, the effect on the performance should be insignificant.

The chart in Figure 5.10 illustrates a steady trend of the time required by the multicasting phase for the same network with different traffic loads going from 4 lightpaths up to 28 lightpaths. The multicasting phase took about $1.75 second$ in average. It is less than the time taken by the flooding phase, the reason being much less nodes are involved into the multicasting message exchange as opposed to the flooding message exchange in which the whole network takes part. The 95% confidence interval for the multicasting phase is $[\mu - 90ms, \mu + 90ms]$.

As a conclusion, the traffic load has no effect on the performance of the multicasting phase. This conclusion joins the theoretical claim described above.

5.4.6 Time needed by the determining phase

The time in Figure 5.11 represents the difference between the time when the last determining message has been received by the executive sink and the time when the first determining message has been published by a sink within the limited perimeter area. It does not involve the delta time defined for the determining phase.

As Figure 5.11 illustrates, the time needed by the determining phase is expressed in terms of ms. The chart shows the steady trend of the determining phase time. The 95% confidence interval for the determining phase is $[\mu - 49ms, \mu + 49ms]$.

As a conclusion, we might say that the traffic load has no effect on the performance of the determining phase. The few more messages, to be handled in the case of more traffic with more lightpaths fully comprised within the limited-perimeter area, do not influence the performance.

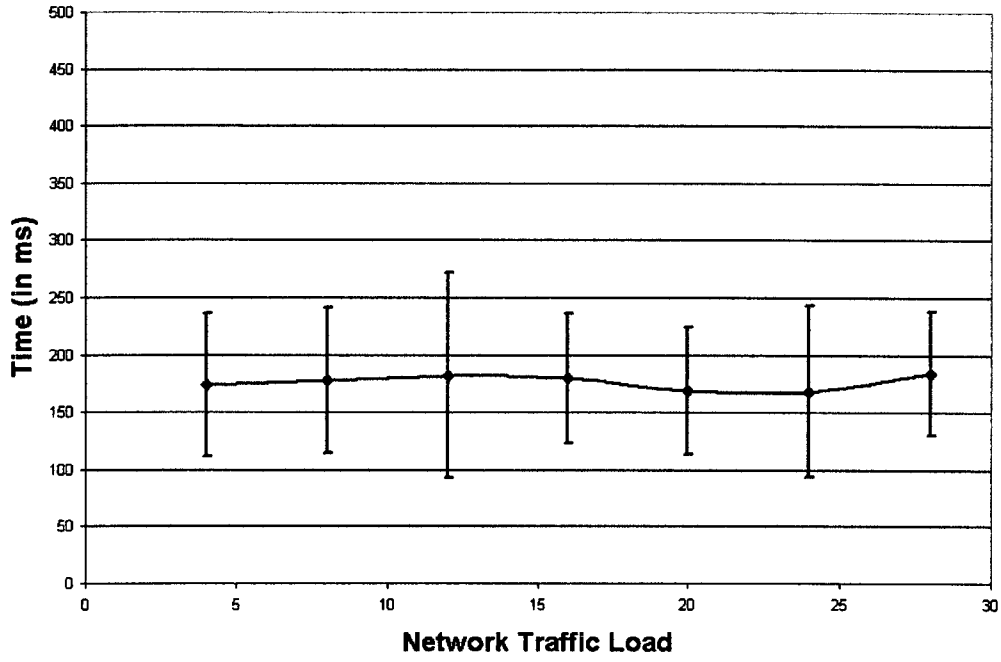


FIGURE 5.10: Time required by the multicasting phase for networks with different traffic loads

5.4.7 Time required to broadcast the failure message

The time shown in Figure 5.12 represents the time required for the failure broadcast once the faulty link is localized by the executive sink. It is that same executive sink which broadcasts the information to all the nodes in the network in order for the restoration process to start.

The time required to broadcast the failure message in Figure 5.12 shows a steady trend. In general, the broadcast time is about $0.17 second.$ This time is not comparable to the flooding time even though all the network nodes are involved in both the flooding phase and during the failure broadcast. The flooding time is in terms of seconds while the failure broadcast time is in terms of milliseconds. The reason behind this big difference in time is the fact that the flooding phase involves many publishers simultaneously while there is one and only one failure

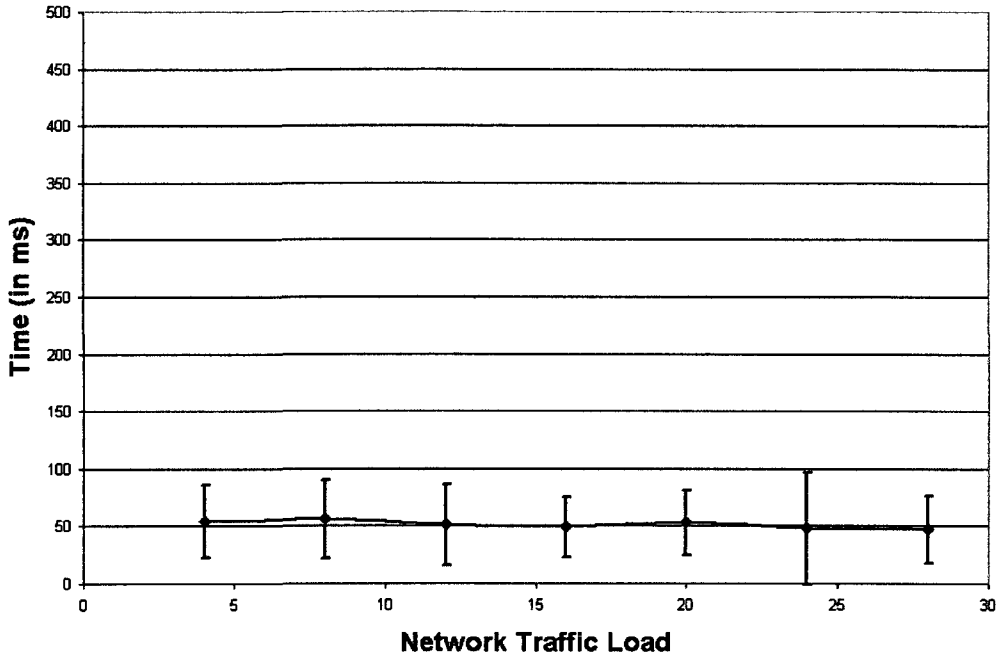


FIGURE 5.11: Time required by the determining phase for networks with different traffic loads

message published by the executive sink. The 95% confidence interval for the failure broadcast phase is $[\mu - 76ms, \mu + 76ms]$.

5.5 Network Diameter Effect

Theoretically, the LVM protocol performance would depend on the diameter of the network. The message exchange during the flooding phase makes the protocol take a little bit more time since the flooding process depends on the diameter of a network. If the time needed to send a message from a node A to an adjacent node B is t, then flooding a message in the network would take at least $\lceil d/2 \rceil xt$.

LVM Explorer implements the message exchange, as seen in detail in the previous chapter, through a topic which allows every two nodes to exchange messages

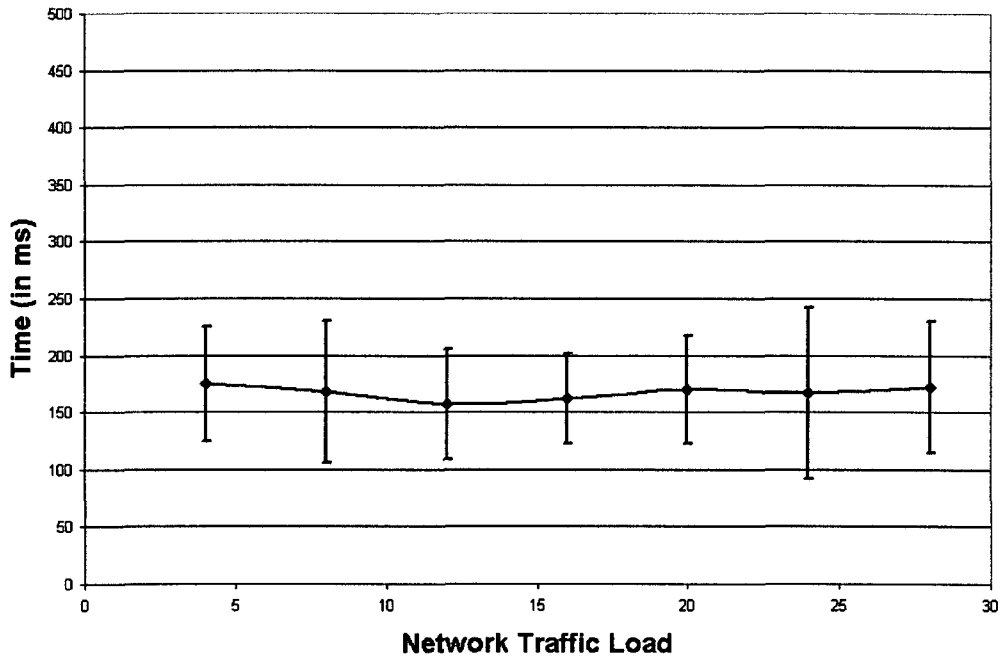


FIGURE 5.12: Time required by the failure message broadcast

directly even though they are not adjacent. As a result, the time needed to flood a message in the network would be similar to the time needed to send a message from a node A to an adjacent node B. It is the time required to publish the message into the topic (by the sending node) and to read it from the topic (by the receiving node) no matter how far are the sending and the receiving nodes in the network.

As a conclusion, the network diameter has no effect on the LVM Explorer and the LVM Protocol, thanks to the technological choices we have made.

5.6 Delta Times Effect

Obviously, the delta times defined for the flooding, multicasting and determining phases have an effect on the performance of the LVM Explorer. This is due to the fact that the synchronization process implemented through these delta times does not allow any node to start the next phase unless all the other nodes are done with the current phase.

The performed tests are a practical tool to tune up the delta times in order to match optimal values that would result into an excellent performance of the LVM Explorer.

As an example, we might reconsider the delta times based on the charts above as follows:

- 2.5 seconds for the flooding phase
- 0.3 seconds for the multicasting phase
- 0.15 seconds for the determining phase

This tuning would result into a gain of $4.35 seconds$ for each run of the protocol with guaranteed results brought by the LVM Explorer in around $6 seconds$ including a delta time of $3 seconds$ for the detecting phase. The detecting time can also be tuned based on certain characteristics of the network such as the diameter and the average propagation delay. This tuning does not fall into the scope of our project. Therefore we have used $3 seconds$ as a maximum value of the detecting phase delta time for all the tests we have performed.

5.7 Summary

Through a series of tests performed on different network topologies and different traffic loads, we concluded the following:

- Network size effect: We have run a series of tests on networks with the same traffic load for which we have changed the network size. Also, we have kept the delta times the same in order to see the effect of the network size on the LVM Explorer and the LVM protocol. We have found that the network size affects the performance of some parts of the LVM Explorer and some phases of the LVM protocol but has very little effect on the time required by the whole process to localize the faulty link. This is because of the configurable delta times.

- Network traffic load effect: We have run a series of tests on networks of the same size for which we changed the traffic load. We have used the number of lightpaths to measure the traffic load. Also, we have kept the delta times the same in order to see the effect of the traffic load on the LVM Explorer and the LVM protocol. Unsurprisingly, we have found that traffic load has no effect on the LVM protocol performance and the LVM Explorer performance. This is mainly because the LVM protocol runs on a limited-perimeter area.

- The performed tests would be a good start to tune up the configurable delta times. An optimal configuration would be the one that makes the waiting time of the corresponding phase exactly equal to the time required by that phase.

- The network diameter has no effect on the LVM Explorer and the LVM Protocol, thanks to the technological choices we have made.

Chapter 6

Conclusion and Future Work

6.1 Summary and Concluding Remarks

The main objective of this thesis is to present an emulation framework for the LVM protocol which has been claimed analytically [3] and by simulation [4] to be efficient in all-optical networks. This emulation framework proves that the LVM protocol can be easily implemented and confirms its high performance.

The present thesis first describes why efficient fault localization protocols are of particular importance in all-optical networks nowadays. Then, it presents a survey of recently proposed fault localization techniques for all-optical networks as well as their merits and weaknesses. This survey serves as a reference with which we can compare the performance of the LVM protocol. We have showed that the LVM protocol has a number of attractive features that make it among the best proposed protocols. Some of these features are the following:

- The LVM protocol does not require any monitors in the all-optical network;

- The average number of exchanged messages is reduced in the LVM protocol. This is because it starts operating in a limited-perimeter area rather than the whole network;
- Another advantage of the LVM protocol over some existing fault localization techniques such as cycle-based, path-based or trail-based approaches is that it does not require the assignment of loopback supervisory channels. It makes use of the channels of the network to localize the failure in an unambiguous way.

The emulation process is described in this thesis through a detailed explanation of the way the LVM protocol has been deployed in real optical networks. The architecture design of the LVM Explorer software, implementing the LVM protocol based on Multitier architecture, is explained in detail. The three layers of the LVM Explorer, including the presentation layer, the business layer and the data layer, are also described in detail. The use of the Service Oriented Architecture (SOA) is highlighted since it presents an attractive and important feature of the LVM Explorer. The SOA feature has allowed the creation of a virtual network, mapped on the physical network, running the LVM protocol. The physical switches are involved only in the alarm notification process at the time a failure occurs in the physical network.

The data exchange in the LVM Explorer involves three data sources including the database, the physical network and the configuration files. The database model is designed in a way to be vendor-independent in terms of the equipment description. The physical network is the source of the alarms launching the LVM protocol. The configuration files contain the delta time values for certain phases of the LVM protocol. These delta times are crucial to the synchronization of the LVM protocol running in a distributed system. The data communication process taking place between the different components of the LVM Explorer and between the virtual switches is also illustrated.

Finally, the performance of both the LVM protocol and the LVM Explorer has been evaluated based on a series of tests carried out on different network topologies and different traffic loads. We have calculated confidence intervals through five experiments of each test scenario. These tests are based on the best case for which the faulty link is localized within the smallest limited-perimeter area. We have concluded the following:

- Changing the network size has very little effect on the time required by the LVM Explorer and the LVM protocol to localize a faulty link. This conclusion has been drawn from a series of tests on networks with the same traffic load and different sizes.
- Changing the network traffic load has no effect on the LVM protocol performance and the LVM Explorer performance. This is mainly because the LVM protocol runs on a limited-perimeter area. This conclusion has been drawn from a series of tests on networks with the same size and different traffic load measured through the number of established lightpaths in the network.
- Changing the network diameter has no effect on the LVM Explorer and the LVM Protocol, thanks to the technological choices we have made.

The thesis provides, through Appendix A, a detailed presentation of the LVM protocol. It also contains a detailed example which explains the five phases of the LVM protocol: Pausing, Flooding, Multicasting, Matching and Concluding. The LVM protocol has been designed for all-optical networks. Therefore, it skips any optical power monitoring or spectrum analysis at the intermediate nodes of the established lightpaths.

Survivability is of a great concern in all-optical networks. An all-optical network has to incorporate efficient restoration procedures in order to maintain a

certain level of survivability. Fault localization is vital when link restoration is chosen to assure the survivability in an all-optical network. The LVM fault localization protocol can achieve excellent performance in terms of fault localization time while maintaining the maximum level of transparency of all-optical networks. The present thesis confirms the LVM protocol high performance and highlights its important advantage of being run out of the physical switches.

6.2 Future Research

The LVM protocol referred to in this thesis is the one capable of detecting single failures, or multiple failures only in non-overlapping limited-perimeter areas. A new version of the LVM protocol has been designed for detecting multiple failures even in overlapping limited-perimeter areas [41]. As an extension of this work, we suggest the following future research points:

- The current version of the LVM Explorer can be upgraded to support multiple failure localization in overlapping limited-perimeter areas. The architecture of the software will be of a great help since most of the components (GUI, Controller, Switch Service Publishing and Invoking) will stay practically the same. Only the protocol code is to be reviewed.
- The design and the emulation of a multi-wavelength LVM protocol is also a future research area. It consists of taking into consideration wavelength conversion within an all-optical network instead of considering the assumption that no wavelength conversions occur in the network.
- The design and the emulation of a new version of the LVM protocol capable of localizing both node failures and link failures. At the moment, the LVM protocol focuses on link failures since they are more frequent and more challenging.

- The design of a traffic generation mechanism which would be of a great help in the case the LVM Explorer fails to detect the faulty link because of not enough traffic. This mechanism would guarantee the faulty link localization.
- The LVM Explorer can be used to emulate any other fault localization protocol operating as a distributed system. The only requirement would be to develop the protocol code and to update the database model if needed.

Bibliography

[1] J. Zheng and H. T. Mouftah, *Optical WDM networks- concepts and design principles.* IEEE Press/Wiley and Sons, New York, NY, 2004.

[2] H. T. Mouftah and P.-H. Ho, *Optical Networks- Architecture and Survivability.* Kluwer Academic Publishers, Boston, 2003.

[3] A. V. Sichani and H. T. Mouftah, "Limited-Perimeter Vector Matching Fault Localization Protocol for Transparent All-Optical Networks," *IET Communications*, vol. 1, issue 3, pp. 472–478, June 2007.

[4] Z. A. Jaffari, J. Zheng, A. Eshoul, and H. T. Mouftah, "Performance Evaluation of LVM Fault-Localization Protocol for All-Optical Networks," *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE2008), Niagara Falls, Ontario*, pp. 1357–1360, May 2008.

[5] E. E. Mannie, "Generalized Multiprotocol Label Switching (GMPLS) Architecture," *RFC 3945*, October 2004.

[6] H.-B. Guo and G.-S. Kuo, "Improvements on Fault Localization in GMPLS-based Networks," *Proceedings of 2005 IEEE International Workshop on High Performance Switching and Routing (HPSR 2005), Hong Kong, China*, May 2005.

[7] J. Lang and et al, "Link Management Protocol (LMP)," *RFC 4204*, October 2005.

[8] A. Fredette and E. J. Lang, "Link Management Protocol (LMP) for Dense Wavelength Division Multiplexing (DWDM) Optical Line Systems," *RFC 4209*, October 2005.

[9] A. Amrani, J. Roldan, and G. Junyent, "Optical Monitoring System for Scalable All-Optical Networks," *Proceedings of IEEE Lasers Electro-Optical Society, San Francisco, California*, pp. 270–271, November 1997.

[10] Y. Kobayashi, Y. Tada, S. Matsuoka, N. Hirayama, and K.Hagimoto, "Supervisory Systems for All-Optical Network Transmission Systems," *Proceedings of IEEE Globecom'96, London, UK*, vol. 2, pp. 933–937, November 1996.

[11] M. Medard, S. Chinn, and P. Saengudomiert, "Attack Detection in All-Optical Networks," *Technical Digest of Optical Fiber Conference (OFC)*, pp. 272–273, February 1998.

[12] J. Patel, S. Kim, D. Su, S. Subramaniam, and H. Choi, "A Framework for anaging Faults and Attacks in All-Opical Transparent Networks," *presented at the DARPA Information Survivability Conference Exposition (DISCEX II), Anaheim, California*, June 2001.

[13] R. J. Bates, *Optical Switching and Networking Handbook.* McGraw-Hill, New York, NY, 2001.

[14] M. A. Rosegrant, *Introduction to Telecommunications.* Pearson Education, Upper Saddle River, New Jersey, 2007.

[15] R. Ramaswami and K. N. Sivarajan, *Optical Networks: A Practical Perspective.* Morgan Kaufmann Publishers, San Francisco, California, 2002.

[16] Hill Associates Telecommunications Training Experts, *Telecommunications: A Beginner's Guide.* McGraw-Hill/Osborne, Berkeley, California, 2002.

[17] W. D. Grover, *Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking.* Prentice Hall PTR, Upper Saddle River, New Jersey, August 2003.

[18] H. Zeng, A. Vukovic, and C. Huang, "A Novel Fault Detection and Localization Scheme for Mesh All-Optical Networks based on Monitoring-Cycles," *Photonic Network Communications.*

[19] Y. Kobayashi, Y. Tada, S. Matsuoka, N. Hirayama, and K.Hagimoto, "Supervisory Systems for All-Optical Network Transmission Systems," *Proceedings of IEEE Globecom'96, London, UK*, vol. 2, pp. 933–937, November 1996.

[20] M. Ilyas and H. T. Mouftah, *The Handbook Of Optical Communication Networks.* CRC Press, Boca Raton, Florida, 2003.

[21] A. V. Sichani and H. T. Mouftah, "A Novel Broadcasting Fault Detection Protocol in WDM Networks," *Proceedings of Biennial Symposium on Communications, Kingston, Ontario*, pp. 222–224, May 2004.

[22] A. V. Sichani, *Signaling Protocols for Survivable All-Optical Networks.* PhD thesis, SITE, U. of Ottawa, April 2006.

[23] S.-K. Lu, H.-C. Wu, S.-J. Yan, and Y.-C. Tsai, "Testing and Diagnosis Techniques for LUT-based FPGA's," *Proceedings of the IEEE 13th Asian Test Symposium, Kenting, Taiwan*, pp. 414–419, November 2004.

[24] A. Sichani and H. Mouftah, "Rolling Back Signaling Protocol -A Novel Fault Localization WDM Mesh Networks," *CIC China Communications Magazine*, vol. 1, no. 1, pp. 101–105, December 2004.

[25] M. Medard, S. G. Finn, and R. A. Barry, "WDM Loopback Recovery in Mesh Networks," *Proceedings of IEEE INFOCOM, New York, NY*, vol. 2, pp. 752–759, March 1999.

[26] A. G. HaileMariam, G. Ellinas, and T.Stern, "Localized Failure Restoration in Mesh Optical Networks," *Optical Fiber Communication Conference, Los Angeles, California*, vol. 1, p. 2, February 2004.

[27] P.-H. Ho, J. Tapolcai, and T. Cinkler, "Segment-Shared Protection in Mesh Communications Networks with Bandwidth Guaranteed Tunnels," *Networking, IEEE/ACM Transactions*, vol. 12, no. 6, pp. 1105– 1118, December 2004.

[28] S. Stanic, G. Sahin, H. Choi, S. Subramaniam, and H.-A. Choi, "Monitoring and Alarm Management in Transparent Optical Networks," *Broadband Communications, Networks and Systems. Fourth International Conference, Raleigh, North-Carolina*, pp. 828–836, September 2007.

[29] S. Stanic, S. Subramaniam, H. Choi, G. Sahin, and H.-A. Choi, "Efficient Alarm Management in Optical Networks," *Proceedings of DARPA Information Survivability Conference and Exposition (DISCEX-III), Washington, DC*, vol. 1, pp. 252– 260, April 2003.

[30] S. Stanic and S. Subramaniam, "Distributed Hierarchical Monitoring and Alarm Management in Transparent Optical Networks," *Communications, 2008. ICC '08. IEEE International Conference on. Beijing, China*, pp. 5281–5285, May 2008.

[31] C. Mas, P. Thiran, and J. Boudec, "Fault Localization at the WDM Layer," *Kluwer Photonic Network Communications*, vol. 1, no. 3, pp. 235–255, 1999.

[32] C. Mas and P. Thiran, "An Efficient Algorithm for Locating Soft and Hard Failures in WDM Networks," *Proceedings of IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1900–1911, October 2000.

[33] C.-S. Li and R. Ramaswami, "Automatic Fault Detection, Isolation, and Recovery in Transparent All-Optical Networks," *Journal of Lightwave Technology*, vol. 15, pp. 1784–1793, October 1997.

[34] B. Wu, K. L. Yeung, and P.-H. Ho, "Monitoring Cycle Design for Fast Link Failure Detection in All-Optical Networks," *Journal of Lightwave Technology*, vol. 27, pp. 1392–1401, 2009.

[35] B. Wu and K. L. Yeung, "M2-CYCLE: an Optical Layer Algorithm for Fast Link Failure Detection in All-Optical Mesh Networks," *Proceedings of Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE. San Francisco, CA*, pp. 1–5, December 2006.

[36] B. Wu, P.-H. Ho, and K. L. Yeung, "Monitoring Trail: A New Paradigm for Fast Link Failure Localization in WDM Mesh Networks," *Proceedings of Global Telecommunications Conference, 2008. GLOBECOM '08. IEEE. New Orleans, LA*, pp. 1–5, November 2008.

[37] H. Zeng, C. Huang, A. Vukovic, and M. Savoie, "Fault Detection and Path Performance Monitoring in Meshed All-Optical Networks," *Proceedings of Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE. Dallas, TX*, vol. 3, no. 3, p. 20142018, December 2004.

[38] S. S. Ahuja, S. Ramasubramanian, and M. M. Krunz, "Single-Link Failure Detection in All-Optical Networks Using Monitoring Cycles and Paths," *Networking, IEEE/ACM Transactions on*, vol. 17, no. 4, pp. 1080–1093, August 2009.

[39] J. Tapolcai, B. Wu, and P.-H. Ho, "On Monitoring and Failure Localization in Mesh All-Optical Networks," *Proceedings of IEEE INFOCOM 2009, Rio de Janeiro, Brazil*, pp. 1008–1016, April 2009.

[40] Z. A. Jaffari, "Performance Evaluation of Limited-Perimeter Vector Matching Fault Localization Protocol," Master's thesis, School of Technology and Health, KTH-Royal Institute of Technology, Sweden, June 2008.

[41] M. Khair, J. Zheng, and H. T. Mouftah, "Distributed Multi-Failure Localization Protocol for All-Optical Networks," *Proceedings of the 13th IEEE*

Conference on Optical Network Design and Modeling (ONDM 2009), Braunschweig, Germany, pp. 5.3.1–5.3.6, February 2009.

[42] M. G. Khair, *Fault Localization For Next-Generation Survivable All-Optical Networks.* PhD thesis, SITE, U. of Ottawa, December 2008.

[43] InsiTech, Inc., "J2EE Architecture." `http://www.insitechinc.com/`, April 19, 2010.

[44] Nortel Networks, "OPTera Metro 5200 Multiservice Platform- TL1 Interface: Chapter 18 to 30." Standard, Release 3.2, Issue 1, June 2001.

[45] H. T. Mouftah, "Meta-TL1 Specifications," tech. rep., University of Ottawa, Canada, September 2006.

[46] Sun Microsystems, "Getting Started with Java Message Service (JMS)." `http://java.sun.com/developer/technicalArticles/Ecommerce/jms/index.html`, April 18, 2010.

[47] Sun Microsystems, "Publish-Subscribe Based Messaging System." `http://java.sun.com`, April 19, 2010.

[48] M. Medard, S. G. Finn, and R. A. Barry, "WDM Loopback Recovery in Mesh Networks," *Proceedings of IEEE INFOCOM, New York, NY*, vol. 2, pp. 752–759, 1999.

Appendix A

LVM Fault Localization Protocol

A.1 Introduction

The LVM protocol is a novel fault-localization protocol designed based on the principles of distributed control and management mechanisms. It is capable of localizing link failures within an all-optical network by making use of the alarms notified by the physical switches at which the affected lightpaths end. One of the major merits of the LVM protocol is the concept of the limited-perimeter area. This concept allows the protocol to localize the faulty link without having to take the whole network into consideration unless it is necessary.

A.2 All-optical Networks

All-Optical networks (AONs) are constructed based on transparent optical nodes, in which the data does not undergo optical-to-electrical or electrical-to-optical conversions [15]. In All-optical networks, the signal that is received at one end is in effect the same as the one that has been originally sent, albeit possibly switched and amplified [15].

All-optical components such as optical amplifiers have limited or no electronic monitoring and analysis abilities. This is the reason why they may be able to detect loss of signal but cannot manage any fault-localization procedures. In contrast, there are components such as optical cross-connects (OXCs) which are capable of detecting failures and taking proper actions in response to service disruptions [3].

All-optical networks could be created based on different designs such as the overlay, augmented, peer-to-peer, or integrated. The all-optical WDM network architecture considered in this study is the overlay model in which optical switches interconnect data links and create the data network, while the control units including E/O/E conversions and optical amplifiers interconnect control links and construct the control network [3].

Unlike the SONET/SDH networks that operate in a point-to-point basis using the peer model, all-optical networks work in an end-to-end basis. Thus these networks could analyze data only at the end-points of set-up lightpaths (sinks) [3].

A.3 Optical Layer

The LVM protocol focuses on fault localization using all-optical components to preserve the high bandwidth of all-optical communications [3]. It skips any optical power monitoring or spectrum analysis at the intermediate nodes of the established lightpaths in order to provide the maximum level of transparency [3].

The optical layer consists of the Optical Channel (OCh) layer, also known as path layer, the Optical Multiplex Section (OMS) layer (line layer), and Optical Transmission Section (OTS) layer [15]. Optical restoration schemes are performed in both the OCh and OMS layers. The OCh restoration schemes deal with lightpaths, while the OMS techniques handle the restoration on the entire group of affected wavelengths comprising a link (a fiber). Figure A.1 shows different layers

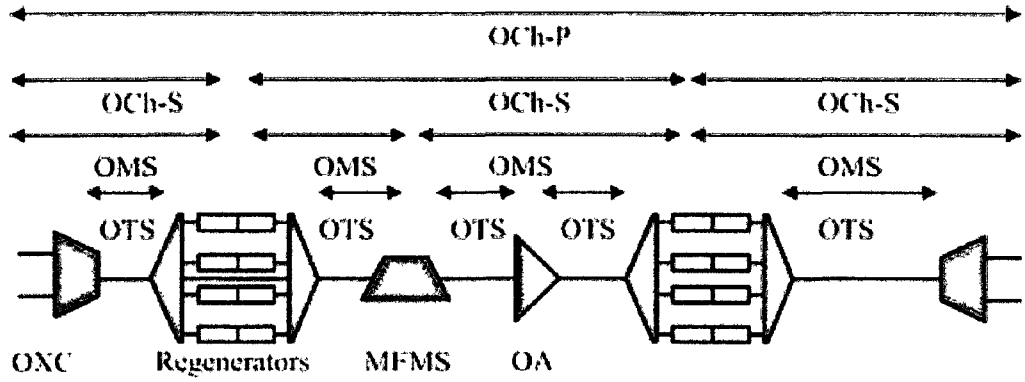


FIGURE A.1: OTS, OMS and OCh layers within the optical layer [3]

within which protection techniques can operate. The LVM protocol operates in the OCh-P layer based on the employed structural design, controlling model and definition [3].

A.4 LVM Fault Localization Protocol

The LVM protocol is a fault-localization protocol for link restoration in all-optical networks. Its strength comes from its associated time and space complexities narrowed down thanks to restricting the fault localization area to a small area called limited-perimeter.

Unlike existing fault localization protocols, the LVM protocol provides the maximum level of transparency by skipping any power monitoring or spectrum analysis at the intermediate nodes.

Although the LVM protocol has been designed to localize single failures, it could however track down multiple failures that occur in non-overlapped limited-perimeter area. It consists of five localization phases:

1. Detecting phase: When a link fails, all the destination nodes of the lightpaths passing through that link, called potential executive-sinks, will be able to detect the signal loss as a result of a high Bit Error rate (BER). Generally, the closest sink to the failed link will be the first to discover the failure. In the case several lightpaths terminate at the same potential executive-sink, the shortest path among them is selected to be the associated path to that sink. Each potential executive-sink pauses for a predetermined period of time to allow the entire network nodes to reach their steady states. The pausing time is very short and depends on the network characteristics such as the diameter and the average propagation delay.

2. Flooding phase: During this phase, each potential executive-sink floods an associated alarming signal, which encapsulates the node ID and the length of its associated path, to all network nodes to report its status. As a result of this, each potential executive-sink receives the alarm signals sent by the other potential executive-sinks, and then it compares the length of its associated path with the lengths of those contained in the received alarm signals. The potential executive-sink having the associated path with the smallest length is then recognized to be the executive-sink which is elected to conduct the fault-localization process. In the case more than one potential executive-sink have an associated path with the same smallest length, IDs of those nodes are also compared. In such a case, the sink with the smallest ID is chosen to be the executive-sink.

3. Multicasting phase: The executive-sink starts the fault localization process by creating a vector of its affected lightpath links, called Affected-Link-Vector (ALV). In the meantime, the executive-sink also defines a limited-perimeter area, which comprises all the ALV links and their neighboring nodes. Then, the executive sink multicasts a copy of the ALV vector to all the nodes within the limited-perimeter area. Under receiving the ALV vector, all the nodes stop taking further steps as a potential executive-sink.

4. Matching phase: In this phase, each ALV recipient node creates its own link-vector and then compares it with the ALV vector. If its link-vector has a common link with the ALV vector, the node will send a binary vector back to the executive sink. The binary vector consists of a matching vector and the status of the recipient's lightpath which is set to '1' if the lightpath is working. Otherwise, it is set to '0'. The matching vector contains the same links in the ALV vector. For a working lightpath (status '1'), an element in the matching vector is set to '0' if the corresponding link is found in the recipient's link vector. Otherwise, it is set to '1'. For a failed path (status '0'), an element in the matching vector is set to '1' if the corresponding link is found in the recipient's link vector. Otherwise, it is set to '0'.

5. Determining phase: In this phase, the executive-sink collects the binary vectors from all the sinks within the limited-perimeter. Then, it performs a logical AND on all the received binary vectors to determine the location of the disconnection. The link corresponding to '1' in the resulted binary vector will be determined as the failed link. The executive-sink broadcasts the location of the failure to all nodes in the network. After receiving the location of the failure, the end nodes of the failed link will initiate a link restoration protocol to restore the affected paths. In the case there more than one set bit in the resulted binary vector, the executive-sink could launch a second round of search by extending the limited-perimeter to a further edge such as the neighbors of the previously considered nodes.

We use the mesh network in Figure A.2 for clarifying our descriptions. We assume that each fiber is wavelength multiplexed in the applied network. Then the number of lightpaths traversing a link could be as large as the multiplexing degree. In general, once such a link fails the following scenario will occur: A set of sinks, connected to the affected established lightpaths, detect signal loss or high signal-to-noise ratio by examining data. Since we assume that the occurred failure for

each lightpath is only detectable by the end terminal (at the dropping point of OXC or OADM), the closest sink to the failed link is the one that discovers the failure sooner than others. This sink node is the one called the potential executive-sink. If the related OXC drops the data of more than one lightpath, then the shortest lightpath among them is selected as 'potential executive-sink'.

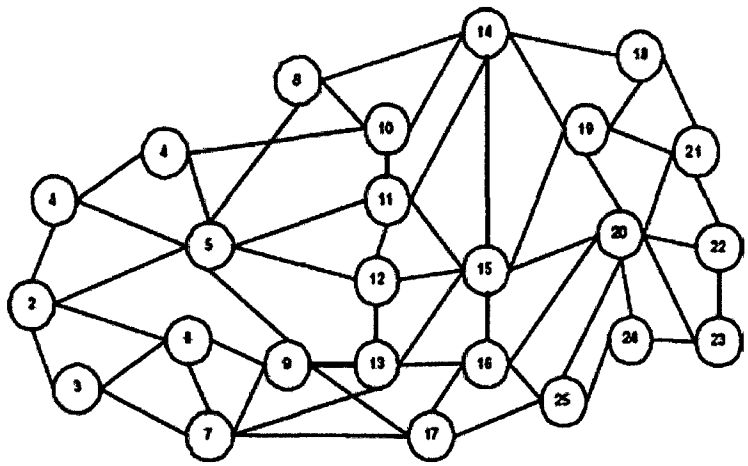


FIGURE A.2: Network topology before failure [3]

On Figure A.3, a group of potential executive sinks have been illustrated following a failure between node 12 and node 15. Those potential executive sinks are the destination nodes of all the affected lightpaths. They compete to be selected as the active executive-sink. The recipient whose the associated lightpath has the smallest length wins the competition and becomes the effective executive-sink.

Figure A.4 illustrates the ALV nodes and the associated limited-perimeter area. The executive sink multicasts a copy of the ALV to the nodes within the limited-perimeter area. Each one of the nodes which has received the ALV vector will answer back the executive sink by unicasting a binary vector containing the results of its calculations during the matching phase.

Once the LVM protocol detects the faulty link, a fast link-restoration protocol such as loop-back [48] is executed to restore the affected paths.

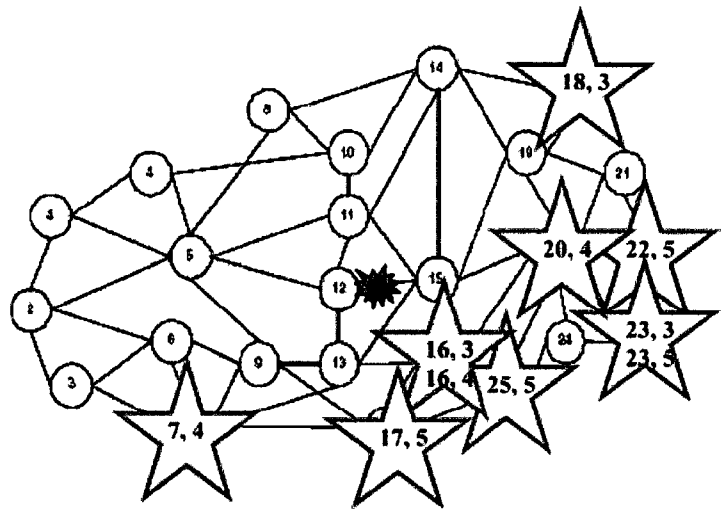


FIGURE A.3: The candidate Executive-Sinks (Potential Executive-Sinks) [3]

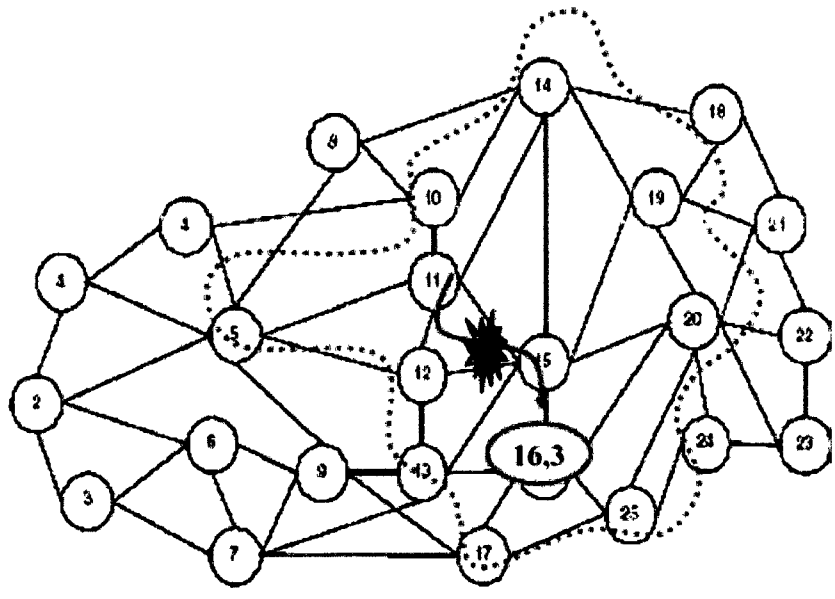


FIGURE A.4: Executive-sink and its associated path and limited-perimeter [3]

A.5 Example

Following is an example of the procedure of the LVM protocol in detail for all the five phases namely pausing, flooding, multicasting, matching, and concluding. We consider a small mesh network with a limited number of small lightpaths for simplicity. The network topology used in this example is shown in Figure A.5. The network has 5 established lightpaths which are the following:

- P1: 7, 8, 9, 10, 1
- P2: 8, 9, 10, 1
- P3: 7, 8, 9 2
- P4: 4, 10, 9, 3
- P5: 6, 2, 9, 10, 4
- P6: 11, 10, 1, 5

We assume that a failure occurs in the link between nodes 9 and 10. Because of this failure, four lightpaths (P1, P2, P3 and P4) are disturbed and three sinks (1, 3 and 4) detect a loss of signal. As soon as an affected sink sends an alarm, the LVM protocol is initiated which works as follows.

A.5.1 Detecting Phase

This is the phase where the failure occurs and all the sinks (1, 3 and 4) detect it. Each of these sinks is called potential executive sinks. In case of multiple lightpaths

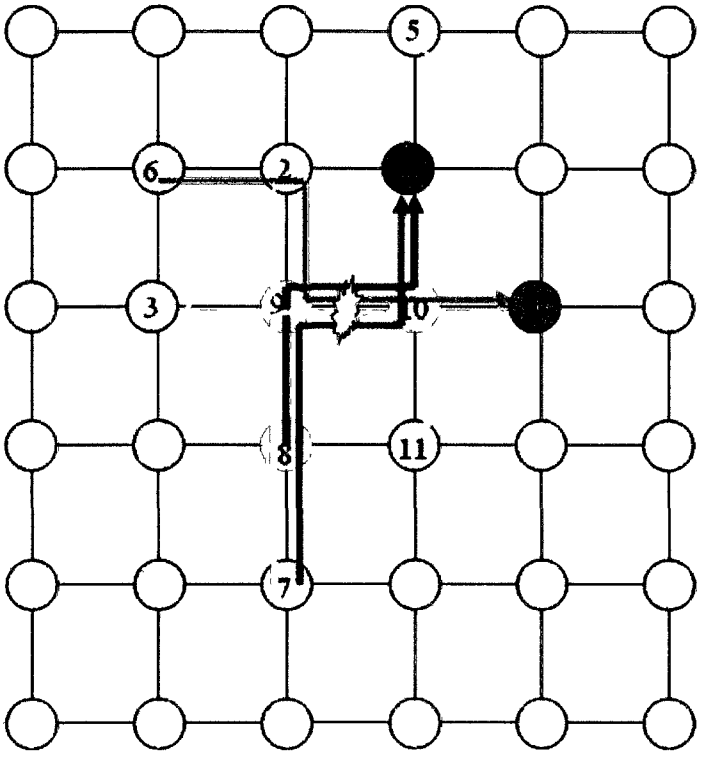


FIGURE A.5: Example network topology

terminating on one potential executive sink, the shortest one is selected as the associated path to the sink. All potential executive sinks wait for a predefined period of time to allow all the network nodes to reach the steady state. This waiting time depends on the network diameter and the average propagation delay in the network.

A.5.2 Flooding Phase

In this phase all the potential executive sinks flood an alarm message to the network to report their status. The message contains information about the Node ID and the length of the associated lightpath. Therefore, each potential executive sink receives the alarm signal sent by the other potential executive sinks. Each

potential executive sink compares the length of its associated lightpath with the associated lightpaths of the other potential executive sinks.

The potential executive sink which has the smallest associated lightpath is declared as the executive sink.

In our example on Figure A.6, two potential executive sinks have small associated lightpaths which are 1 and 3. In this case node ID is checked and the node with the smallest ID wins the competition. Thus in this case, node 1 is declared as the executive sink.

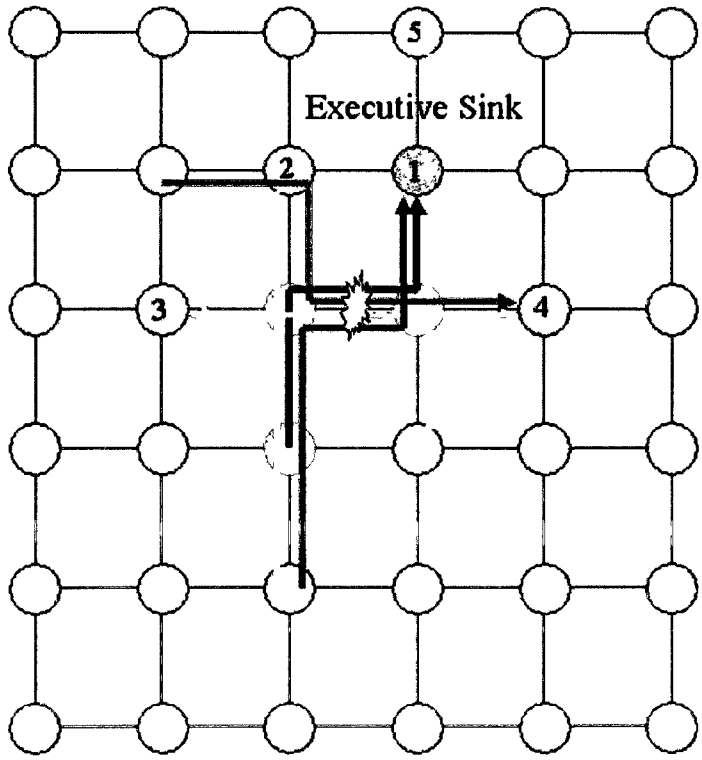


FIGURE A.6: Selection of the Executive Link

A.5.3 Multicasting Phase

In this phase the executive sink creates a vector of its associated lightpath links, called Affected Link Vector (ALV). The executive sink defines a limited-perimeter area which consists of all the ALV links and their neighbor nodes. In our example, Figure A.7 shows the ALV vector which is generated by the executive sink as well as the defined limited-perimeter area. Then, the executive sink multicasts a copy of the ALV vector to the all nodes within the limited-perimeter region.

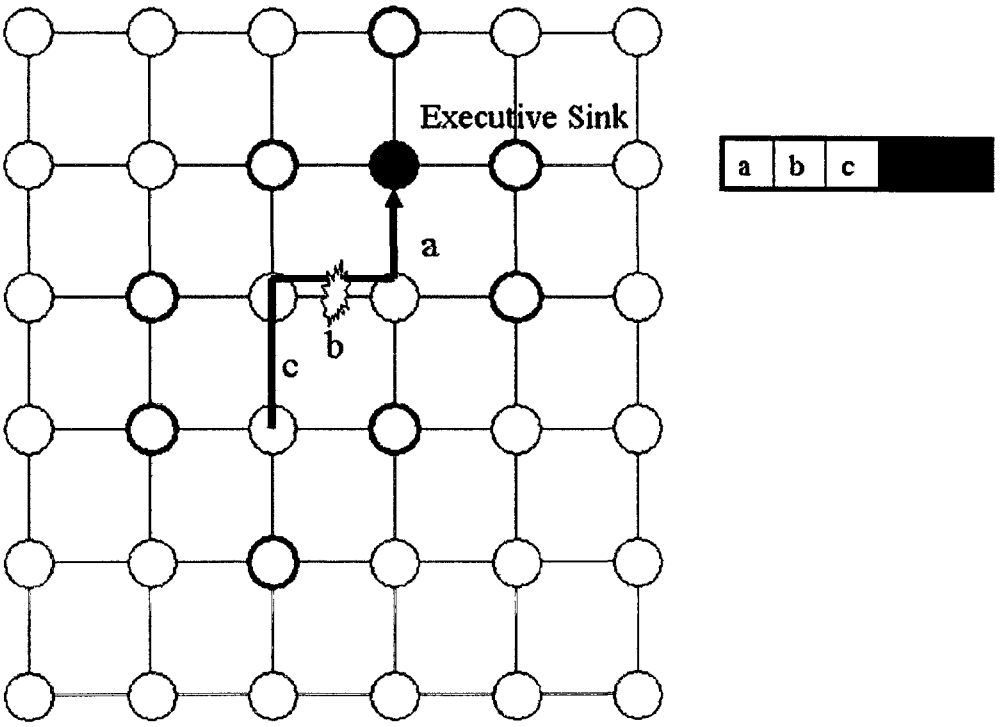


FIGURE A.7: Generation of ALV and definition of the limited-perimeter

A.5.4 Matching Phase

In this phase, each sink that has received the ALV vector creates its own link vector and compares it with the ALV vector. If its link vector has some link in common

with the ALV vector then it will send back a binary vector to the executive sink, which consists of a matching vector and the status of its associated lightpath. In our example, Figure A.8 shows that nodes 2, 3, 4 and 5 will send back a binary vector to the executive sink which sent the LVM vector.

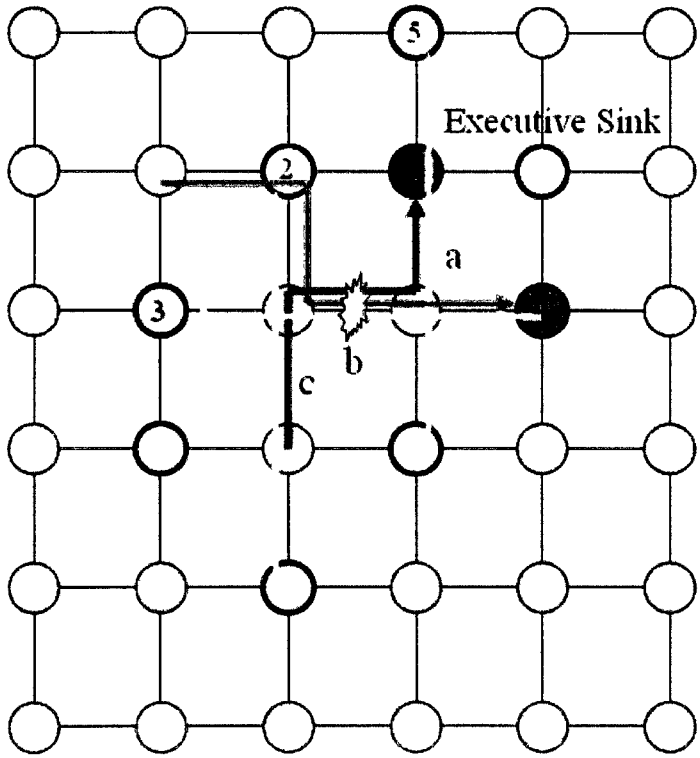


FIGURE A.8: Nodes replying to the ALV Vector sent by the executive sink

The values in the binary vector depend on the state of the lightpath. If the lightpath is working, its status is set to "1" and an element in the matching vector is set to "0" if the corresponding link is found in the sink's link vector. Otherwise, it is set to "1". For a failed lightpath (status "0"), an element in the matching vector is set to "1" if the corresponding link is found in the sink's link vector. Otherwise, it is set to "0". Figure A.9 shows the values of the binary vectors sent by nodes 2, 3, 4, and 5.

a	b	c	Status
1	1	0	1

Binary vector from Node 2

a	b	c	Status
0	1	0	0

Binary vector from Node 3

a	b	c	
0	1	0	

Binary vector from Node 4

a	b	c	Status
0	1	1	1

Binary vector from Node 5

FIGURE A.9: Binary Vectors sent back to the executive sink

A.5.5 Determining Phase

In this phase the executive sink collects the binary vectors from all the sinks within the limited-perimeter and performs logical AND on all the binary vectors to determine the location of the failure. The link corresponding to "1" in the resulted binary vector will be determined as the failed link. For our example, Figure A.10 shows the process of localizing the faulty link.

In the resulting vector, link b is the only link having "1" as a value, hence link b is the one localized as the faulty link in our example.

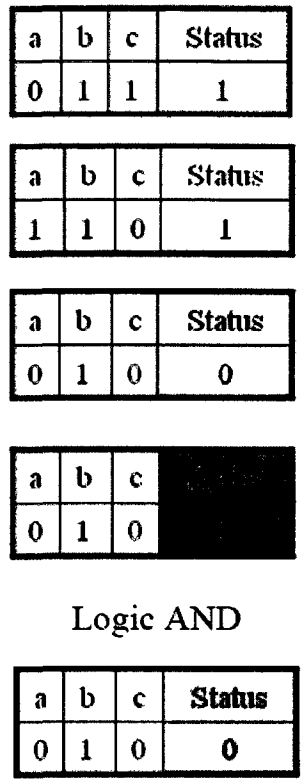


FIGURE A.10: Result Vector calculated by the executive sink

A.6 Summary

This appendix focuses on presenting the LVM protocol in detail. It describes the context in which the protocol is useful as well as the layer where it operates. It also presents the five phases of the protocol and illustrates them in a detailed example. One of the major merits of the LVM protocol is the concept of the limited-perimeter area. This concept allows the protocol to localize the faulty link without having to take the whole network into consideration unless it is necessary.

Appendix B

LVM Explorer User Manual

B.1 Introduction

This appendix presents the LVM Explorer user manual. it illustrates the screens one by one in the chronological order that should be followed the first time the LVM Explorer is used. Once installed the first time, LVM Explorer can be accessed through a super-user (administrator) that is created during the installation. The administrator creates the accounts to be used by the other users. After a successful login, a connected user can create his/her own networks and grant other users the required rights to access those networks.

B.2 Authentication

The authentication screen presents the entry point to the LVM Explorer for the LVM Explorer users. A login and password combination is required. Once a user enters his/her credentials and submit a request to connect to the application, the request is sent to the controller which interacts with the database to make sure the user has access to the application. In the case the user is authenticated

successfully, a profile object is returned back to the GUI which displays graphically the network elements. On the other hand, if the authentication fails, an error message is displayed to the user.

Figure B.1 illustrates the login screen.

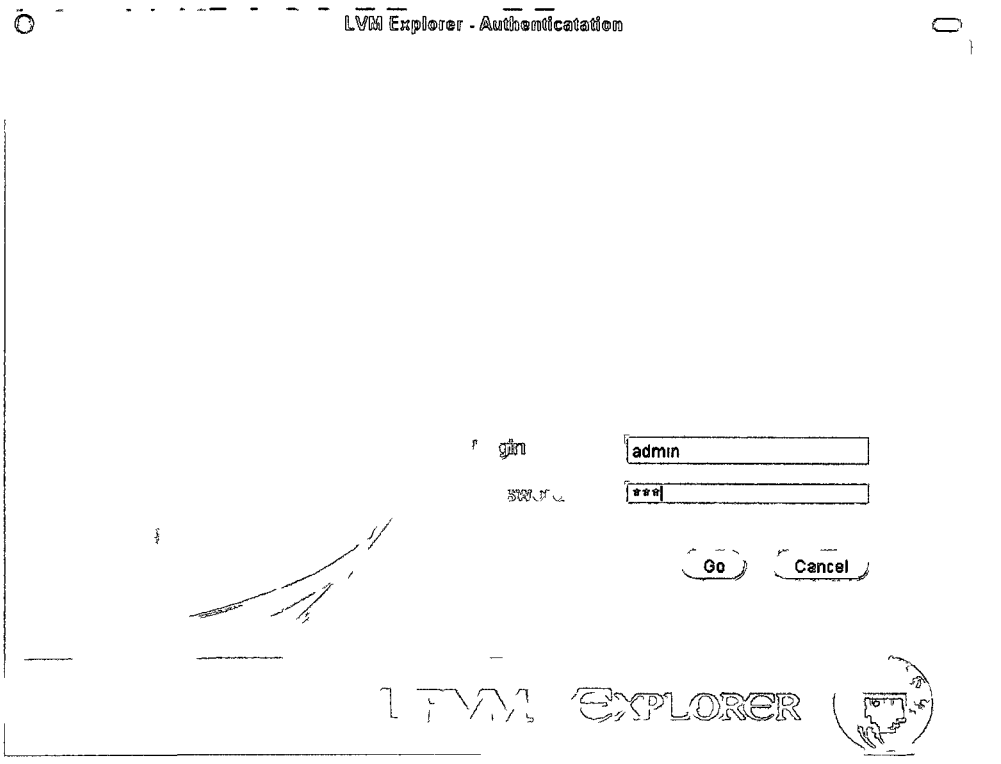


FIGURE B.1: Authentication Screen

B.3 Main Screen

The main screen is the first one displayed in the case of a successful authentication. It holds the list of networks to which the authenticated user has access. Each network contain all the data required to draw it graphically including nodes, physical links and lightpaths.

A GUI toolbox holding all the buttons, which call the required actions in LVM Explorer, is part of the main screen. It contains buttons to manage the switches, the physical links, the lightpaths and the users.

The left side of the main screen contains a small window illustrating the list of network to which the connected user has access. Each network in the list can be highlighted to view its network elements and edit them if required. Each node has an icon showing its status and the status of its corresponding switch service. The red icon means the corresponding BPEL process does not exist. The orange icon means that the corresponding BPEL process is deployed but the switch service is not running. The green icon means that the corresponding BPEL process is deployed and the switch service is running and listening for alarms. A contextual menu presents actions to change the states of the nodes. A user can deploy/undeploy a BPEL process or (re)start/stop the switch service.

At the bottom of the main screen a GUI log window displays the logs and all the message exchanges performed between the GUI and the other components of the LVM Explorer.

Figure B.2 illustrates the main screen.

B.4 User Functions

B.4.1 Add/Edit User

Figure B.3 shows the screen used to manager users in the LVM Explorer. It is displayed through the "manage users" button in the toolbox of the main screen. The "User Account Manager" screen shows the assigned users for each network. A contextual menu can be used to create a new user or to edit/delete an existing user. The "Edit" and "New" buttons call the same screen shown in B.4.

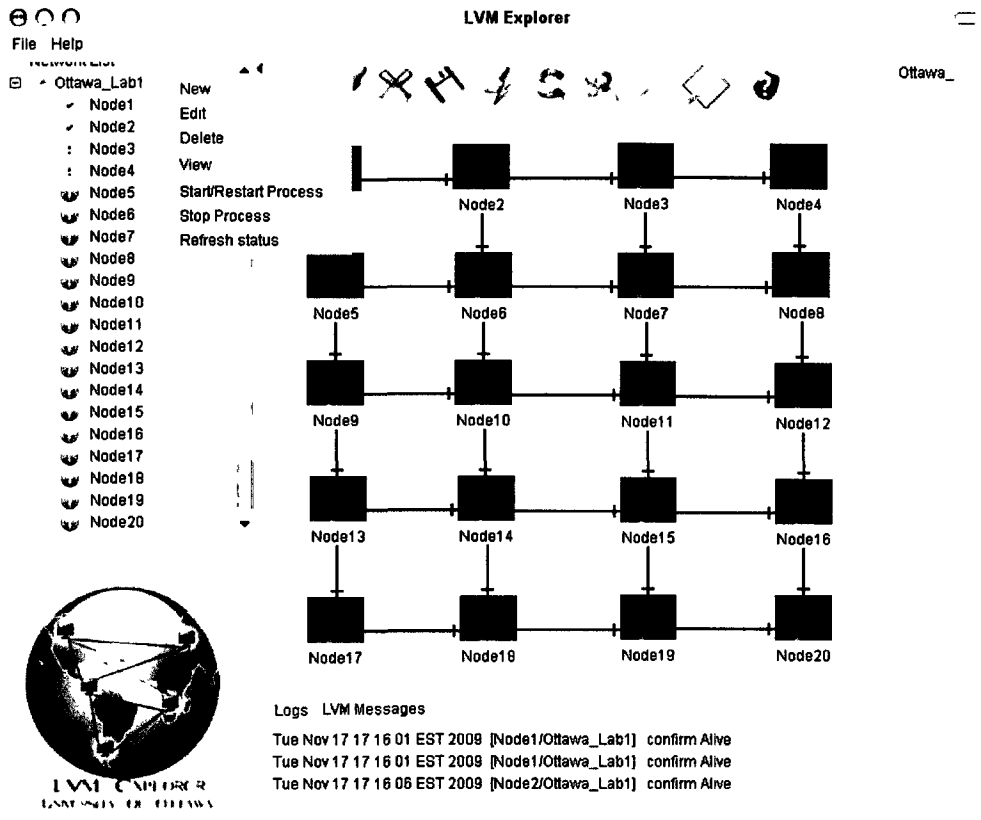


FIGURE B.2: Main Screen

B.4.2 Delete User

To delete an existing user, the user should select it in the "User Account Manager" screen in Figure B.3 and then use the contextual menu to remove it from the list. If the user account to delete is assigned to one network, the assignment to that network is removed and the user account is deleted from the database. But, if the user account is assigned to more than one network, only the assignment to the current network in the "User Account Manager" is removed and all the other assignments are still valid until they are removed for each network.

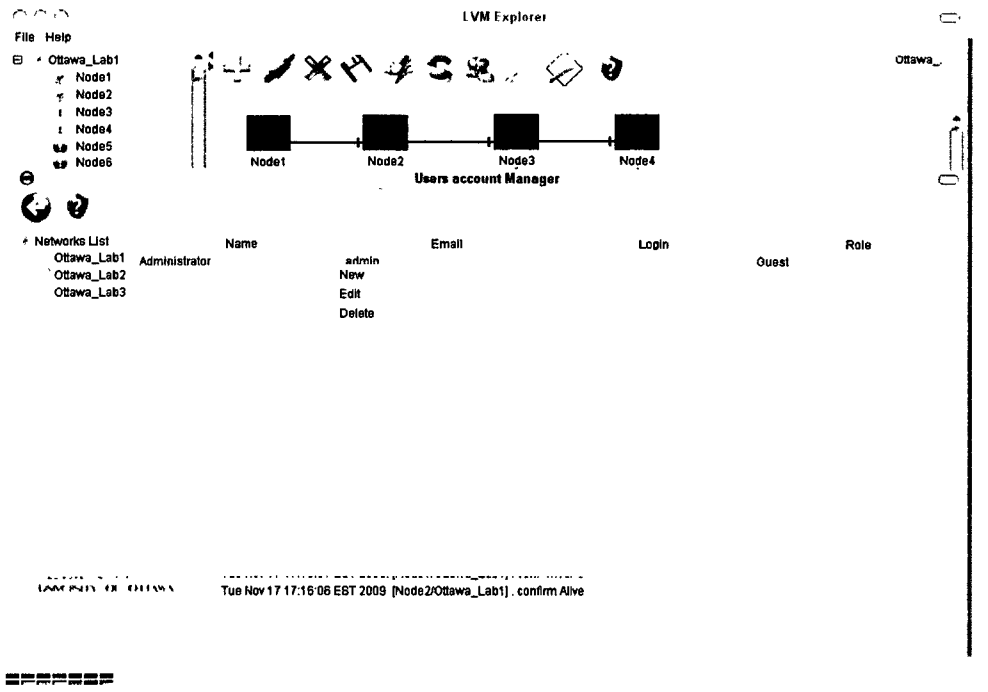


FIGURE B.3: User Account Manager screen

B.5 Network Functions

B.5.1 Add/Edit Network

The network contextual menu on the left window of the main screen can be used as it is illustrated in Figure B.5 to create, edit or delete a network.

The network contextual menu has three other menu options. The first two can be used to start/restart or stop the switch services of all the nodes belonging to the current network while the third one refreshes their statuses based on whether the corresponding BPEL processes exist and whether their switch services are running and listening for alarms.

Figure B.5 shows the screen used to create or edit a network.

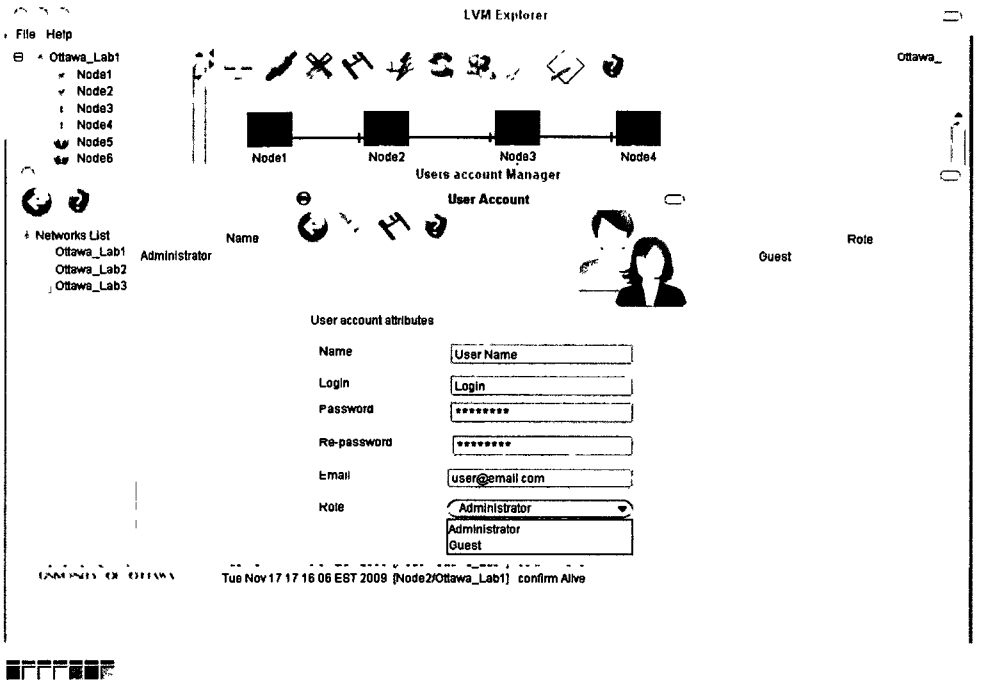


FIGURE B.4: User Account Screen

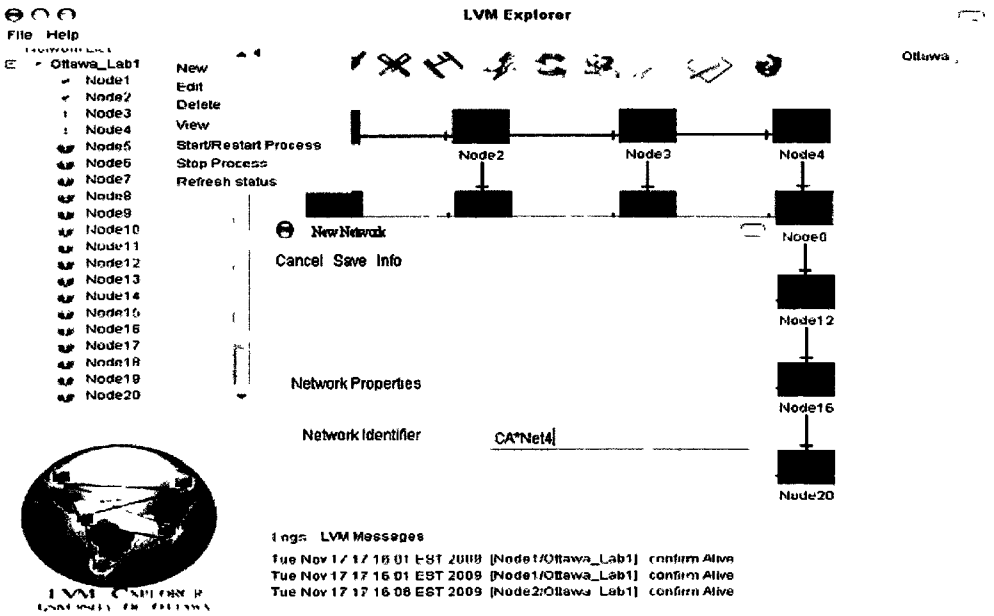


FIGURE B.5: Network Screen

B.5.2 Delete Network

In order to delete a network, select it on the left window of the main screen and then use The "Delete" menu option of the network contextual menu. A validation message is displayed to the user to confirm the action.

B.6 Switch Functions

B.6.1 Add/Edit Switch

In order to display the "New Switch" screen, either you use the switch button on the toolbox of the main screen or the node contextual menu on the left window of the main screen.

Figure B.6 illustrates the screen used to create a new switch or to edit an existing one. The data on the screen should be entered by the user before clicking on the "Save" button. A URL should be chosen from the list offered on the screen. The endpoints (Ports and Interfaces) are created through another screen which can be displayed by clicking on the button "Endpoints". The endpoint screen is shown on Figure B.7. The "Save" button on the endpoint screen creates the endpoint on the memory object. The "Save" button on the switch screen sends the node data to the database. At the same time, it launches the deployment of the node BPEL process.

The "Save" button on the switch screen sends a request to the controller to record the switch data on the database and to create its corresponding BPEL process on the URL selected by the user. If the request fails either because of a database problem or a deployment problem such as server not started, an error message is displayed to the user.

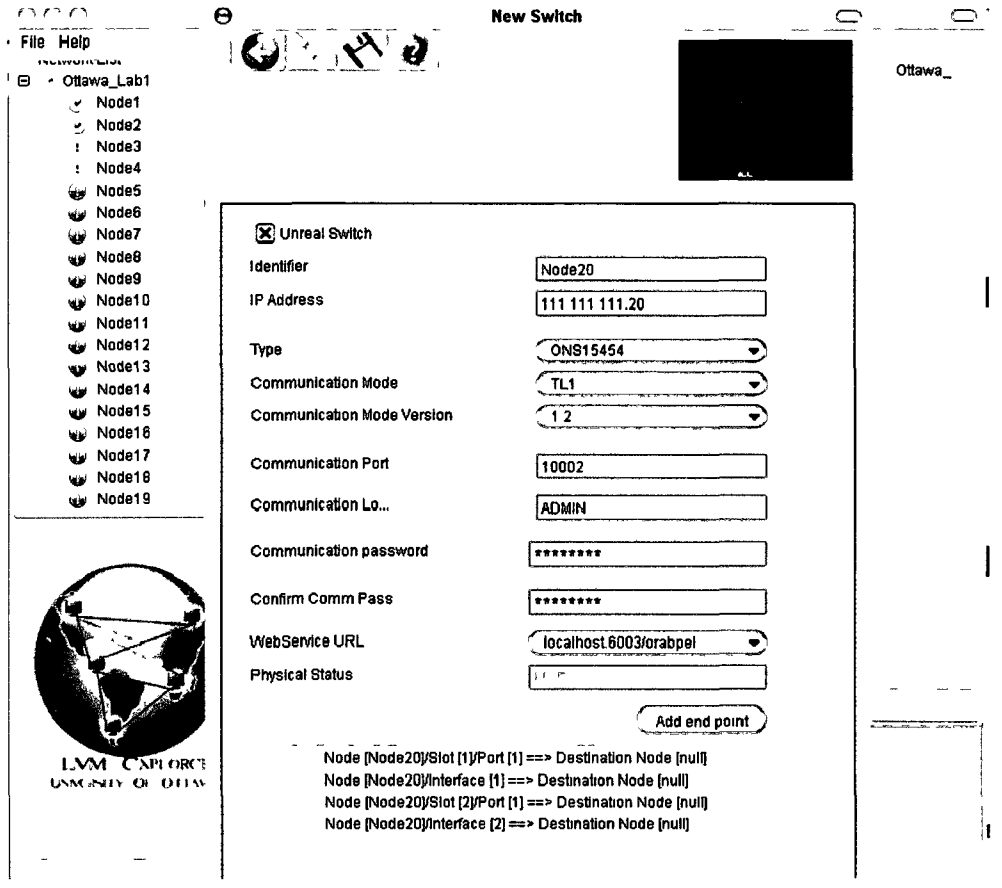


FIGURE B.6: Switch Screen

B.6.2 Delete Switch

There are two ways to delete a switch:

- Select it on the "Network Drawing Area" of the main screen and then use the "Delete" button on the toolbox;
- Select it on the left window of the main screen and then use the "Delete" menu option of the node contextual menu.

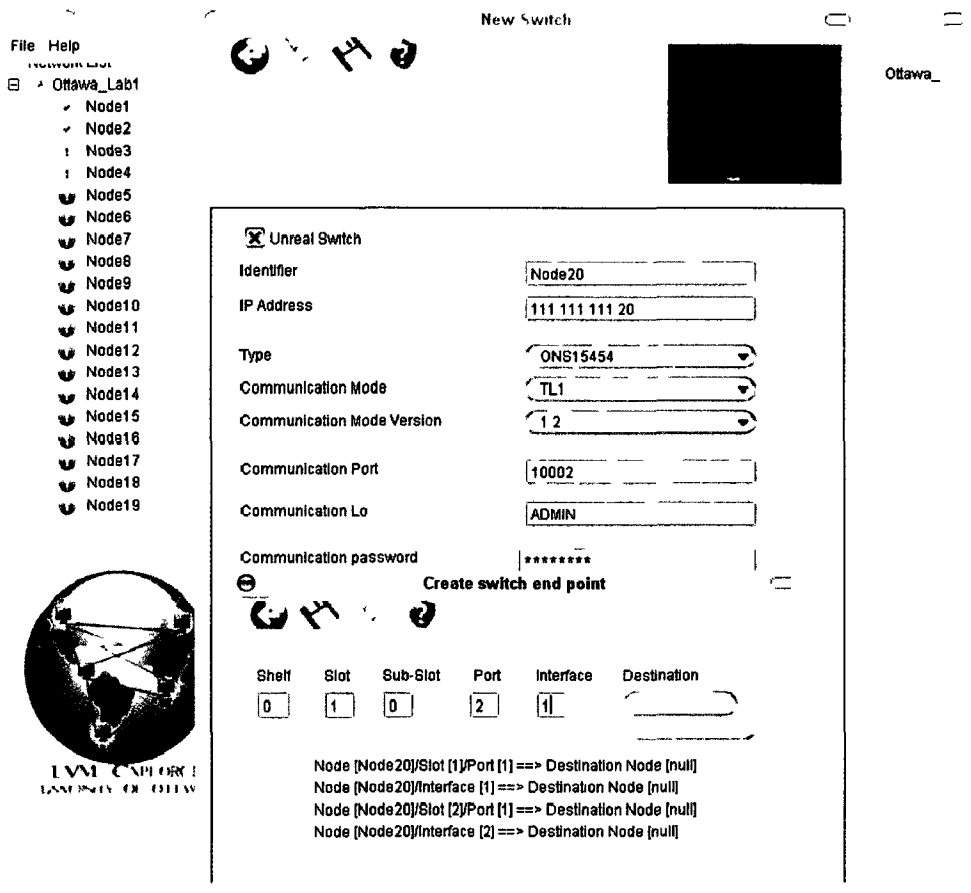


FIGURE B.7: Endpoint Screen

A validation message is displayed to the user to confirm the action. Once the user confirms the action, the switch is deleted from the database and its corresponding process is undeployed.

B.7 Physical Link Functions

B.7.1 Add Physical Link

In order to display the "New Link" screen, the user should use the link button on the toolbox of the main screen. Figure B.8 illustrates the screen used to create a new physical link. A physical link should be defined between two interfaces of two different nodes, otherwise an error message is displayed to the user.

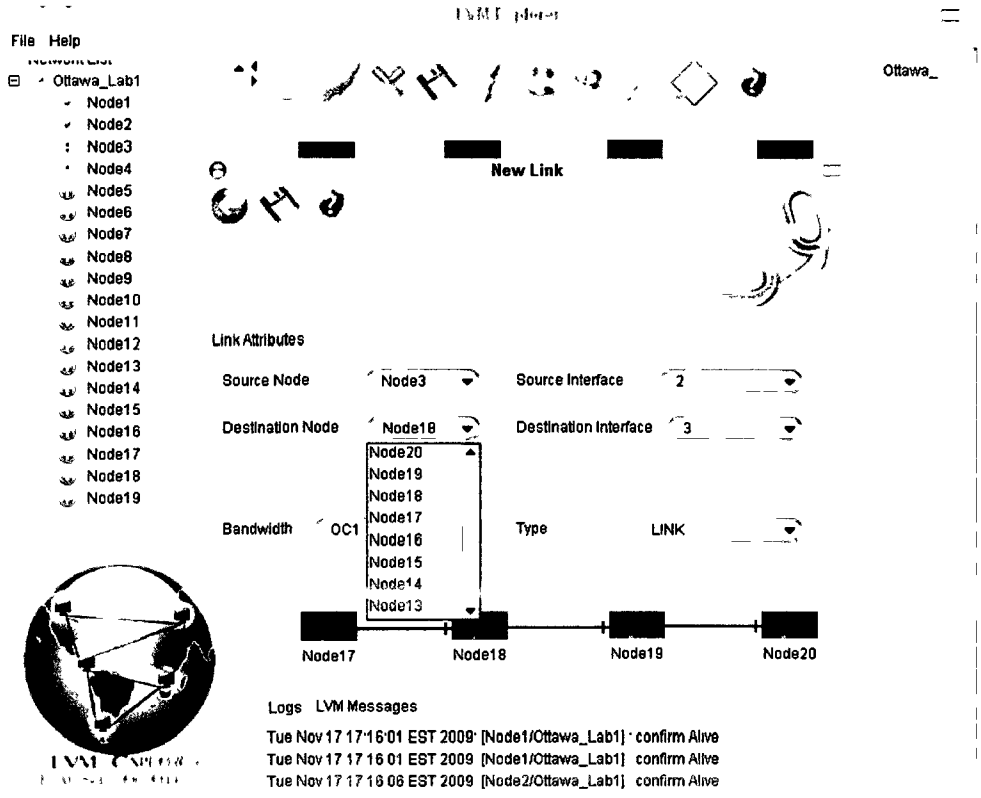


FIGURE B.8: Physical Link Screen

B.7.2 Delete Physical Link

To delete a physical link, the user should select on the "Network Drawing Area" of the main screen and then use the "Delete" button on the toolbox. A validation message is displayed to the user to confirm the action before the physical link is deleted from the database. The "Delete" button on the toolbox is used to delete both a node and a physical link. The network element to delete should be selected before clicking on the "Delete" button.

B.8 Lightpath Functions

B.8.1 Display Lightpaths

The "Lightpath Manager Screen" presents all the lightpaths of the networks to which the authenticated user has access. They are displayed by network. A contextual menu allows the user to create, edit or delete a lightpath. The "Lightpath Manager Screen" also gives the possibility to display the lightpaths in different colors on the network drawing area of the main screen. Figure ?? shows an example of displayed lightpaths using different colors.

Figure B.9 illustrates the "Lightpath Manager Screen" showing a list of colors used to distinguish lightpaths in the network drawing area. For example, if the user selects the green color for a given lightpath, that lightpath is displayed in green on the main screen. This helps the user in identifying the source node, the destination node and the intermediate nodes of the selected lightpath.

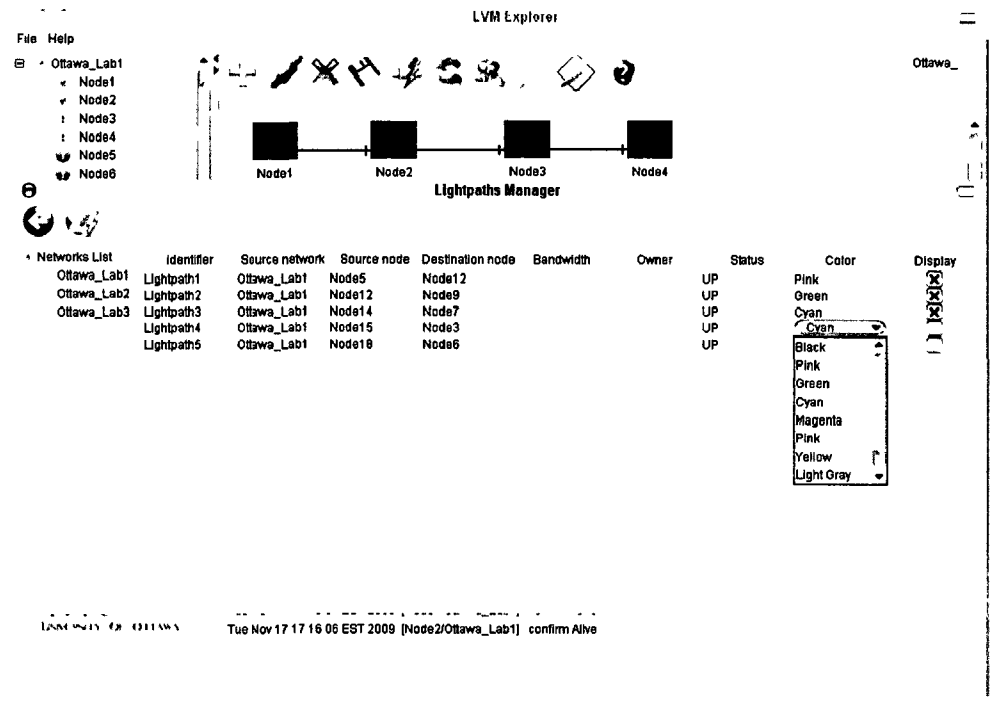


FIGURE B.9: Lightpath Manager Screen

B.8.2 Add Lightpath

To create a lightpath, the user should first chose its source port (the port of the source node from which it originates), then chose the physical links it crosses, and finally chose its destination port (the port of the destination node to which it ends).

Once the source port is defined, the LVM Explorer displays automatically its parent interface. The displayed interface should have at most one specific destination which is the second interface composing the physical link. The user should then select the next interface based on the next physical link which is crossed by the lightpath that is being defined. The next interface is then displayed by the system, and so on. The user continues to select the next interface and the system continues

to display the destination interface until we reach the destination node where the user chooses the destination port instead of an interface.

Figure B.10 shows the screen where the user selects the network on which the lightpath should be created. Figure B.11 illustrates the process of selecting a source port. Figure B.12 illustrates the process of selecting a destination port as an end to a path defined using a list of interfaces. The interfaces define the physical links which are crossed by the lightpath. Figure B.13 shows the description text zone which holds the name of the lightpath being created.

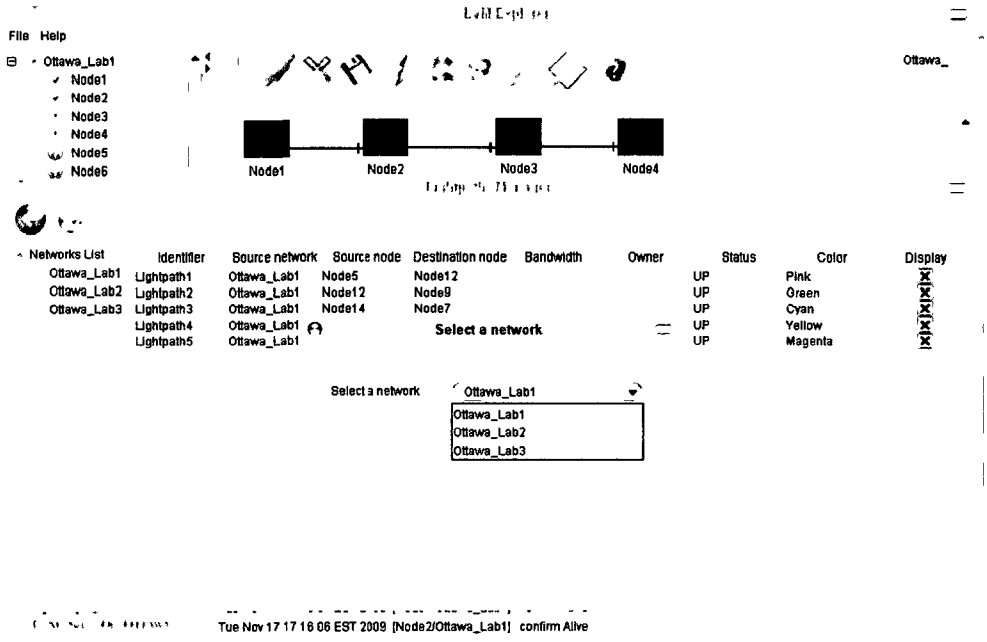


FIGURE B.10: Lightpath Screen 1

B.8.3 Delete Lightpath

To delete a lightpath, the user should select it in the "Lightpath Manager Screen" and then use the contextual menu to delete it. A validation message is displayed

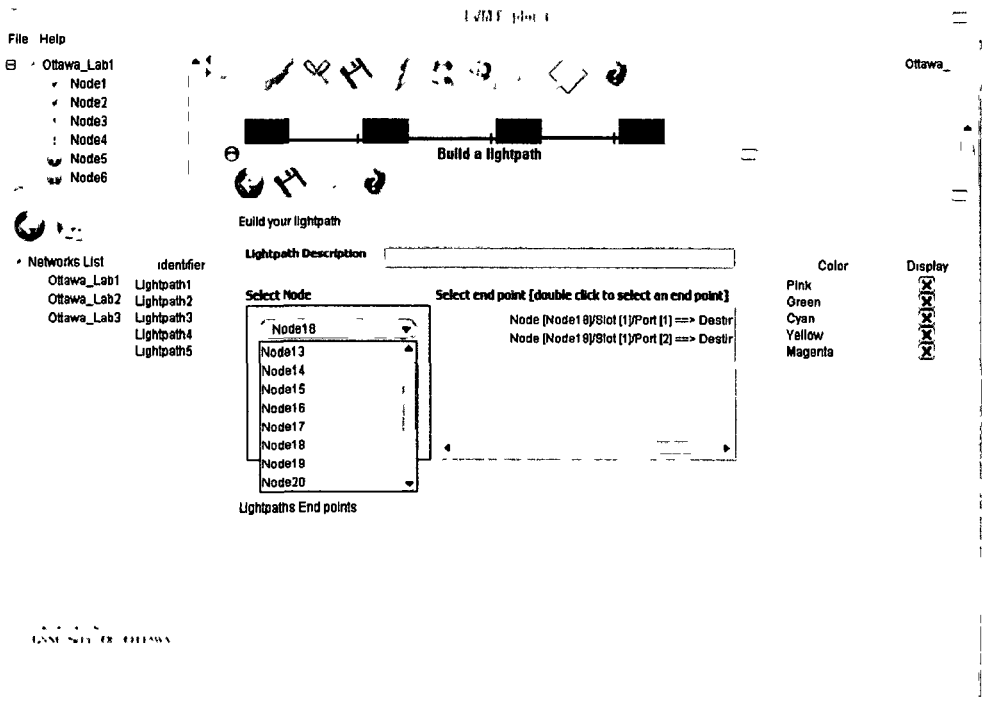


FIGURE B.11: Lightpath Screen 2

to the user to confirm the action before the selected lightpath is deleted.

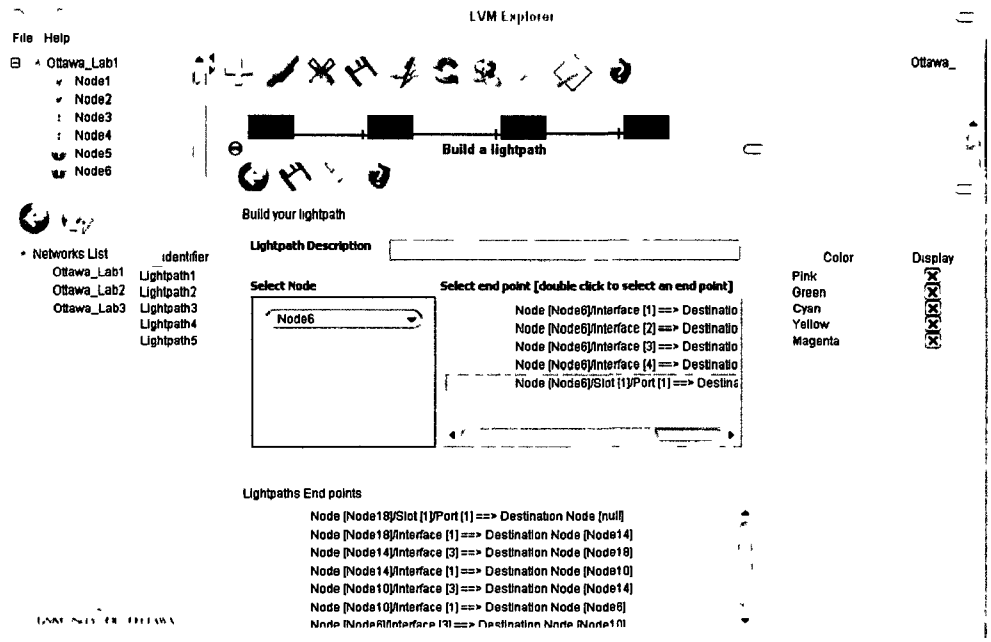


FIGURE B.12: Lightpath Screen 3

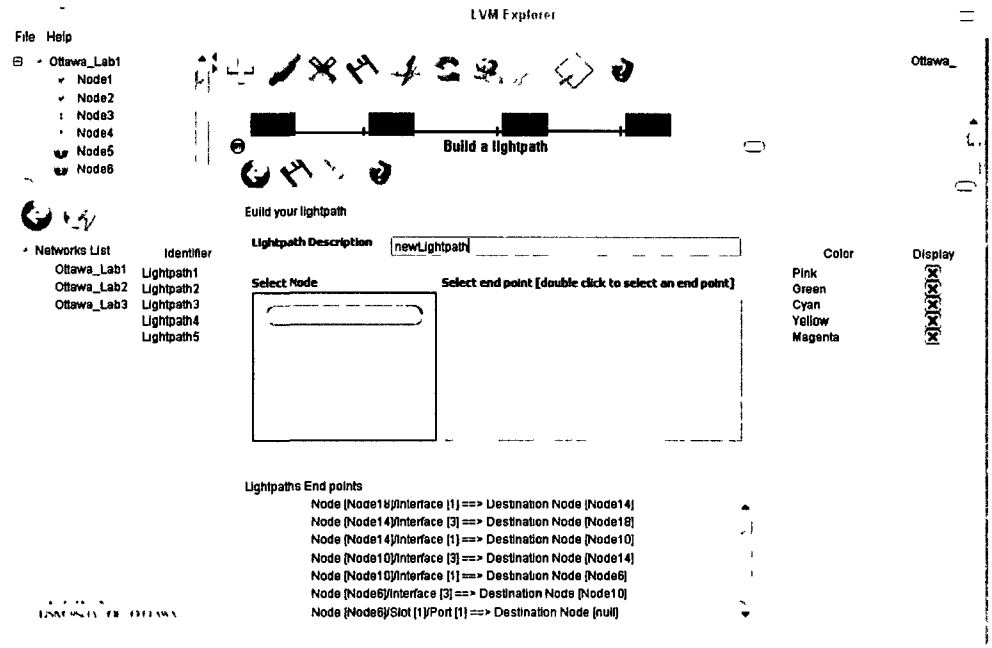


FIGURE B.13: Lightpath Screen 4

Appendix C

Confidence Interval Estimation

Let X be an unknown parameter to be estimated based on a random sample $X_1, X_2, ..., X_n$, from a distribution with that parameter X. We want an interval $[L, U]$, where L and U are two statistics calculated from $X_1, X_2, ..., X_n$, such that the interval includes X with a preassigned probability $1 - \alpha$. Specifically we want to have $PL \leq X \leq U \geq 1 - \alpha$ regardless of the value of X. Note that L and U are random variables, being functions of $X_1, X_2, ..., X_n$, so $[L, U]$ is a random interval, and L and U are confidence limits.

Consider a random sample $X_1, X_2, ..., X_n$ with a normal distribution $N(\mu, \delta^2)$, where δ^2 is assumed to be unknown and μ is the sample mean $\overline{X}$. From the properties of the normal distribution, it follows that there is 95% probability that X falls within a distance of $\frac{1.96\delta}{\sqrt{N}}$ from μ: $P[\mu - \frac{1.96\delta}{\sqrt{N}} \leq \overline{X} \leq \mu + \frac{1.96\delta}{\sqrt{N}}]$. $\overline{X}$ is within a distance of $\frac{1.96\delta}{\sqrt{N}}$ from μ if and only if μ is within a distance of $\frac{1.96\delta}{\sqrt{N}}$ from $\overline{X}$.

Therefore, we have, $[L = \mu - \frac{1.96\delta}{\sqrt{N}} \leq \overline{X} \leq \mu + \frac{1.96\delta}{\sqrt{N}} = U] = 0.95$.

For instance, if we consider a group of data for 12 independent runs per simulation experiment: $X_1, X_2, ..., X_n = 20, 35, 29, 25, 35, 39, 22, 27, 37, 40, 36, 21$, then, we calculate our parameters including $\overline{X} = 30.5$, $\delta = 7.342281$, and $\frac{1.96\delta}{\sqrt{N}} = 4.17124285$.

As a result, confidence limits would be $[L = 26.32875715, U = 34.67124]$. In our emulation results, 95% confidence intervals are adopted.