Design of a Fast Restoration Algorithm for IP Networks

By

Phi Thang

A Thesis
Submitted to the Faculty of Graduate Studies
In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering
Univeristy of Manitoba
Winnipeg, Manitoba, CANADA

0-612-80059-8

Design of a Fast Restoration Algorithm for IP Networks

BY

Phi Thang

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University

of Manitoba in partial fulfillment of the requirements of the degree

of

Master of Science

Abstract

The increase in demand for faster and more reliable networks has placed a heavy burden upon network and protocol designers. The issue of restorability is not new. It has been around for many years. It is now, as the number of Internet users and demands for higher data rates climb exponentially, that this issue is more prevalent. This thesis discusses the current restoration methods at the physical and network layer. A strong emphasis is placed on two technologies, SONET in the physical layer, and OSPF in the network layer. A two-phase restoration scheme is proposed. In this scheme, preconfigured routing tables are used to provide for fast restoration. An optimization model is proposed as a solution to determining the optimal set of preconfigured routing tables necessary. The steps needed for implementation using MPLS and Active Networking are briefly described.

Acknowledgments

I would like to thank my advisors, Dr. Jose Rueda and Dr. A.S. Alfa for giving me the opportunity to do research in the telecommunications field. Without their expertise in the related areas, I would not have been able to complete this thesis.

Special thanks goes out to TRLabs for providing an exceptional research environment where students can equip themselves with the knowledge to succeed in the future. This includes the TRLabs staff and students who have contributed their thoughts and comments into this thesis.

Thanks also goes out to my family for their support and encouragement through the years. And lastly, thanks to God for without Him, none of this is possible.

Contents

List of Figures

List of Tables

List of Acronyms

ABR	Area Border Router
AIS	Alarm Indication Signal
ANEP	Active Network Encapsulation Protocol
ANSI	American National Standards Institute
APS	Automatic Protection Switching
ARIS	Aggregate Route-based IP Switching
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
BGP	Border Gateway Protocol
BLSHR	Bi-directional Line-switched Self Healing Ring
BSHR	Bi-directional Self Healing Ring
FANP	Flow Atttribute Notification Protocol
IGRP	Interior-Gateway Routing Protocol
IP	Integer Programming
IP	Internet Protocol
IS-IS	Intermediate System-Intermediate System
JVM	Java Virtual Machine

LDP	Label Distribution Protocol
LLDown	Logical Link Down
LOH	Line Overhead
LOP	Loss of Pointer
LOS	Loss of Signal
LSA	Link State Advertisement
LSR	Label Switched Routers
MOSPF	Multicast Open Shortest Path First
MPLS	Multiprotocol Label Switching
OAM&P	Operations, Administration, Maintenance, and Provisioning
OC	Optical Carrier
OC-N	Optical Carrier level N
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
POH	Path Overhead
RFC	Request For Comments
RIP	Routing Information Protocol
RSTA	Restorable Spanning Tree Algorithm
RSVP	Resource Reservation Protocol
RTA	Restoration Table Advertisement

RTA3	Restoration Table Advertisement Active Application
SDH	Synchronous Digital Hierarchy
SDK	Software Developer Kit
SHR	Self Healing Ring
SOH	Section Overhead
SONET	Synchronous Optical Network
SPE	Synchronous Payload Envelope
SPF	Shortest Path First
STG	Spanning Tree Generator
STM	Synchronous Transport Module
STS	Synchronous Transport Signal
STS-N	Synchronous Transport Signal level N
TCP	Transmission Control Protocol
TDP	Tag Distribution Protocol
TOH	Transport Overhead
TTL	Time To Live
UPSHR	Uni-directional Path-switched Self Healing Ring
USHR	Unidirectional Self Healing Ring
VPN	Virtual Private Network
WDM	Wavelenth Division Multiplexing

Chapter 1

Introduction

1.1 Problem Statement

In the networks of today and the future, network survivability is, and will still be, an area of great concern to all users. The need for more reliable and robust networks is even more apparent as the number of Internet users continues to climb at an exponential rate.

With transmission rates in the order of Gigabits per second (Gbps), the need for reliable data transfer is critical. At such speeds, large corporations, small businesses, and even the average household Internet user can potentially lose large amounts of revenue with just a minor disruption in their communications network.

It would be incorrect to assume that current networks are not survivable, this is indeed far from the truth. There are many methods and techniques already in place that handle failures within the network. The problem is that current restoration techniques reside either at the Physical Layer or are handled by routing protocols.

Physical layer techniques provided by SONET [16] (Synchronous Optical Network) provide fast restoration within the tens of milliseconds. However, Physical Layer faults are often catastrophic faults that do not encompass faults at the Network Layer. With the growing support for IP (Internet Protocol) and its capability to enhance communication services, IP networks require stability and reliability. This leads to a need for restoration methods provided by the routing protocols that run at the IP Network Layer.

RIP (Routing Information Protocol) [1] uses a 30 second interval between routing table broadcasts. In the event of a failure within the RIP network where a routing table update is not received, RIP will wait 6 times the update interval (180 seconds) before declaring the routing table entry unreachable. OSPF (Open Shortest Path First) [2], on the other hand has a much better restoration scheme in place. With a typical hello interval (equivalent to RIPs update interval) of 10 seconds, OSPF waits for 4 times the hello interval before recalculating routes. This 40 second interval is far better than the 3 minutes that RIP requires, but at gigabit rates, tremendous amounts of information can still be lost. In the case of Transmission Control Protocol/Internet Protocol (TCP/IP) this could cause an excessive amount of retransmissions and lead to congestion due to the nature of the TCP protocol.

Since Physical Layer faults do not encompass Network Layer faults and restoration using the routing protocols is slow due to the mechanisms that are used, a new approach is needed in handling IP faults at the Network Layer. Restoration at the Network Layer can help to alleviate some of the problems associated with the two existing methods. This thesis will propose techniques to improve restoration times at the Network Layer. A novel approach is introduced to provide fast restoration using preconfigured routing tables. Our approach uses a two-phase system that can be incorporated with existing network protocols as an external module, thereby not interfering with the normal operation of the protocol itself.

1.2 Related Work

Restoration is a term typically associated with Physical Layer protection mechanisms. In most of the available literature to date, restoration appears in the context of survivable ATM (Asynchronous Transfer Mode) or SONET networks.

Network Layer restoration is a new area of research of very high interest, due to the rapid growth in demand for IP networks. There are several differences in the problem formulations for restoration at the Physical Layer, or even at layer 2 of the OSI (Open Systems Interconnection) Reference model (for example the ATM layer), and restoration at the Network Layer. One of the main differences is that restoration at the Network Layer might be temporary (not necessarily in response to a failure in the transport system). Failures at this layer are not restricted to open links or high bit error rates (BER), but also to congestion, system downtime, reconfiguration, rebooting of a router, addition of resources, etc. Restoration at the Network Layer refers to fast recovery and convergence to a new optimal state. Restoration must be dynamic, that is, there is not necessarily a unique optimal state. The optimal state can vary and can be recalculated as required.

In this section, a survey of available literature, both recent and classic, are studied for restoration techniques and applications. The reader is encouraged to review these publications and references therein. A common theme in the references discussed below is that they consider restoration of failures in high-speed networks which are near catastrophic. This type of failure is typically due to failures in the communications equipment. A failure in a relatively small amount of time can lead to very high data losses.

One of the most important contributions to restoration at the Physical Layer is the self-healing ring concept [3]. In that paper Grover, presented a study of the restoration problem in telecommunications networks and proposed a mechanism for decentralized restoration. This decentralized mechanism was intended to aid and complement centralized protection systems. The paper discussed problems related to restoration due to loss of all or most of the physical transmission facilities. Grover pointed out that restoration via rerouting through redundant connections should not be confused with individual call rerouting. The self-healing ring concept has been widely accepted and it has been applied in the industry.

A study of restoration schemes for survivable ATM networks was presented by Murakami and Kim [5]. They also proposed a methodology for end-to-end restoration through a comparative analysis of the minimum link capacity installation cost. The methodology is an optimal capacity and flow assignment algorithm for the self-healing of ATM networks based on end-to-end and line restoration. One of their main results is that the economic advantage of the end-to-end restoration method might be marginal for a well-connected and/or unbalanced network. In the solution of the linear programming problem they presented an elegant approach by solving and utilizing the dual problem for recalculation of the solution. Murakami and Kim also proposed a two-step scheme for fast restoration. In their method, an accelerated recovery procedure is executed upon the knowledge of a failure. After recovery procedures are complete, a new optimal solution is calculated for the entire network.

Schwartz and Stern [6] presented a study of routing techniques for computer networks. Their study was published in 1980, years before the wide-spread use and knowledge of the public Internet. They discussed the routing architectures SNA (IBM) and DNA (DEC). Their paper provided a description of the practical issues related to routing. Today, the shortest path technique is implemented in more efficient routing algorithms such as the OSPF. This algorithm is based on the calculation of the shortest path from one node to another and it stores information about the next hop only. This way, a routing table is created.

A recent paper by Balakrishnan, Magnanti and Mirchandani [7] discussed serviceability and survivability issues in the event of catastrophic failures. Serviceability refers to the availability of vast bandwidth and survivability refers to failure protection. They presented a problem formulation for restorability based on a multi-level architecture. Although the paper is not specific to Internet routing, it provided a valuable point of view on how to design multi-level restoration algorithms.

Ben Yahia and Robach [8] presented a new approach that utilizes virtual path management functions and self-healing rings. This is a very interesting perspective, as they discuss interactions between the physical and data link layers of the OSI model. Their paper also concentrated on catastrophic failures. One of the disadvantages of this approach is that it utilizes virtual paths which are permanently established between nodes.

Gersht and Shulman [9] have introduced a technique for fast restoration of virtual paths in mesh ATM networks. The technique is based on the allocation of spare virtual paths. This is, in essence, similar to restoration at the SONET layer based on dual fiber facilities. As has been pointed out by Grover [4], mesh techniques are not necessarily the best way to go, due to the high cost and a hybrid technique will be more desirable.

Spare capacity assignment for different restoration strategies in survivable mesh networks has been presented by Van Caenegem, Wauters, and Demeester [10]. In their paper they discussed strategies for single link failures. They also discussed two optimization techniques, simulated annealing and linear integer programming to solve the problem of single link failure. Their results can be applicable, for example, to WDM (Wavelength Division Multiplexing) networks. However, there was no interaction with other layers.

Hsing, Cheng, Goncu and Kant [11] presented a restoration methodology based on pre-planned routing for ATM networks. The idea is based on communication of neighbouring nodes with the sources to find an alternate route. This methodology of having routes calculated prior to any faults occurring is very beneficial in terms of speed. Once a failure occurs, all that is necessary is for the nodes to do a lookup on the pre-planned routes, which is faster than performing calculations as faults occur.

Finn et. al. [12] [13] introduced a new approach for protection switching using trees. In their approach each node in the network will find two directed spanning trees, one to be used for normal conditions and one for failed conditions. With assumptions that the network be vertex or edge redundant, the algorithm is restricted to certain network topologies. Their method can be viewed as a generalization of some techniques used in SONET, such as Automatic Protection Switching and Self-Healing Rings.

Grover and Stamatelakis [14] [15] have developed a restoration scheme using pre-configured cycles (p-Cycles). This methodology is similar to SONET rings yet use spare capacity much more efficiently for restoration. The p-Cycles restoration scheme for physical and IP networks were discussed with focus being placed on the SONET physical layer. These papers also give a comparison of p-Cycles and SONET rings.

1.3 Objectives and Thesis Organization

This thesis aims to provide a new technique for restoration at the Network Layer by introducing a two-phase restoration system as shown in Figure 1.1. The occurence of a network failure will cause the primary scheme to be activated. This scheme assumes control of network routing and uses the preconfigured routing tables to determine the next hops. If the failure is quickly corrected or does not persist for a certain amount of time, the system will return to its original state before the failure. However, if the failure is persistent, OSPF will activate its restoration mechanism after the RouterDeadInterval time has elapsed to bring the network to an optimal state. This will be discussed in greater detail in Chapter 3.

Relevant issues related to restoration will be addressed without any consideration for other important elements. The main focus of this thesis will be on the restorable

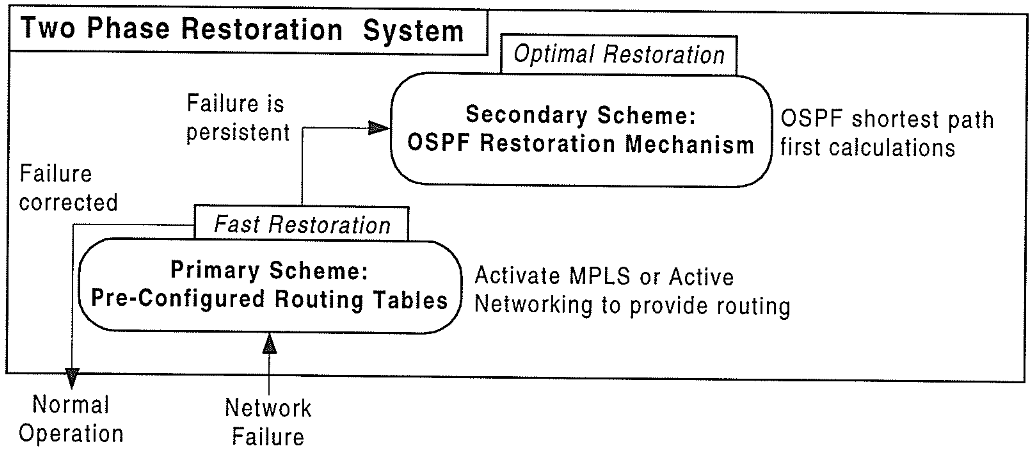

Two Phase Restoration System
Optimal Restoration
Failure is persistent
Secondary Scheme: OSPF Restoration Mechanism
OSPF shortest path first calculations
Failure corrected
Fast Restoration
Primary Scheme: Pre-Configured Routing Tables
Activate MPLS or Active Networking to provide routing
Normal Operation
Network Failure

Figure 1.1: Two Phase Restoration System

spanning tree algorithm and how it can be implemented in today's networks.

This dissertation is organized as follows. The next chapter discusses current restoration mechanisms that are in place at the Physical Layer with specific emphasis on SONET. It also provides insight into the two main IP routing protocols, RIP and OSPF. Chapter 2 will conclude with a brief introduction to graph theory. A two phase approach to Network Layer restoration is discussed in Chapter 3, including a formal problem definition and details of the restorable spanning tree algorithm. A review of the obtained results is given in Chapter 4 as well as a case study on the heuristic spanning tree algorithm. Chapter 5 will look at two possible methods of incorporating the restorable spanning tree algorithm into an existing network. It also discusses some of the issues that might arise when considering active networking or MPLS [32] in the two phase restoration scheme. Concluding that chapter is a discussion of other areas that were examined to improve restoration. Finally, Chapter 6 summarizes the major contributions of this thesis and proposes possible extensions for future research.

Chapter 2

Restoration Mechanisms and Background

2.1 Introduction

The issues of restorability and survivability have existed for a long time. Solutions that address these issues to provide restorable networks either reside at the Physical Layer or are provided by routing protocols at the Network Layer. It is necessary to review the current techniques to gain a full understanding of all the advantages and disadvantages therein. In doing so, new techniques can be developed that enhance or augment the existing schemes. This chapter provides background on current restoration mechanisms used in Synchronous Optical Network (SONET) [17] at the Physical Layer, Routing Information Protocol (RIP) [18], and Open Shortest Path First (OSPF) [19] at the Network Layer. Concluding the chapter is a brief introduction to graph theory as it pertains to the thesis.

2.2 Synchronous Optical Network

2.2.1 Introduction

Synchronous Optical Network (SONET) is a standard for digital transmission over fiber optic networks originally proposed by Bellcore and standardized by the American National Standards Institute (ANSI). An almost identical version, called Synchronous Digital Hierarchy (SDH) was published in Recommendations G.707, G.708, and G.709 by the ITU-T.

The lowest level in SONET is referred to as STS-1 (Synchronous Transport Signal, level 1) which operates at 51.84 Mbps. This is also called OC-1 (Optical Carrier level 1). By combining multiple STS-1 signals, higher level signals (STS-N) can be formed which will run at N times the speed of an STS-1 signal. In the case of SDH, these signals are called Synchronous Transport Modules (STMs). The lowest of these rates is referred to as STM-1 which operates at 155.52 Mbps which is equivalent to an STS-3 in SONET.

The SONET standard also defines the operations, administration, maintenance, and provisioning (OAM&P) procedures that are used to detect and maintain the network. These procedures also perform tasks such as configuration, performance evaluation, and customer support along with other functions.

2.2.2 Frame Format and Overhead

The smallest frame in SONET is the STS-1 frame. This frame consists of 810 octets that is transmitted once every 125 μs, which provides a data rate of 51.84 Mbps. The frame consists of 9 octets of Line Overhead (LOH), 18 octets of Section Overhead (SOH), 9 octets of Path Overhead (POH), and the remaining 774 octets are used for data (payload) as shown in Figure 2.1. Each frame is composed of 9 rows and 90 columns and is transmitted row by row over the network. The first 3 columns of an

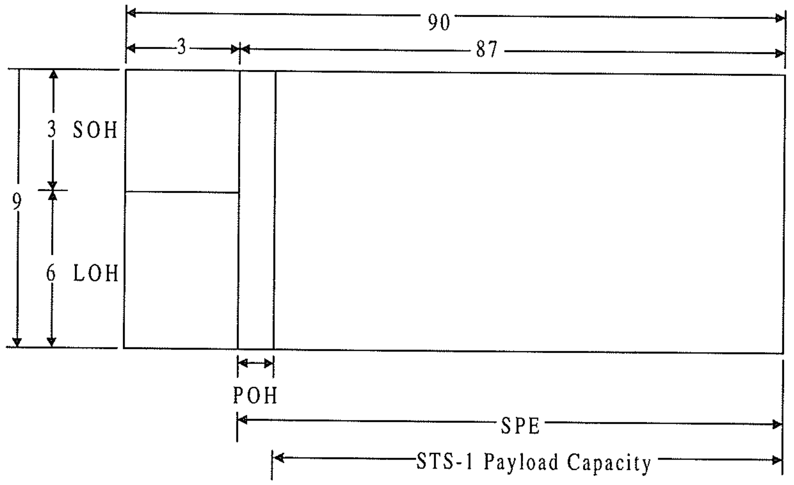


Figure 2.1: STS-1 Frame Structure

STS-1 frame are dedicated to the SOH and LOH. Together, the SOH and the LOH make what is called the Transport Overhead (TOH). The TOH uses the first 3 columns of the STS-1 frame which leaves 87 columns remaining. These 87 columns are referred to as the Synchronous Payload Envelope (SPE). The SPE contains both overhead and user data. The POH can begin in any byte location within the SPE. With this in mind, the SPE will typically overlap into the next frame. The SONET overhead bytes are shown in Figure 2.2. To construct higher level frames in the SONET hierarchy, simply increase the number of columns by some multiple of the STS-1 frame structure. For example, the structure of an STS-3 frame will still consist of 9 rows, but the number of columns will increase to $270 \times (3 \times 90)$. This implies that the STS-3 overhead columns

A1 Framing	A2 Framing	J0/Z0 STS-ID Trace/Growth
B1/undefined BIP-8	E1/undefined Orderwire	F1/undefined User
D1/undefined Data Com	D2/undefined Data Com	D3/undefined Data Com
H1 Pointer	H2 Pointer	H3 Pointer
B2/undefined BIP-8	K1/undefined APS	K2/undefined APS
D4/undefined Data Com	D5/undefined Data Com	D6/undefined Data Com
D7/undefined Data Com	D8/undefined Data Com	D9/undefined Data Com
D10/undefined Data Com	D11/undefined Data Com	D12/undefined Data Com
S1/Z1 Sync Status/Growth	M0 or M1/Z2 REI-L/Growth	E2/undefined Orderwire

Section And Line Overhead

J1 Trace
B3 BIP-8
C2 Signal Label
G1 Path Status
F2 User Channel
H4 Indicator
Z3 Growth
Z4 Growth
Z5 Tandem Connection

Path Overhead

Figure 2.2: SONET Overhead Bytes

are now 3 times wider than the STS-1 overhead leaving 261 $\times$ (3 $\times$ 87) columns for the SPE. The STS-3 frame structure as a whole consists of 2,430 bytes with a line rate of: (2430 bytes/frame) $\times$ (8000 frames/sec) $\times$ (8 bits/byte) = 155.52 Mbps

2.2.3 SONET Architecture

SONET is made up of different layers. These layers are used to help divide the communications task into smaller more manageable pieces. There are four layers in the SONET architecture, they are the Photonic layer, Section layer, Line layer, and Path layer as shown in Figure 2.3.

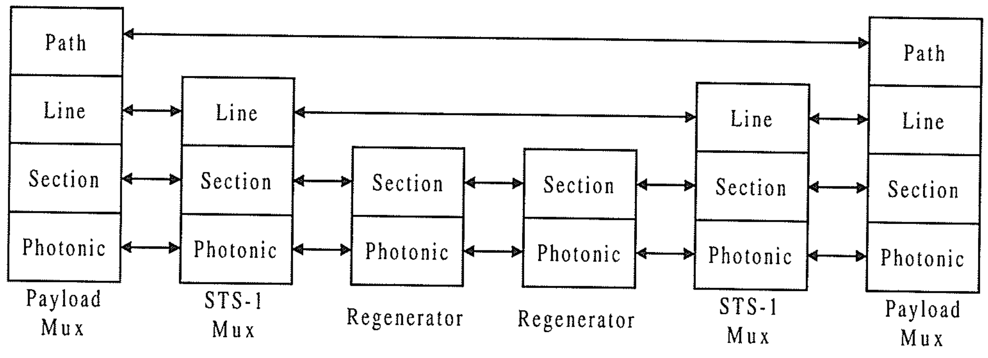


Figure 2.3: SONET Architecture Layers

2.2.4 SONET OAM&P

In this section we will focus more on the maintenance aspect of OAM&P [21] [23]. Maintenance is defined here as the steps that are taken to maintain a network in working order. These steps would include such things as detection of failures, generating alarms and signals, and restoring the network. There are many types of alarms and signals that go off when a failure is detected in the network. Some of these are Loss of Pointer (LOP), Loss of Signal (LOS), and Alarm Indication Signal (AIS). For a more comprehensive list of these alarms and signals, see Appendix A. After a failure has been detected, something needs to be done to correct it. There are techniques that exist that help to avoid the failure, but correction of the failure still needs to be done. Three techniques are discussed to reroute around failures that will help improve network integrity.

2.2.5 Survivable SONET Network Architectures

Automatic Protection Switching (APS) is shown in Figure 2.4(a). For this protection scheme, there is one protection fiber for N working fibers. It is called diverse protection because this scheme places the protection fiber in a physically diverse route. Compared to 1:N protection where the protection fiber is located physically in the same route as the working fibers, 1:N diverse protection is a better alternative. For higher survivability a 1:1 diverse protection scheme can be deployed. A 1:1 diverse protection will provide 100% restoration, but will require more resources than the 1:N diverse protection scheme.

Dual homing is a concept in which two hubs are assigned to each office, see Figure 2.4(b). In this approach, demand originating from the special office is split into two hubs, the home hub and the backup hub. In the case of a home hub failure, traffic is routed through the backup hub.

Self Healing Rings (SHRs) as shown in Figure 2.4(c) is an automatic protection scheme that provides 100% restoration capability. There are three types of SHRs currently being deployed and will be discussed later.

DCS mesh networks reroute traffic around a failure at the SONET path layer. As in Figure 2.4(d), traffic is rerouted through DCS #3 when a failure occurs on the link between DCS #1 and DCS #2. From this we can see that, a DCS mesh network does not require extra protection facilities to perform restoration. Instead, it uses the spare capacity on the working lines to restore the network. The DCS mesh network can be centralized or distributed. The one major fault of this system is that the time needed to communicate with the controllers to perform the rerouting is relatively large.

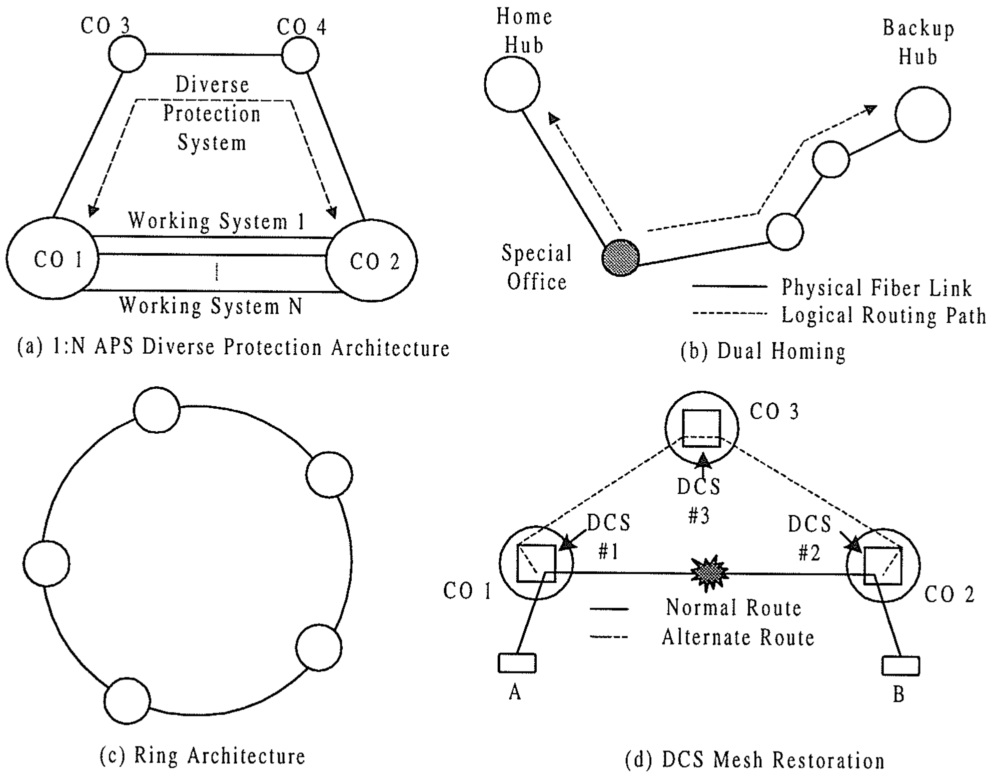


Figure 2.4: Survivable SONET Architectures

Automatic Protection Switching

This protection scheme is very fast. It can switch fibers in less than 50ms, using the K1/K2 bytes of the LOH. Figure 2.5 shows the steps that are taken to switch to the protection line when a fault is detected using APS.

1. The system is functioning properly along the working lines.

2. A fault occurs and the Tail End requests usage of the protection line using the K2 byte.

3. The Head End receives K2 for three consecutive frames and then bridges it's input to the protection line.

4. The Head End then sends a K1 and K2 byte to the Tail End saying that it has finished bridging and requests that the Tail End switches.

5. Tail End bridging:

   (a) The Tail End bridges its output of the protection line to the switch.

   (b) The Tail End bridges its input to the protection lines.

6. The Tail End sends a K2 byte to the Head End as an acknowledgement that it has finished switching.

7. The Head End bridges the output of the protection lines to the switch. The switch located at both ends will now select the protection line as the source for it's data. One other thing to note is that all K1/K2 byte signaling is done using the protection lines.

Self Healing Rings (SHR)

A SONET SHR can be unidirectional or bidirectional. Each of these types, USHR (unidirectional SHR) and BSHR (bidirectional SHR), can either be line switched or path switched. Line switched SHRs are triggered by the SONET line layer signals K1 and K2. Path switched SHRs are triggered by the SONET path layer signal Path AIS. BSHR can be further divided into 2-fiber BSHR and 4-fiber BSHR and are labeled as BSHR-2 and BSHR-4 respectively. Current deployment uses path switching for

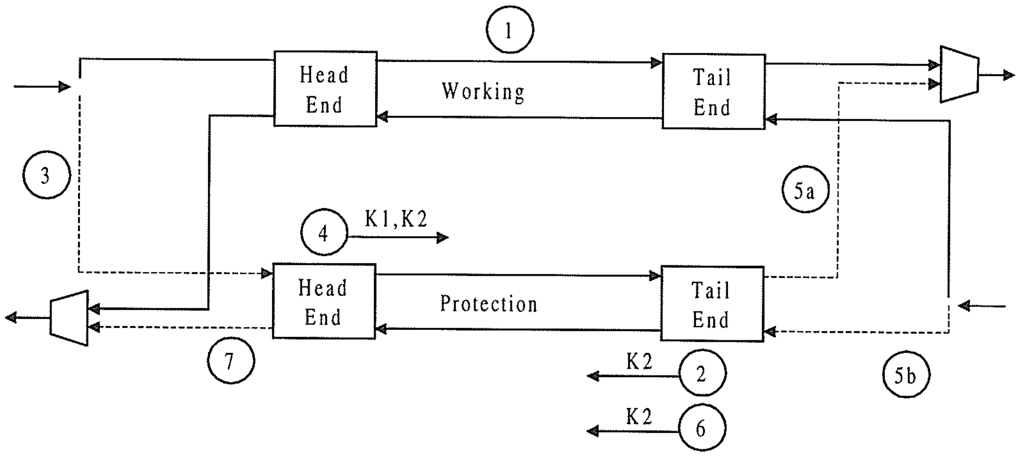


Figure 2.5: Protection Switching in APS

USHR and line switching for BSHR. Figure 2.6 shows how restoration is done for a path switched USHR (UPSHR).

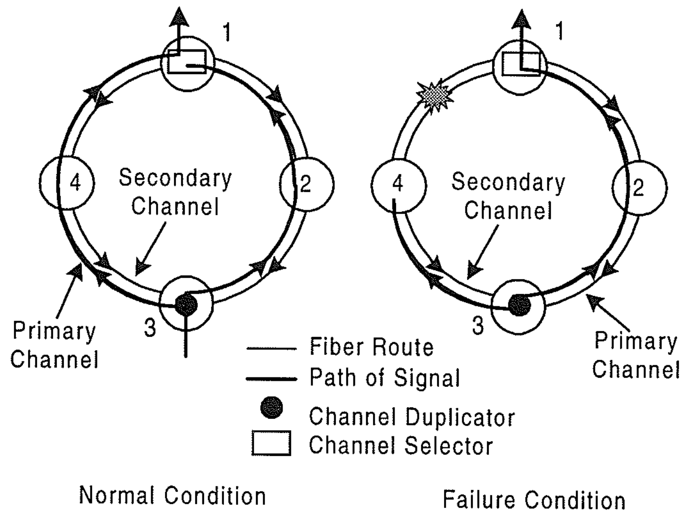


Figure 2.6: UPSHR under normal and failure conditions

A UPSHR has two fibers that form the ring. The inner ring is called the Secondary channel (or protection channel) with traffic running in a counter clockwise motion. The outer ring is called the Primary channel (or working channel) with traffic running in a clockwise motion. Though it is not shown in the figure, each node contains a channel duplicator and selector. The left ring of Figure 2.6 shows node 3 trying to send data to node 1. At node 3, the channel duplicator duplicates the data and the data is sent in both directions around the ring. This means that node 1 will receive two copies of the same data with different delays. So, the channel selector of node 1 will choose the data from the primary channel under normal circumstances and discard the data received on the secondary channel. In the event of a failure between nodes 4 and 1, node 1 will choose the data from the secondary channel because it is aware of the failure on the other link. Node 3 however, still transmits the data along both channels.

Figure 2.7 shows a line switched BSHR-2 (BLSHR-2) during normal and failure conditions. The BLSHR-2 is composed of two unidirectional rings. So in a BLSHR-2, data going from an end-node to another end-node will travel through the same intermediate nodes in both directions. Restoration is accomplished by reserving 50% of the capacity on both of the fibers. For example, suppose we have an Optical Carrier 12 (OC-12) BLSHR-2. The first six channels in both fibers will be used for normal operation. The other 6 channels, (7-12) will be used for protection. When a failure occurs on W1, data is moved from W1 to channels 7-12 on W2.

For the case of a BLSHR-4, the situations are almost identical. Instead of two fibers, we now have four fibers. Two working fibers (W1, W2) and two protection fibers (P1, P2). This will allow full capacity to be used on both working fibers unlike the BLSHR-2 where only half the capacity can be used so that full restoration is possible. Figure 2.8 shows the normal and failure conditions of a BLSHR-4. Under

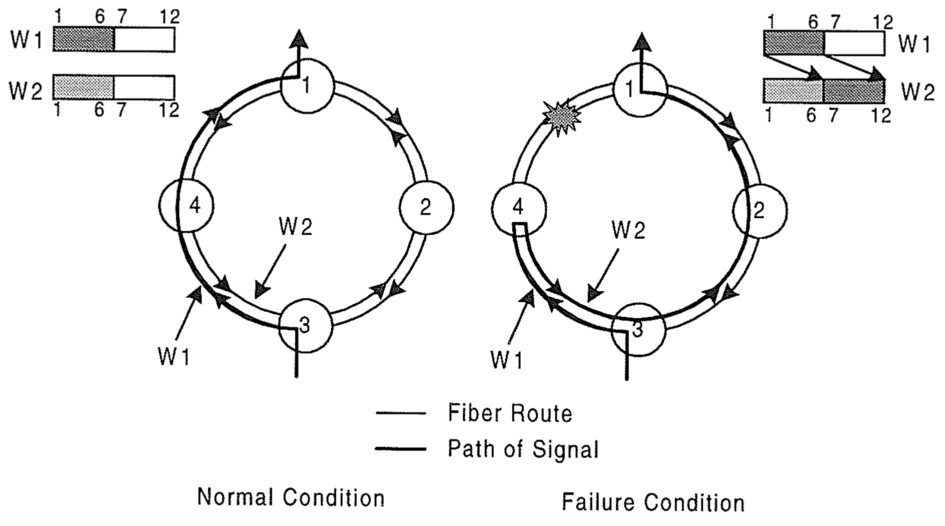


Figure 2.7: Normal and failure conditions for a BLSHR-2

normal conditions, traffic is sent on the working fibers. In the event of a fiber failure on W1 between nodes 4 and 1, data being sent from node 3 to 1 will be switched over to the protection fiber, P1, at node 4. This is similar to that of a BLSHR-2 shown in Figure 2.7. It is apparent that a BLSHR-4 will be more reliable than that of a BLSHR-2, because of the extra fibers. A BLSHR-4 can recover from multiple fiber cuts as well as span failures and so is the most reliable among SONET SHR alternatives.

DCS Mesh Networks

A network is called a mesh if each end to end connection can be supported through at least three link diverse routing paths. There are two types of restoration that can be performed depending on which layer is being used as the protection layer. Line restoration is performed if the line layer is being used and path restoration is performed if the path layer is used. Figure 2.9 shows both line restoration and path

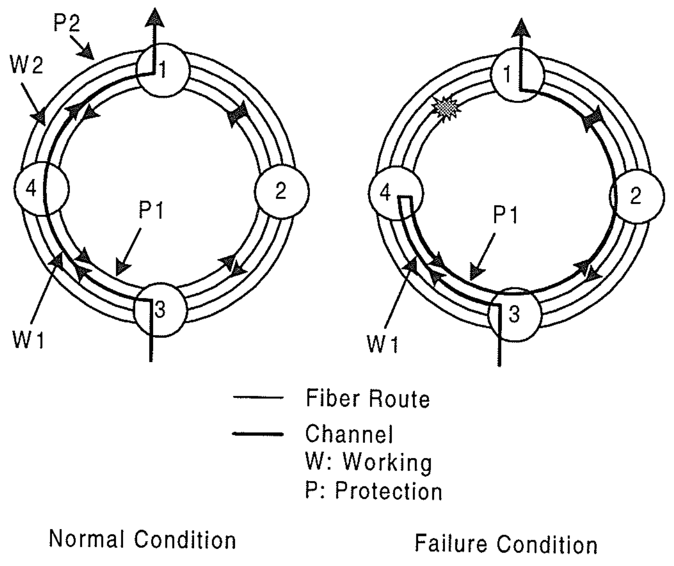


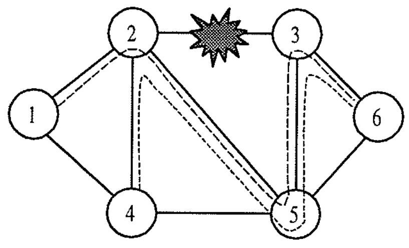
Figure 2.8: Normal and failure conditions for a BLSHR-4

restoration for a single span failure between nodes 2 and 3. As can be seen from Figure 2.9, line restoration finds an alternate path from node 2 to node 3 that will avoid the failure. But in the case of path restoration a new path is found from end-node to end-node. As mentioned before, this system can be centralized or distributed and is relatively slow compared with the other two schemes.

2.3 Network Layer Routing

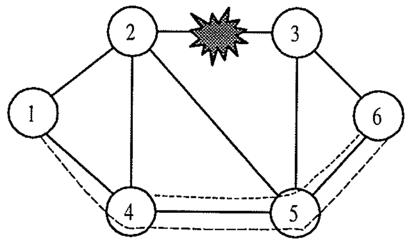
2.3.1 Introduction

There are many internet routing protocols in existence today such as: Routing Information Protocol (RIP), Interior-Gateway Routing Protocol (IGRP), Intermediate System-Intermediate System (IS-IS), and Open Shortest Path First (OSPF). These protocols can be classified into two classes. RIP and IGRP are both distance vector



STS-1 Paths	Original Path	Restored Path
(1,6)	1-2-3-6	1-2-5-3-6
(4,6)	4-2-3-6	4-2-5-3-6

(a) Line Restoration



STS-1 Paths	Original Path	Restored Path
(1,6)	1-2-3-6	1-4-5-6
(4,6)	4-2-3-6	4-5-6

(b) Path Restoration

Figure 2.9: Line and path restoration in a mesh network

routing protocols while IS-IS and OSPF are link state routing protocols. Distance vector routing uses an algorithm to determine the distance from a particular routing node to every other routing node in the network. Distance vector routing has many shortcomings such as counting to infinity (i.e. slow convergence) and the bouncing effect (i.e. routing loop) [18]. These shortcomings are addressed by link state routing protocols. In short link state routing protocols exchange link topologies instead of exchanges of distances. The exchange in topology between every routing node will result in a network map. Using this map, every routing node will compute a shortest path tree from itself to every other routing node in the network. Since every routing node has the same link state database (network map), link state protocols inherently avoid looping. This section will discuss the two most commonly used routing protocols, RIP in distance vector routing protocols and OSPF in link state routing protocols, with a main focus on the latter.

2.3.2 Routing Information Protocol

As the first widespread routing protocol, RIP is still very much in existence in today's networks. As such, it is important to discuss the pertinent issues related to restoration as addressed by RIP.

RIP is a distance vector protocol employing a distributed computation mechanism to manage it's routing table. Updates are sent from a router to all of its neighbours. These updates are received causing the neighbours to update their routing tables and send new updates to all their neighbours. This procedure will eventually converge as all the routing tables stabilize in the network. This would be ideal if the network topology never changed, but in practice, changes in network topology occur far too often.

Every RIP router sends an update message to all of its neighbours at 30 second intervals. If a router does not receive an update message from one of its neighbours within a 180 second interval, it will assume connectivity is lost. The route to this lost neighbour is then marked invalid and update messages continue to be sent out. The failed route will be updated with a valid one upon receiving a valid route from other neighbours. The main point to note is that there is a 180 second interval where connectivity is possibly lost. This is because the protocol must be certain that there is a failure before performing any changes on its routing table.

This 180 second loss in data transmission is motivation enough to warrant further research in restoration techniques. For the sake of brevity, we refer the reader interested in more information about the RIP protocol to RFC 2453 [18].

2.3.3 Open Shortest Path First

OSPF [20] [22] packets run directly over the IP Network Layer. The IP protocol number in the IP header is set to 89, which represents an OSPF packet. The Time To Live (TTL) field is almost always set to the value of 1 because OSPF packets only travel a single hop. The destination IP address is always set to the neighbours IP address, one of the multicast OSPF addresses, ALLSPFRouters (224.0.0.5), or ALLDRouters (224.0.0.6). All OSPF packets begin with a standard 24 byte header as shown in Figure 2.10. They include a field for OSPF version, packet type, length, area ID, checksum, authentication type, and authentication data.

There are five OSPF packet types as shown in Table 2.1. They are Hello packets,

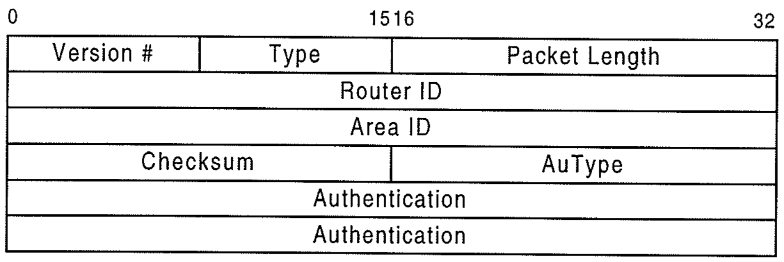


Figure 2.10: OSPF Packet Header

database description packets, link state request packets, link state update packets, and link state acknowledgement packets. The OSPF area ID enables the router to associate a packet to the proper level of the OSPF hierarchy.

All OSPF packet types deal with Link State Advertisements (LSAs) except the Hello packet. The Database Description, Link State Update, and Link State Acknowledge-

Table 2.1: OSPF Packet Types

ID	Type	Purpose
1	Hello	Used for neighbourhood discovery and maintenance
2	Database Description	Used to form adjacencies by passing a summary of its LSAs to the potential adjacent router
3	Link State Request	Used to request LSAs from neighbours to update or complete it's database
4	Link State Update	Used to transmit LSAs between routers either in response to a link state request or to flood new LSAs
5	Link State Acknowledgement	Used to acknowledge each LSA that is received to perform reliable flooding

ment packets all carry LSAs within their payload. Link State Request packets on the other hand are associated with LSAs by requesting them from neighbours. A brief description of the Hello packet and LSA header are given in the following sections. For a more detailed description of these headers and other OSPF packet types, refer to [19].

The Hello Packet

An OSPF packet of type 1 is known as a Hello packet. These packets are sent out periodically on all interfaces to establish and maintain neighbour relationships. Through the exchange of Hello packets, a router can determine its neighbours. A configurable parameter called the HelloInterval is the amount of time between the sending of two consecutive hello packets. The default for this parameter is 10 seconds. There is another timer called the RouterDeadInterval which plays a key role in OSPF restoration. This timer is defaulted to 4 times the amount of the HelloInterval value. The RouterDeadInterval is the amount of time that must have elapsed without seeing a hello packet from a neighbour before a router can declare its neighbour down. The reason that OSPF waits for 4 missed hello packets is the same as in RIP: to make certain that a failure exists before any routing calculations are performed. Although, by default, OSPF performs better than RIP in terms of the amount of time wherein there could be no transmission, it still suffers from high data loss during that time. There needs to be mechanisms in place that can either verify a fault or reroute data without modifying the routing table. This is addressed in Chapter 3.

The Hello packet is shown in Figure 2.11. The list of neighbours in the Hello

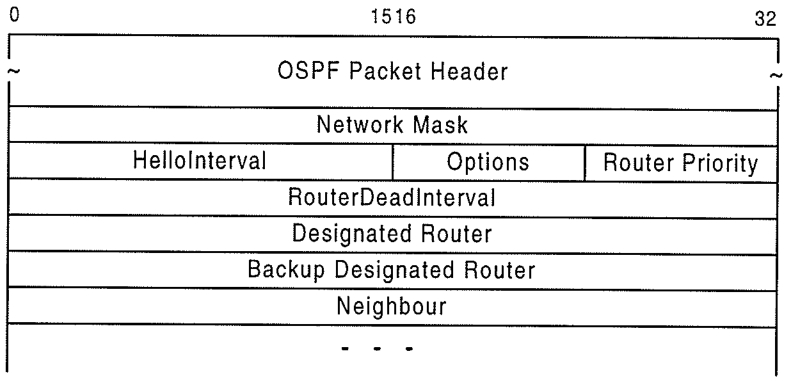


Figure 2.11: Hello Packet

packet are Router IDs of each router from whom a valid Hello packet has been seen recently (within the last RouterDeadInterval time). By updating this neighbour list, changes in the topology are noticed throughout the network.

Link State Advertisements

At the very center of a link state routing protocol such as OSPF, there is a distributed, replicated database. The database describes the topology of the network. Each router in the network will advertise its interfaces to others in the form of link state advertisements, or LSAs. These LSAs are distributed or flooded over the network by a process called reliable flooding. All together, these LSAs form the network topology in what is called the link state database. From this database, each router can calculate its IP routing table, which will be used to forward IP traffic.

Each OSPF router sends one or more LSAs to describe its local part of the routing domain. All LSAs will be used to form the link state database. In order to ensure proper use of the LSAs, extra information is sent with the LSA in the form of a header. There are three components of the header that will identify an LSA. They are the LS Type, Link State ID and the Advertising Router fields. There are occasions when multiple instances of the same LSA exist in the routing domain, and thus it it necessary to determine which one is more recent. This is accomplished by examining the LS age, LS sequence number, and LS checksum fields. The following subsections will give a quick overview of these fields in the Link State Advertisement header as shown in Figure 2.12.

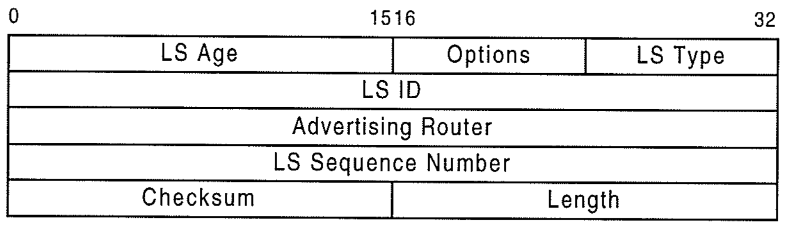


Figure 2.12: Link State Advertisement Header

LS Type Field

There are five LS types in the base OSPF specification which are categorized according to their functions.

1. Router-LSAs: each router sends one router-LSA describing its active interfaces and neighbours

2. Network-LSAs: this describes a network segment along with the identities of the currently attached routers

3. Network-Summary-LSAs: used in hierarchical routing within OSPF

4. ASBR-Summary-LSAs: used in hierarchical routing within OSPF

5. AS-External-LSAs: used in hierarchical routing within OSPF

Extensions can be made to OSPF by adding new types. Routers that do not recognize a type are not required to store or forward the LSA. This rule is maintained through the options field.

Link State ID Field

The Link Stated ID field uniquely identifies LSAs that the originating router sends from all other routers of the same type. This field can also carry addressing information such as the IP address of an external reachable network in the case of a type 5 LSA.

Advertising Router Field

This field is set to the router ID of the originator of the LSA. Routers are permitted to update and delete only self-originating LSAs.

LS Sequence Number

The LS sequence number field is used to determine which LSA is more recent where there are two instances of the same LSA. The instance with the larger sequence number is more recent. The meaning of larger depends on the organization of the sequence numbers. OSPF uses a signed 32-bit value for its sequence number. The first originated LSA is assigned the smallest negative number (0x80000001). Upon an update of the LSA, the sequence number is incremented by one. Eventually, the router must roll over the sequence numbers starting at the beginning again. To get the other routers to accept this new LSA, the originating router must first delete the LSA with S_{max} from the routing domain before flooding the new LSA. An LSA is not allowed to be updated more than once every 5 seconds as configured by the administrator. With this rule and with no errors, it would take more than 600 years for the sequence numbers to roll over.

LS Checksum Field

This field is a computed value which depends on the contents of the data in the LSA. It is used to detect data corruption within the header and the contents. It is originally calculated by the originator of the LSA and then recalculated by the receiver. If the checksum calculated by the receiving router does not match the value in the checksum field, then the LSA is discarded. A router also periodically verifies the checksums of all LSAs in its link state database to guard against hardware and software failures of its own. Once an LSA is originated, the checksum that was calculated will not be altered. This means that the LS Age field cannot be included in the checksum because the LSA Age field is modified in the flooding procedure.

LS Age Field

This field indicates the number of seconds since the LSA was originated. Under normal circumstances, the LS Age will be between 0 and 30 minutes. If the age reaches 30 minutes, the originating router will refresh the LSA by originating a new LSA. In the event that the originating router fails, the age filed will increment to a maximum value of 1 hour. At this time, the LSA will be deleted from the link state database. To remove the LSA from the database, the LSA is flooded at that time (1 hour mark). Note that if a router does go down, that the LSA still remains in the link state database. However, this entry will not be used in routing calculations because both sides must advertise a link before it can be used. Premature aging can be used to delete a LSA from the database. This can only be done by the originating router by setting the LSAs LS Age field to MaxAge and reflooding.

Options Field

This field can contain special instructions for the handling of LSAs during flooding or routing calculations. The base OSPF specification only identifies 2 option bits being used, but currently, 5 of the 8 bits have meaning as defined in RFC 2328 [19].

Length Field

This field contains the length of the LSA, including the header, in bytes. So there is a minimum of 20 bytes (header) to a maximum of over 65,000 bytes. Not 65535 because of the IP header.

Link State Database

Each router as mentioned before has an identical link state database. The databases are exchanged between neighbouring routers soon after they have discovered each other. After the exchange, the databases will be synchronized through reliable flooding.

OSPF Routing

From the link state database, an IP routing table can be created. Typically, an OSPF router uses Dijkstras Shortest Path First algorithm [20] to compute the shortest path tree from itself to every other router. From this, a router will know the entire path to every destination. However, the IP routing table only needs the next hop to a destination. In the case of multiple paths with the same cost, OSPF will store up to 6 of these paths.

Hierarchical Routing in OSPF

Hierarchical routing is partitioning a routing domain into smaller pieces, which in turn are grouped into levels. OSPF uses a two level hierarchical routing scheme through the use of OSPF areas. A 32-bit Area ID uniquely identifies each area. When an OSPF routing domain is split into subdomains, each subdomain is called an area. The OSPF backbone area must be connected to each of the subdomains. This backbone area is always given the Area ID of 0.0.0.0. Routing within an area is flat. That is, each router knows about every other router and about all network segments that are contained in the area. Detailed knowledge about the topology of an area is hidden from other areas because an area router-LSAs and network-LSAs are not flooded past the area borders. At the border on an OSPF area, there is a router called an area border router (ABR). An ABR is a router that is connected to two or more areas. Using OSPF summary-LSAs, ABRs can provide information from one area to another. ABRs advertise the addresses of their directly connected areas by sending summary-LSAs across the backbone. Each ABR then receives these summary-LSAs through flooding. After receiving these summary-LSAs, for each destination, an ABR examines all the summary-LSAs and chooses the best entry to be used in the routing table. The router then advertises these destinations to its own area in the form of another summary-LSA.

Hierarchical Routing Advantages and Disadvantages

Hierarchical routing has many advantages in OSPF. The first is the reduction in size of the link state database because only router and network LSAs for the area are contained in the database. Along with the reduction in database size, there is an accompanying reduction in the amount of network traffic needed to synchronize the database. Aggregation at the area boundaries reduces the size of the routing tables thereby decreasing the amount of time needed for a shortest path calculation.

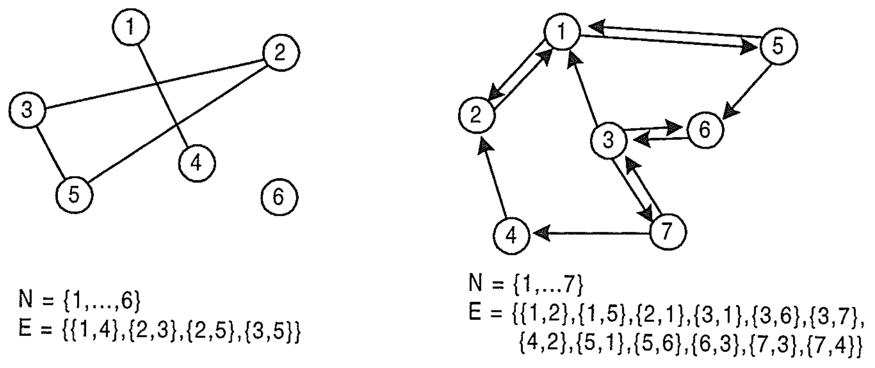
The major disadvantage to hierarchical routing stems from the splitting of the routing domains. This adds some amount of summary-LSAs to the routing calculations and database. As can be easily seen, the advantages far outweigh the disadvantages.

2.4 Graph Theory

This section will detail some of the common terms in graph theory as it relates to networking and the scope of this thesis. For more information into graph theory, see [24] [25].

2.4.1 Graphs

A *graph* is a pair $G = (N, E)$, where the set $N = \{n_1, n_2, n_3, ...\}$ is a finite set of elements called *nodes* and $E = \{e_1, e_2, e_3, ...\}$ is a finite set of elements called *edges*, where an edge is a connection between two nodes. Each edge in E is identified by a pair of nodes in N. Graphs can be either *directed* or *undirected.* A directed graph is a graph G such that the edges are identified by an ordered pair of nodes, otherwise G is undirected. Figure 2.13 depicts two graphs, one directed and one unidrected.



(a) Undirected (b) Directed

Figure 2.13: Example of directed and undirected graphs

2.4.2 Paths and Cycles

A *walk* in a graph $G = (N, E)$ is a finite sequence $W = n_0, e_1, n_1, e_2, n_2, e_3, \ldots, e_k, n_k$ of alternating nodes and edges. The beginning and ending nodes are such that n_{i-1} and n_i are the end nodes of edge e_i, $1 \leq i \leq k$. A walk may be denoted as $W = (n_0, n_k)$ with the walk being from the origin node n_0 to the terminus node n_k. Figure 2.14 shows an example of a walk.

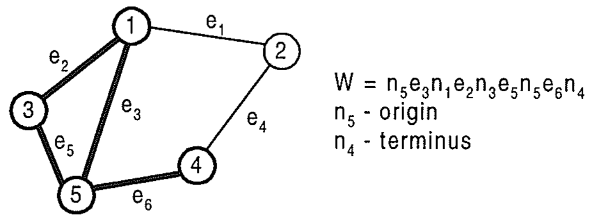


Figure 2.14: Example of a walk in a graph

A walk W is a *trail* if all of its edges are distinct. A trail is open if its end nodes are distinct; otherwise the trail is closed. An open trail is a *path* if all its nodes are distinct. Figure 2.14 is also an open trail but not a path since the node n_5 occurs twice.

Two nodes n_i and n_j are said to be connected in a graph G if there exists an edge $n_i - n_j$ in G $\forall i, j \in N$. Building on this, a graph G is said to be connected if there exists a path between every pair of nodes, otherwise it is called a disconnected graph. A closed trail is a *cycle* if all of its nodes except the origin and terminus are distinct. An *acyclic* graph is one that contains no cycles.

A *subgraph* $G1$ of G is a graph containing a subset of the nodes and edges that are contained in G.

2.4.3 Trees

A connected acyclic graph is also called a *tree*, where any two nodes are connected by one unique path. A subgraph $G1$ of G is called a tree if any two of the following statements are true:

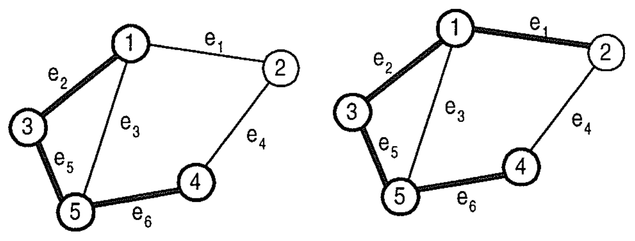
- subgraph $G1$ is connected

- $G1$ has no cycles

- the number of edges in $G1$ is k-1

where G has n nodes and G_1 has k nodes with $k \leq n$.

A spanning tree of graph G is a tree such that all nodes of G are contained in the tree. This leads to a spanning tree having $n - 1$ edges in it. An example of a tree and a spanning tree are shown in Figure 2.15.



$T = \{e_2, e_5, e_6\}$ $ST = \{e_1, e_2, e_5, e_6\}$

(a) Tree (a) Spanning Tree

Figure 2.15: Example of a tree and a spanning tree

2.5 Summary

Although SONET provides very fast restoration, it does not encompass Network Layer faults. With IPv4 being widely used and the emerging IPv6, it is necessary to provide robustness at the Network Layer. At the Network Layer, robustness or survivability is handled by the routing protocols. These protocols do have restoration mechanisms in place, but with ever increasing data transmission rates and exponentially increasing users, these mechanisms fall short of their intended purpose. Potentially large amounts of data can be lost while these protocols wait to ensure the validity of a fault. It is necessary if not essential to provide better restoration schemes for today's growing networks. Chapter 3 proposes a restorable spanning tree algorithm that can improve restoration response time.

Chapter 3

Problem Formulation for OSPF Restoration

3.1 Introduction

Both RIP and OSPF have suitable restoration mechanisms in place. It is not necessary to replace these mechanisms. Instead, these mechanisms should be enhanced or adapted to provide better performance. The method proposed here is to have a two-phase restoration scheme. The first phase is to use a proposed restorable spanning tree algorithm when a failure is first detected. The second phase is to use the default restoration mechanisms provided by the routing protocols. The combination of the restoration mechanisms provides for a more robust and survivable network. Our approach will use OSPF as the protocol for study.

3.2 Fault Detection

Restoration systems at the Physical Layer respond to the loss of physical transmission media, however, faults in IP networks are not necessarily as catastrophic. Congestion, software failures, and equipment configuration are some examples of non-catastrophic faults. For these types of faults, short time restoration and reconfiguration may be needed.

Fault detection is a major component of the restoration scheme. Having the ability to detect and verify the validity of a fault will greatly impact the restoration time. In OSPF, the OSPF Hello protocol is given the responsibility of detecting and verifying faults within the network. The Hello protocol broadcasts Hello packets every HelloInterval on each interface of a node. If a node does not receive a Hello packet from one of its neighbours within a timeout period called the RouterDeadInterval, the node assumes that its neighbour is now dead or a fault has occurred between the two. Typically, the RouterDeadInterval time is configured to be four times the value of the HelloInterval time. The reason for this is to verify that a fault is valid so that OSPF will not perform any unnecessary calculations. Determining the validity of a fault is necessary, but with a HelloInterval of 10 seconds, there will be a period of 40 seconds where communication is lost. With ever increasing transmission speeds, the amount of data lost within this time could be extremely high. A better approach is needed to detect faults within the network.

One approach is to detect the loss of carrier or loss of connection between two neighbours. Communication with lower level protocols will provide this detection ability which OSPF already supports. The OSPF Logical Link Down (LLDown) flag is set when indication from lower-level protocols determine that the neighbour connected to this interface is unreachable. With the use of this flag, immediate action can be taken when a loss of carrier occurs.

Another approach is to start recovery procedures after the first missed Hello packet. This method could be used for faults that are not as severe as a loss of carrier. However, recovery after one missed Hello packet could result in unnecessary routing calculations if the failure that caused the missed Hello packet is short lived. This is one of the reasons why OSPF requires that no Hello packets are to be received for the RouterDeadInterval before declaring a fault.

Both approaches have one common issue that needs to be addressed. The issue is that there may be a possibility that the occurring fault is very short. In both methods, shortest path first (spf) calculations would have taken place immediately after the LLDown flag was set or a missed Hello packet. When the temporary fault is resolved, another spf calculation would take place to return OSPF to its original state before the fault occurrence. One of the most common occurrences of a temporary fault would be an accidentally disconnected cable. OSPF has two parameters that can be configured to help with unnecessary spf calculations.

3.2.1 Adaptive SPF timers

A way of decreasing unnecessary spf calculations is to adjust timer values that are associated with spf calculations. There are two timers in particular, spf-holdtime and spf-delay. The spf-holdtime is the minimum amount of delay between two consecutive spf calculations. The spf-delay is the amount of time in which OSPF waits before doing an spf calculation upon receiving a notice of topology change.

If there are many topological changes within a short time, it is more practical to have a larger spf-holdtime so that only one spf calculation would be performed for a number of topology changes. However, having a large spf-holdtime value would delay restoration if the number of topology changes within a short time is small. Adjustment of the spf-delay timer will vary the response time of OSPF. With a small spf-delay time, spf calculations will occur quickly to restore the network. The downside to this is if the fault occurring was only for a short period of time, unnecessary calculations would have occurred.

To adjust the timers, there has to be some knowledge of the type of fault that is occurring. Even with this knowledge, it doesn't give much insight as to the duration of the fault. For the spf timers to be adaptive, a model will need to be created to predict the time duration of the fault. This seems rather cumbersome, which leads to our approach; two-phase restoration.

3.3 Two Phase Restoration

Our approach is to design a supervisory system which will monitor and control the underlying routing protocol. This system will be responsible for detecting faults within the network and taking appropriate action to handle the fault. Our approach focused in on the restoration mechanisms after a fault has been detected. Figure 3.1

shows how the two-phase restoration system works.

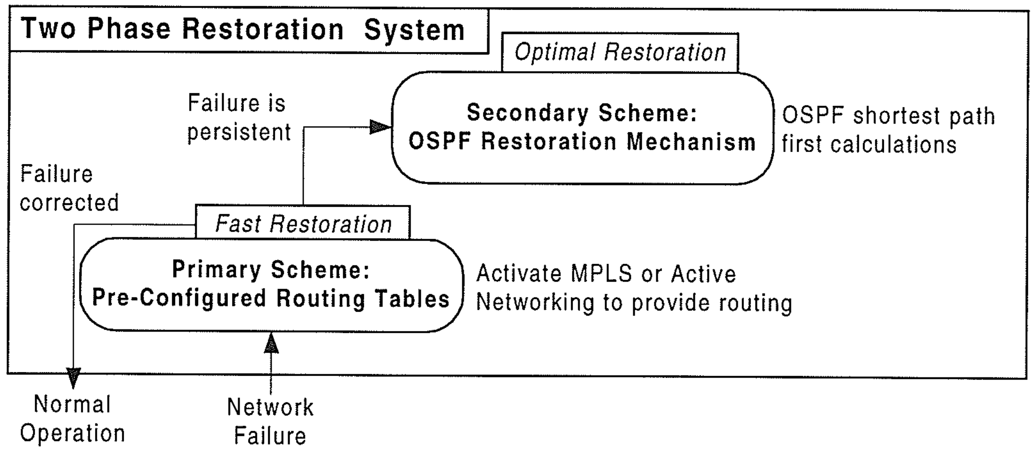

Two Phase Restoration System
Optimal Restoration
Secondary Scheme: OSPF Restoration Mechanism
OSPF shortest path first calculations
Failure is persistent
Failure corrected
Fast Restoration
Primary Scheme: Pre-Configured Routing Tables
Activate MPLS or Active Networking to provide routing
Normal Operation
Network Failure

Figure 3.1: Two Phase Restoration System

When a network failure is detected by some other means, the primary scheme is activated providing fast restoration to the network. If the network failure is short lived (less than the RouterDeadInterval Time), the primary scheme is deactivated. This returns OSPF to it's original state before the network failure. However, for more persistent failures, the secondary scheme is activated. This scheme is the OSPF restoration mechanism which will calculate an optimal shortest path for the network. The following section will give more details as to what takes place in the primary scheme.

3.3.1 Multiple Routing Tables

Our approach is to have routers within the network maintain preconfigured restoration tables. In the event of failure, for example carrier loss detection or a missed Hello packet, we propose a mechanism to switch between the original routing table to the restoration table. The preconfigured restoration tables and mechanism used to switch between tables together form the primary scheme for the two-phase restoration system. The secondary scheme would be OSPF's original restoration mechanism which occurs after the RouterDeadInterval time.

Two possible alternatives to handle the switching from the original routing table to the restoration table are discussed in our approach. These two possibilities are Multi-Protocol Label Switching (MPLS) and Active Networking. With both of these, the restoration table will be used to route packets in the event of a failure. Since this table is independent of the original OSPF routing table, there is no change of the OSPF routing table. Packets will continue to be rerouted using these MPLS or active networking until OSPF recalculates its routes to adjust for the link failure after the specified RouterDeadInterval time. Integration of our approach, MPLS and active networking with OSPF will be discussed in the following chapter.

Two main benefits can be seen with our approach over that of OSPF. The first and most important is that restoration time is greatly reduced as compared to OSPF having to wait a predefined RouterDeadInterval time before responding. With the primary scheme in place, communication can be re-established quickly by activating MPLS or active networking using the restoration table. The second benefit of reducing the number of spf calculations, stems from the fact that precalculated restoration tables are used. By using these tables, the original OSPF routing table is not modified. Since the original OSPF routing table is unaffected by the primary scheme, if there was a short period of failure (less than RouterDeadInterval), no spf calculations would have to take place because OSPF did not recognize that there was a fault in the network. The primary scheme would have handled all communications within the network during that time. MPLS or active networking would be deactivated once the failure is corrected and OSPF can continue normal routing operations. The problem is determining how many additional routing tables there need to be, and the table entries.

3.3.2 Multiple Spanning Trees

Let us recall what the entries in a routing table represent. Typically entries represent the next hop along the shortest path from the source to the destination. This is not the case for all routing protocols, but it is with regards to OSPF. By looking at the network as a whole, each individual routing table will contribute a small part to the overall routing topology.

That is to say, a complete path from source to destination can be determined by looking at the entries in all the routing tables. Completing the path for each source to destination pair using the routing tables will form a set of paths that are interconnected in some fashion. The only condition that needs to be satisfied is that the interconnection of the paths do not form cycles, as defined in the sense of graph theory. So, in essence, routing tables for a network form a connected acyclic graph which is better known as a tree. This however is not specific enough since a tree may not connect all the nodes in the graph. For the routing topology to be usable, all the nodes must be in the tree, or in other words, the tree must span the entire network.

This is commonly referred to as a *spanning tree.*

A definition of a spanning tree of a network is a subnetwork that contains all the nodes but only enough links ($N-1$ links, where N is the number of nodes in the graph) to form a tree. It is noted that a spanning tree of a network can potentially be used as a restoration path for any of the links not contained in the spanning tree. To further clarify, since there exists a path from each node to every other node in the network across the spanning tree, the failure of any of the links not contained in the spanning tree would not affect data flow through the network. Therefore we can conclude that the links not contained in the spanning tree are restorable.

Now consider a set of spanning trees that span the network. If there exists a spanning tree that does not contain a link (i, j), for all links, then the network is considered fully restorable. It will be assumed that all failures are considered to be link failures affecting both arcs between the adjacent nodes. It is quite trivial to find a set of spanning trees that meet the above condition, but we want to minimize the number of spanning trees while still providing maximum restorability. By choosing spanning trees to be as disjoint as possible, we are in effect restoring more links with fewer spanning trees. This can be done by minimizing the number of times a link appears in the set of spanning trees. The effect of this would be to load balance or spread out the spanning trees across the network in terms of the number of times a link is being used in the spanning tree set. Doing so would minimize the number of spanning trees that are needed because links that have not been restored yet will be chosen first in the creation of the next spanning tree.

3.4 Formal Problem Definition

The model developed in our approach is based on several assumptions. These assumptions are not too restrictive, but are reasonable. The following assumptions are made:

- A set of nodes is given for the network.
- A set of arcs which specify the interconnection of the nodes is given.
- A traffic matrix specifying traffic between all nodes is given.
- The number of routing tables to create is known.
- All failures are considered to be links failures.

The network is modeled as a graph $G = (N, A)$, where N is a set constituting the nodes in the graph and A is the set of directed arcs which specify connectivity of the nodes in N. Let us define the difference between an arc and a link. A link (i, j) represents a bidirectional connection between nodes i and j which is composed of two directed arcs, (i, j) and (j, i). To maximize restoration, it is sufficient to minimize the number of times a link appears in the spanning tree set. Let us first define some variables that are to be used in the formulation:

- T represents the number of spanning trees to create with t being the t^{th} spanning tree in the set of spanning trees.
- The pair (i, j) represent a link (i, j) composed of two directed arcs from A.
- o and d represent the origin and destination of a path.

Let w_{ij}^t be the elements of W representing whether or not an arc is being used inthe set of spanning trees, where

$$w_{ij}^t = \begin{cases} 1, & \text{if } \sum_{o \in N} \sum_{d \in N} x_{ij}^{tod} > 0; \\ 0, & \text{otherwise.} \end{cases}$$

with x_{ij}^{tod} is defined as the following:

$$x_{ij}^{tod} = \begin{cases} 1, & \text{if link } (i,j) \text{ is used in a route table } t \\ & \text{and is along the path for the } od \text{ pair;} \\ 0, & \text{otherwise.} \end{cases}$$

The objective of the algorithm is to find the x_{ij}^{tod} elements which will lead us in determining the w_{ij}^t components of the W. This is of course taking into consideration the capacity constraints c_{ij} of the network as well as the flow constraints f_{ij}^t.

The problem can now be stated as follows.

3.4.1 Problem P

$$\textbf{Minimize } d = \sum_{i,j \in A} \left(\sum_{t} w_{ij}^{t} \right)^{2} \tag{3.1}$$

subject to

$$\sum_{j \in N} x_{ij}^{tod} - \sum_{k \in N} x_{ki}^{tod} = b^{tod}(i), \qquad \forall o \in N, \forall d \in N, \forall i \in N, \forall t \in T \tag{3.2}$$

$$f_{ij}^{t} \leq c_{ij}, \qquad \forall i,j \in A, \forall t \in T \tag{3.3}$$

$$\sum_{\forall i,j \in A} w_{ij}^{t} \leq 2 * (|N| - 1), \qquad \forall t \in T \tag{3.4}$$

$$w_{ij}^{t} - 1 \leq y_{ij}^{t}, \qquad \forall i,j \in A, \forall t \in T \tag{3.5}$$

$$1 - w_{ij}^{t} \leq y_{ij}^{t}, \qquad \forall i,j \in A, \forall t \in T \tag{3.6}$$

$$\sum_{o \in N} \sum_{d \in N} x_{ij}^{tod} \leq M(1 - y_{ij}^{t}), \qquad \forall i,j \in A, \forall t \in T \tag{3.7}$$

$$w_{ij}^{t} \leq \sum_{o \in N} \sum_{d \in N} x_{ij}^{tod}, \qquad \forall i,j \in A, \forall t \in T \tag{3.8}$$

$$w_{ij}^{t} = w_{ji}^{t}, \qquad \forall i,j \in A, \forall t \in T \tag{3.9}$$

$$f_{ij}^{t} = \sum_{o \in N} \sum_{d \in N} x_{ij}^{tod} e^{od}, \qquad \forall i,j \in A, \forall t \in T \tag{3.10}$$

$$w_{ij}^{t} \in 0, 1, \qquad \forall i,j \in A, \forall t \in T \tag{3.11}$$

$$x_{ij}^{tod} \in 0, 1, \qquad \forall o \in N, \forall d \in N, \forall i,j \in A, \forall t \in T \tag{3.12}$$

$$y_{ij}^{t} \in 0, 1, \qquad \forall i,j \in A, \forall t \in T \tag{3.13}$$

$$c_{ij}, f_{ij}^{t}, e^{od}, t \geq 0 \qquad \forall o \in N, \forall d \in N, \forall i,j \in A, \forall t \in T \tag{3.14}$$

In this formulation, Constraint 3.2 is the connectivity constraint. It states that each node along the path for an *od* pair, it is equal to 0 if it is an intermediate node, 1 if the node is the origin and -1 if the node is the destination. In other words:

$$b^{tod}(i) = \begin{cases} 1, & \text{if } i = o; \\ -1, & \text{if } i = d; \\ 0, & \text{otherwise.} \end{cases}$$

This connectivity constraint also satisfies the flow conservation law by guaranteeing that all nodes must be connected in some fashion. This does not imply that the actual capacity demand of the flow is being satisfied, but merely ensures that there is a path for each *od* pair. Ensuring that the capacity demands of the flows are being satisfied is governed by constraint 3.3.

Constraint 3.3 represents the capacity constraint of the network. The element c_{ij} is the total available capacity of arc (i, j) and f_{ij}^{t} is the total amount of traffic flowing through arc (i, j) for routing table t. In this formulation, a unit of capacity can represent a T1 link or an OC-3 link depending on the network. For example, if all connections in a network used T1 links, then an arc (i, j) with capacity 3, can be represented by 3 separate arcs of unit capacity spanning nodes i and j. It is necessary to note that although a single arc can be represented by many smaller links, it does not mean that there are physically more arcs spanning two nodes. The representation is merely to provide an abstraction in dealing with the capacity constraint.

Assuming that traffic flows between all nodes, Constraint 3.4 combined with Constraint 3.2 guarantee that the network will be configured in a spanning tree. If the network was not a spanning tree, then the connectivity constraint will be violated and consequently, a routing table can not be generated from a network topology that contains cycles.

Constraints 5 through 8 define the relation between the decision variables x_{ij}^{tod} and w_{ij}^t. The definition of w_{ij}^t can be restated as the following non-linearity:

$$w_{ij}^t = 1 - \prod_{o,d}(1 - x_{ij}^{tod}), \qquad \forall i,j \in A, \forall t \in T. \tag{3.15}$$

This is very similar to an If-Then constraint [26] in integer programming, where if one constraint is satisfied, then the other constraint must be satisfied. Using this approach, definition 3.15 can be restated as Constraints 3.5 through 3.8, where $y \in 0, 1$ and M is equal to the number of possible origin destination pairs.

$$M \leq |N| \cdot |N - 1|.$$

Looking at Constraint 3.7, for the left-hand side of the constraint to be positive (an arc (i, j) is being used by an *od* pair), y_{ij}^t must be equal to 0. This leads to the conclusion that $w_{ij}^t = 1$ from Constraints 3.5 and 3.6, that is

$$w_{ij}^t - 1 \leq 0 \Rightarrow w_{ij}^t \leq 1$$

$$1 - w_{ij}^t \geq 0 \Rightarrow w_{ij}^t \geq 1.$$

If the left-hand side of Constraint 3.7 is 0 (arc (i, j) is not being used by any *od* pair), then the value of y_{ij}^t must be 1. This leads to the following two equations from Constraints 3.5 and 3.6:

$$w_{ij}^t \leq 2$$

$$w_{ij}^t \leq 0$$

From these two equations, w_{ij}^t is less than or equal zero. To force equality, Constraint 3.8 is used. These four constraints (3.5 to 3.8) change the nonlinear constraint 3.15 to a set of linear constraints.

For the flow variable f_{ij}^t, it can be defined as stated in Constraint 3.10, where e_{od} is specified by the traffic matrix E. The traffic matrix is a square $n \times n$ (where $n = |N|$) matrix, with each element representing the average amount of traffic flowing from node i to node j.

There is one question left unanswered, and that is with regards to the value of T. T represents the number of routing tables that are needed. If the value of T is chosen to be too small (less than the minimum), then maximum restorability will not be achieved. On the other hand, if the value of T is chosen to be large (greater than the minimum), then there will be unnecessary routing tables created.

Some results that were obtained from this integer programming problem are discussed in the following chapter.

3.5 Restorable Spanning Tree Algorithm Heuristic

With the size of the integer programming problem increasing exponentially with respect to the number of nodes, the computation time needed to compute the optimal solution also increases exponentially. Although our approach is designed to minimize real time computation, faster computation is always preferred. It is for this reason that a heuristic algorithm is proposed as an alternative to compute the spanning trees. This section will discuss the Restorable Spanning Tree Algorithm (RSTA) heuristic with no implementation details.

The RSTA heuristic takes advantage of the fact that we are looking for multiple spanning trees. With this knowledge, it is possible to code an algorithm that can iteratively solve the problem, whereas the integer problem computes the spanning trees in parallel. From this iterative solution, the RSTA will compute the minimum number of routing tables, T, that are needed. This is more efficient than the integer program where the value of T is chosen without knowing whether it is minimal or not. To determine if the chosen T is minimal, the integer program must be solved multiple times with varying values of T until restoration is maximized.

At a high level, the algorithm first creates a spanning tree for the network. This spanning tree would restore all the edges that are not contained in it. Another way of saying this is that this spanning tree contains all the edges that still have to be restored. The objective now is to create an additional spanning tree that does not use any edges from the first spanning tree. If this is the case, then only two spanning trees are needed to restore the complete network from single edge failures. If it is not possible to find two disjoint spanning trees, then it is necessary to use edges from the first spanning tree to create the second one. This process continues until all edges are restored in the network. With each new spannning tree created, different edges from the first spanning tree are used. Eventually, the algorithm will converge either by restoring all the edges, or reaching a saturation level in which maximum restorability is reached in the network. The case studies presented later in this chapter will further clarify and provide more insight into the heuristic.

With the knowledge that a single spanning tree restores all edges that are not in the spanning tree,

Let graph $G_i = (N_i, E_i)$ represent a network for iteration i, where N_i is the set of nodes in the network and E_i be the set of remaining edges in the network. N_0 and E_0 are equal to the set N and E respectively. Each spanning tree that is calculated for iteration i, is stored in M_i. Variable R_i is the set of restored links with $R_0 = \{\}$. Variables SG_j and SM_j represent subnetworks and their corresponding spanning tree respectively. Finally, the variable T is the number of spanning trees that have been created.

The Restorable Spanning Tree Algorithm (RSTA) heuristic is as follows.

```
1. set $i = 0, T = 0$
2. $M_0 =$ spanning tree of $G_0$
3. $T = T + 1$
4. $E_1 = E_0 - M_0$
5. $R_1 = E_0 - M_0$
6. Find $M_1 =$ spanning tree of $G_1$ if possible
7. if $M_1$ exists
    (a) $T = T + 1$
    (b) $R_2 = R_1 + (E_0 - M_1)$
    (c) 2 disjoint spanning trees found, terminate
8. else the graph has been subdivided into smaller subgraphs
```

(a) For each subgraph SG_j, find it's spanning tree SM_j

(b) $i = i + 1, T = T + 1$

(c) Choose links from previous $M_k, k < i$ spanning trees, such as to connect the subgraph SM_j to complete a spanning tree for G_i. The links are chosen such that the number of times an edge is used in the spanning tree set are as low as possible.

(d) $E_{i+1} = E_i - M_i$

(e) $R_{i+1} = R_i + (E_0 - M_i)$

(f) if $(R_{i+1} == E_0)$ then

   i. all links are restored, terminate

(g) else if $(R_{i+1} == R_i)$ then

   i. Maximum restoration reached, terminate

(h) else revert back to G_1 and go to step 8(b)

After initialization, the algorithm starts by computing a spanning tree, M_0, of the graph G_0. Steps 4 and 5 update the edges that remain in the graph, E_1, and the currently restored edges, R_1. If a spanning tree can be created from G_1, then the algorithm will terminate because two disjoint spanning trees have been created, thereby restoring all links.

If a second spanning tree could not be created, then graph G_1 is not a connected graph. In other words, the graph has been divided into multiple subgraphs. Step 8(c) connects these subgraphs together using edges from previous spanning trees($M_k, k <$

i). The edges from the previous spanning trees are selected in a way as to minimize the number of times the edge is used in the spanning tree set. This can be easily done by associating a counter value with each of the edges in the graph.

Steps 8(d) and 8(e) are used to update the edges and remaining edges of the graph. The next couple of steps of the algorithm cover the termination criterion. The first possible termination criterion is that all links are restored with the current number of spanning trees. This is accomplished in step 8(f) by testing the equivalence of the restoration set R_i with the original set of edges E_0. The second possibility is that the network is not fully restorable, in which case R_i is equivalent to R_{i-1}. Finally, if any of the termination criterion are not satisfied, the algorithm continues at step 8(a) with the remaining edges E_i forming a new graph $G_i = (N, E_i)$.

3.6 Summary

Network failures come in a variety of flavours and having the ability to detect and correct them are mandatory for any robust network. This chapter detailed the formulation for the integer programming problem that will provide for fast restoration of single link failures. Our approach in handling these failures is composed of a two step process. The first step is to have a set of preconfigured routing tables that can be used for fast restoration prior to OSPF's restoration mechanism. This first step provides connectivity for the network during short outages without any unnecessary routing table calculations. For more prolonged network failures, our approach provides communication between nodes during OSPF's RouterDeadInterval time.

The solution to the integer programming problem provides an optimal set of spanning trees that can be used as preconfigured routing tables. With the size of the problem growing exponentially with respect to the number of nodes a heuristic is proposed to speed up computation. The following chapter will discuss the results that were obtained from the integer program as well as other experiments that were done with respect to restoration.

Chapter 4

Computational Results and Experiments

4.1 Introduction

This chapter discusses some of the results that were obtained through solving the integer programming problem for numerical examples. It also provides a case study for the Restorable Spanning Tree Algorithm (RSTA) heuristic. This chapter also discusses some other experiments that were performed with respect to restoration.

4.2 Integer Program Solution

The integer program as discussed in Chapter 3, was solved using an implicit enumeration algorithm package called "opbdp" [27]. This package was specifically designed for solving linear 0-1 optimization problems with integer coefficients.

4.2.1 Spanning Tree Generator

The opbdp software package is a generalization of the Davis-Putnam enumeration method for linear pseudo-Boolean inequalities. This is essentially a logic-based implicit enumeration technique. For more information concerning the generalization and implicit enumeration see [26] [28].

The package was used as a means of solving the integer programming problem without looking into efficiency and speed of the computation, where efficiency refers to the speed and size of the problem itself. Further experiments would have to be done to improve the overall computation time while using this package as well as testing other software packages. Aside from actual computation time, changes in the formulation could be done to reduce the size of the problem, thereby reducing the computation time. This is discussed in Chapter 6.

The opbdp package requires that all constraints to the optimization problem be written out explicitly, which is very time consuming with larger problems. The Spanning Tree Generator (STG) software is a front end to the opbdp package. It provides a graphical user interface that allows users to draw a network diagram. From this diagram, STG can either create the configuration file to be read by opbdp, or execute opbdp on the diagram to produce the set of spanning trees. A screenshot of STG is shown in Figure 4.1 with two spanning trees generated.

STG is implemented in Java using the Java Software Developer's Kit 1.2.2 (SDK) running on a Linux Platform. With the opbdp package being available for many

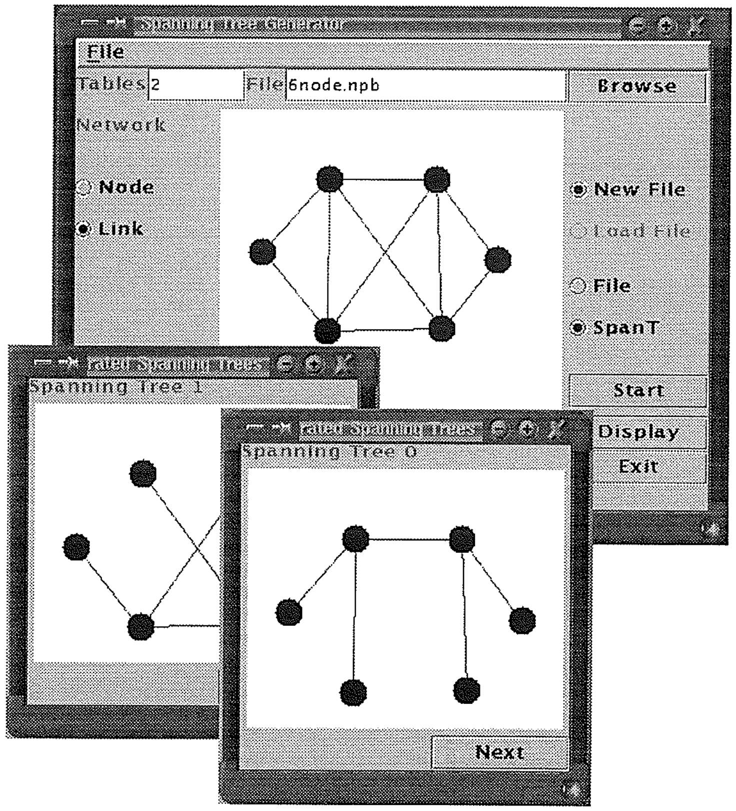


Figure 4.1: Spanning Tree Generator Screenshot

platforms, Java was chosen because of its platform independent capabilities.

4.2.2 Computational Results

This section discusses some of the results that were obtained using STG and the opbdp package. The example network used was a six node network connected in various configurations to illustrate certain scenarios that may arise. These example networks are shown in Figure 4.2.

Figure 4.2(a) is configured in a ring topology which will require the number of spanning trees be equal to the number of links in the network. For this example, the

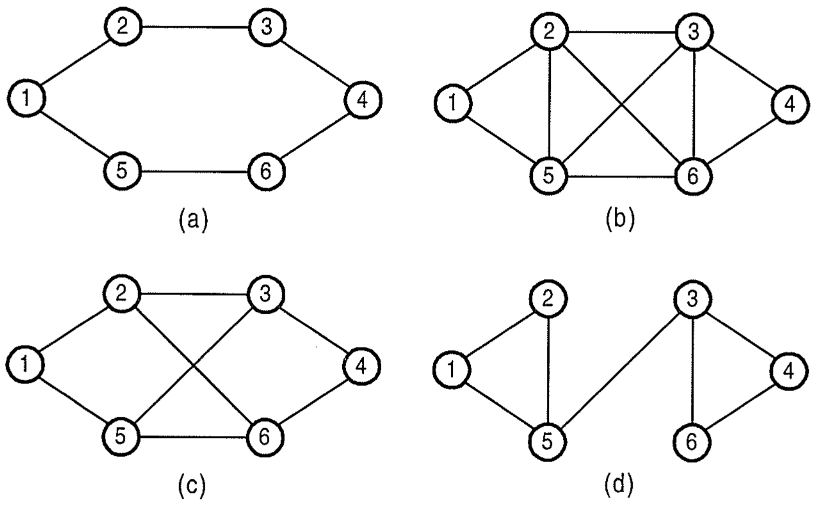


Figure 4.2: Example Networks

value is six. Since $|N| - 1$ links are required for a spanning tree, only one link can be restored for each spanning tree. This topology will require the most spanning trees for full restoration capability.

Figure 4.2(b) is configured such that only two spanning were required for full restoration. To generalize, any fully connected network with the number of nodes greater than three will require only two spanning trees for full restoration. Figure 4.2(b) is an optimal network topology requiring only $2 \times (|N - 1|)$ arcs, which is exactly the minimum number of links needed for two spanning trees.

Figure 4.2(c) will require more than two spanning trees for full restoration because it doesn't contain at least $2 \times (|N| - |)$ links. It is sufficient to say that if a network does not contain at least $2 \times (|N| - 1)$ links, then more than two spanning trees are needed for full restorability. However, a network having $2 \times (|N| - 1)$ or more links does not imply that only two spanning trees are needed for full restorability. In the case of Figure 4.2(c), three spanning trees are needed for full restoration.

Figure 4.2(d) illustrates the case where a network is not fully restorable. It's easily seen that link $(3, 5)$ is needed for all spanning trees that are created for this network. In this case full restorability can not be reached, but maximum restorability is achieved with three spanning trees.

Table 4.1 gives a comparison between these network topologies. For a network

Table 4.1: Comparison between different network topologies

Network	Minimum	Trees	Single Link Restoration	Double Link Restoration
A	180	6	100%	0%
B	20	2	100%	44.4%
C	58	3	100%	32.1%
D	66	3	85.7(max)%	14.3%

with six nodes and 20 arcs(10 links), the optimal value for full restoration would be 20 (10 arcs per spanning tree). This is achieved with network B. With a ring topology, network A required six spanning trees to reach full restoration with a minimum objective function value at 180. This suggests that network topology plays a significant role in restoration.

Although the integer programming problem was not formalized to account for double links failures, it does restore some double links failures. Consider, L to be the number of links in the network and N to be the number of nodes.

$$L-(N-1) = \#\text{ of links restored by one spanning tree} \tag{4.1}$$

$$\binom{L-(N-1)}{2} = \#\text{ of double link failures recovered by one spanning tree} \tag{4.2}$$

$$\binom{L}{2} = \#\text{ of possible double links failures} \tag{4.3}$$

$$\frac{\binom{L-(N-1)}{2}}{\binom{L}{2}} = \%\text{ of double link failures recovered by one spanning tree} \tag{4.4}$$

$$\binom{0}{2} = 0 \tag{4.5}$$

$$\binom{1}{2} = 0 \tag{4.6}$$

For multiple spanning trees, equation 4.4 can be generalized to

$$\frac{\binom{L-(N-1)}{2}}{\binom{L}{2}} + \alpha = \%\text{ of double link failures recovered by one spanning tree} \tag{4.7}$$

$$\alpha = \frac{1}{\binom{L}{2}} \sum_{i=2}^{T} \alpha_i \tag{4.8}$$

$$\alpha_i = \binom{N-1-\beta}{2} \tag{4.9}$$

where, α_i is the disjoint factor for spanning tree i, T is the number of spanning trees, and β is smallest number of links that this spanning tree has in common with any other spanning tree. The term α represents the disjoint factor for all spanning trees. Note that the summation in equation 4.8 is over $i = 2$ to T, not $i = 1$. From these equations, the percentage of double link failures that can be restored is calculated and shown in Table 4.1. Chapter 6 will propose some changes that could be done to provide better restoration for double link failures.

4.3 RSTA Heuristic Case Study

As mentioned in the previous section, there are three possible cases that can occur. The simplest case is when only two spanning trees are needed for full restoration. The second case deals with the situation where the network is divided after removing the links from the first spanning tree thereby requiring more than two spanning trees. Finally, there is the case wherein the network is not fully restorable.

This case study will provide an illustration of how the RSTA algorithm executes. The purpose here is to help readers understand the algorithm. The study will use three of the example networks shown previously in Figure 4.2, one to illustrate each case.

4.3.1 Case 1: Two Disjoint Spanning Trees

The first case is the simplest of the three. It occurs when after computing the first minimal spanning tree, the remaining links also span the network nodes. This leads to full restorability with just two spanning trees.

Using the notation from the description of the RSTA algorithm in Chapter 3, Figure 4.3(a) shows the initial graph $G_0 = (N, E_0)$ with E_0 and R_0 being set to their initial values. Recall that E_i is the set of remaining links in the network as spanning trees are created therefore E_0 is equal to all the links in the network since no spanning trees have been created yet. R_i is the set containing the links that have been restored, with R_0 being equal to a null set.

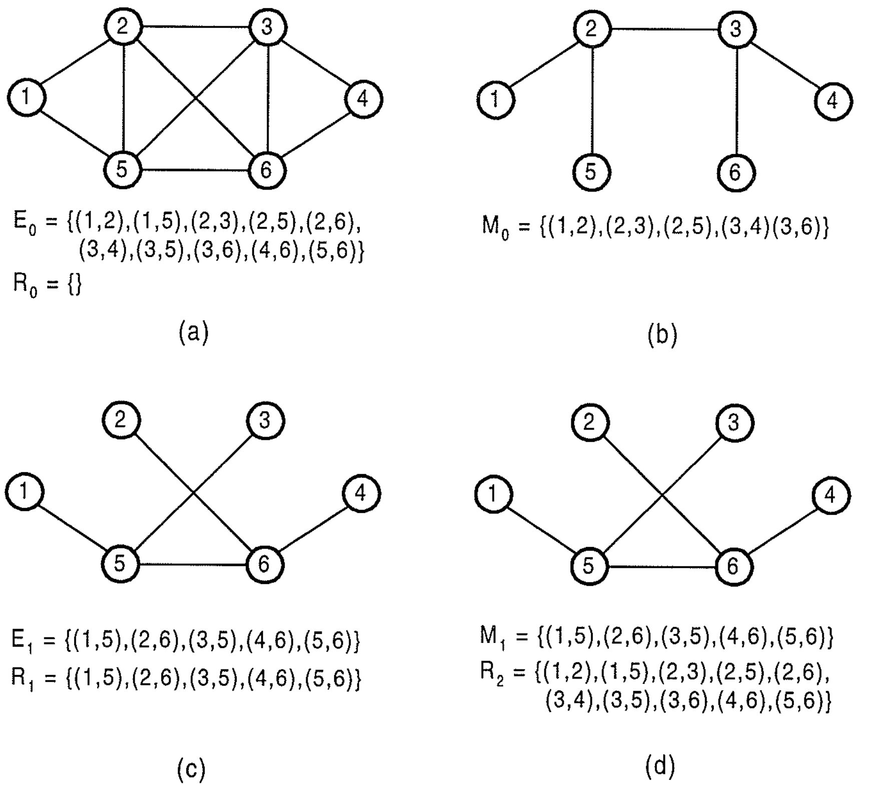

E_0 = {(1,2),(1,5),(2,3),(2,5),(2,6), (3,4),(3,5),(3,6),(4,6),(5,6)}
R_0 = {}
(a)
M_0 = {(1,2),(2,3),(2,5),(3,4)(3,6)}
(b)
E_1 = {(1,5),(2,6),(3,5),(4,6),(5,6)}
R_1 = {(1,5),(2,6),(3,5),(4,6),(5,6)}
(c)
M_1 = {(1,5),(2,6),(3,5),(4,6),(5,6)}
R_2 = {(1,2),(1,5),(2,3),(2,5),(2,6), (3,4),(3,5),(3,6),(4,6),(5,6)}
(d)

Figure 4.3: RSTA Heuristic: Case 1

The RSTA algorithm starts by creating a spanning tree of graph G_0, M_0 shown in Figure 4.3(b). Calculating E_1 and R_1 leads to the graph G_1 shown in Figure 4.3(c). The RSTA algorithm continues by finding a spanning tree M_1 for graph G_1. With two disjoint spanning trees found, the algorithm is complete providing full restorability with the minimum amount of trees possible. This is not always the situation as discussed in case 2 of the case study.

4.3.2 Case 2: Multiple Spanning Trees Needed

The second case, occurs when a second spanning tree can not be found directly from the remaining links. This is because, the graph has been divided into subgraphs after removing the links of the first spanning tree. The first three diagrams of Figure 4.4

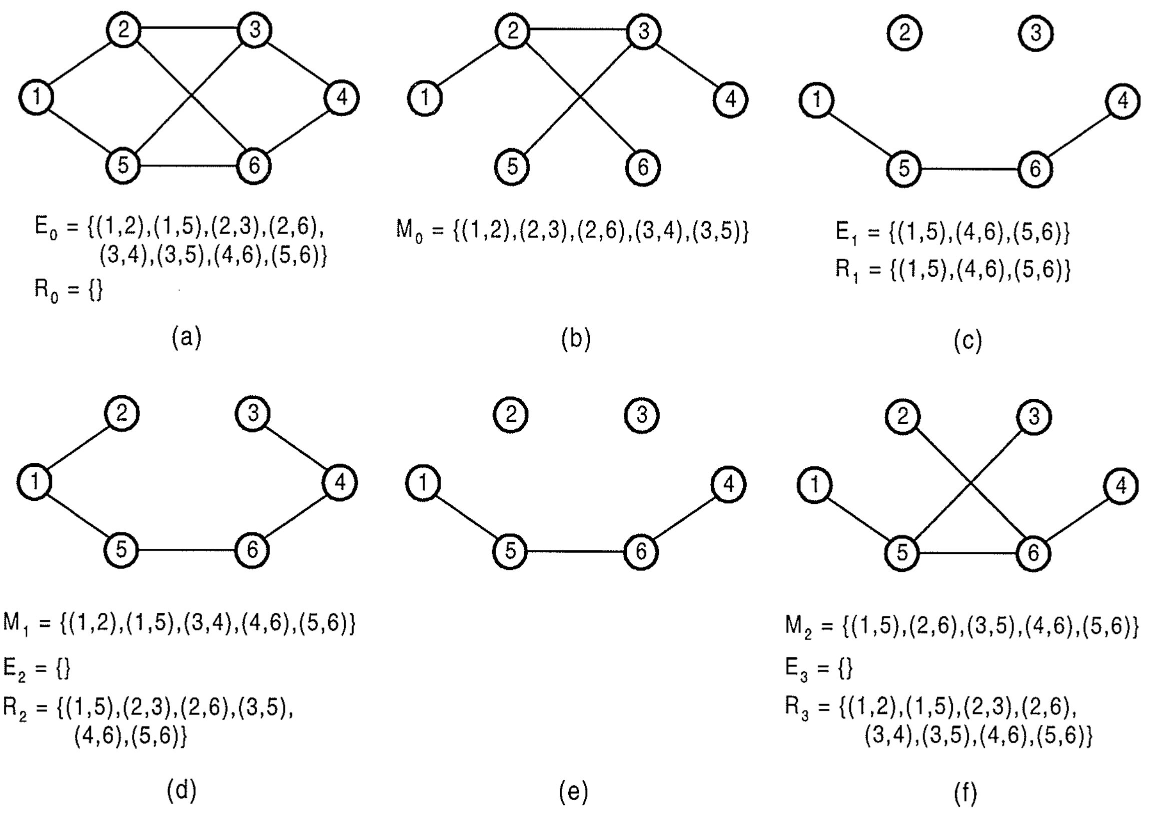

$E_0 = \{(1,2),(1,5),(2,3),(2,6),(3,4),(3,5),(4,6),(5,6)\}$
$R_0 = \{\}$
(a)
$M_0 = \{(1,2),(2,3),(2,6),(3,4),(3,5)\}$
(b)
$E_1 = \{(1,5),(4,6),(5,6)\}$
$R_1 = \{(1,5),(4,6),(5,6)\}$
(c)
$M_1 = \{(1,2),(1,5),(3,4),(4,6),(5,6)\}$
$E_2 = \{\}$
$R_2 = \{(1,5),(2,3),(2,6),(3,5),(4,6),(5,6)\}$
(d)
(e)
$M_2 = \{(1,5),(2,6),(3,5),(4,6),(5,6)\}$
$E_3 = \{\}$
$R_3 = \{(1,2),(1,5),(2,3),(2,6),(3,4),(3,5),(4,6),(5,6)\}$
(f)

Figure 4.4: RSTA Heuristic: Case 2

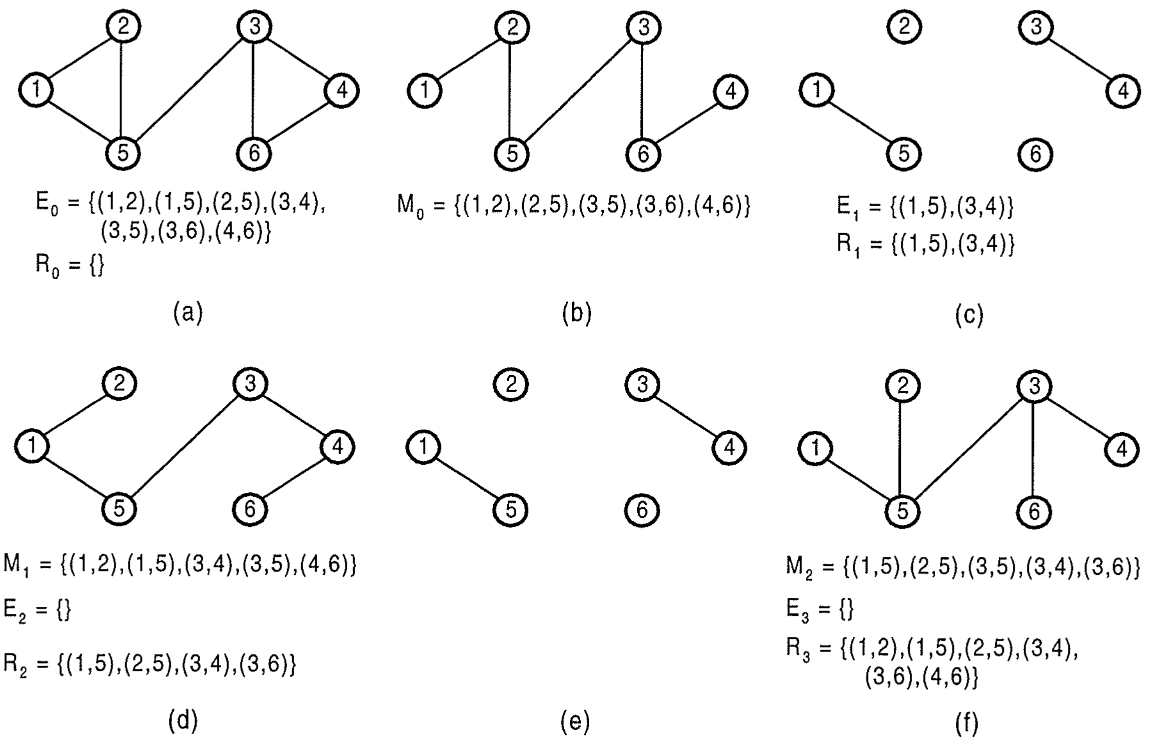
are determined in the same fashion as was done for case 1. Notice that graph G_1 in Figure 4.4(c) is no longer a connected graph but can be considered as three subgraphs. Nodes 2 and 3 would be their own subgraphs while the remaining nodes would create the third subgraph.

Step 8(*b*) of the RSTA algorithm is to find the spanning trees of these subgraphs. With a simple network such as this one, the spanning trees of the subgraphs are the subgraphs themselves. The following step in the algorithm is to choose links from previously calculated spanning trees to connect the spanning trees of the subgraphs. Doing so leads to the graph G_2 shown in Figure 4.4(d). After calculating the values for E_2 and R_2, the algorithm checks to see if all the links have been restored (RSTA step 8(f)) or if maximum restoration for the graph has been reached (RSTA step 8(g)). Since neither of these cases are satisfied, the algorithm reverts back to graph G_1, and chooses different links to connect the spanning trees of the subgraphs.

Spanning tree M_2 is created and after calculating R_3, condition (f) of RSTA step 8 is satisfied meaning that all links are restorable and the program terminates.

4.3.3 Case 3: Not Fully Restorable Network

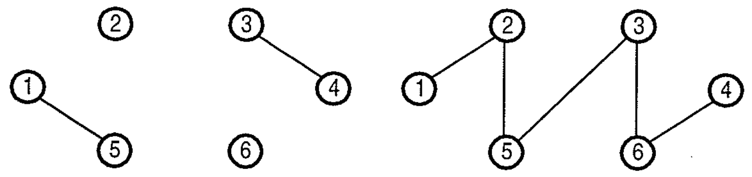
The third and final case occurs when the network is not fully restorable. That is to say that the network does not have sufficient redundancy and a single link failure will make some hosts unreachable by others.



1
2
3
4
5
6
E0 = {(1,2),(1,5),(2,5),(3,4), (3,5),(3,6),(4,6)}
R0 = {}
(a)
M0 = {(1,2),(2,5),(3,5),(3,6),(4,6)}
(b)
E1 = {(1,5),(3,4)}
R1 = {(1,5),(3,4)}
(c)
M1 = {(1,2),(1,5),(3,4),(3,5),(4,6)}
E2 = {}
R2 = {(1,5),(2,5),(3,4),(3,6)}
(d)
(e)
M2 = {(1,5),(2,5),(3,5),(3,4),(3,6)}
E3 = {}
R3 = {(1,2),(1,5),(2,5),(3,4), (3,6),(4,6)}
(f)

Figure 4.5: RSTA Heuristic: Case 3

Without going into great detail, the RSTA algorithm executes as in case 2 up as shown in Figure 4.5. At this point, three spanning trees have already been created, M_0, M_1, and M_2, and the algorithm has already reverted back to G_1. The next step is to find another spanning tree, M_3, by using different links from the previous spanning trees to connect the subgraphs. This is done and shown in Figure 4.6(a). Calculating the values for R_3, condition 8(g) of the RSTA algorithm is satisfied. With no change in the number of links that have been restored from the previous iteration of the algorithm, the algorithm terminates having reached maximum restoration.



$M_3 = \{(1,2),(2,5),(3,5),(3,6),(4,6)\}$
$E_3 = \{\}$
$R_3 = \{(1,2),(1,5),(2,5),(3,4),$
$(3,6),(4,6)\}$

(a) (b)

Figure 4.6: RSTA Heuristic: Case 3

4.4 Additional Restoration Experiments

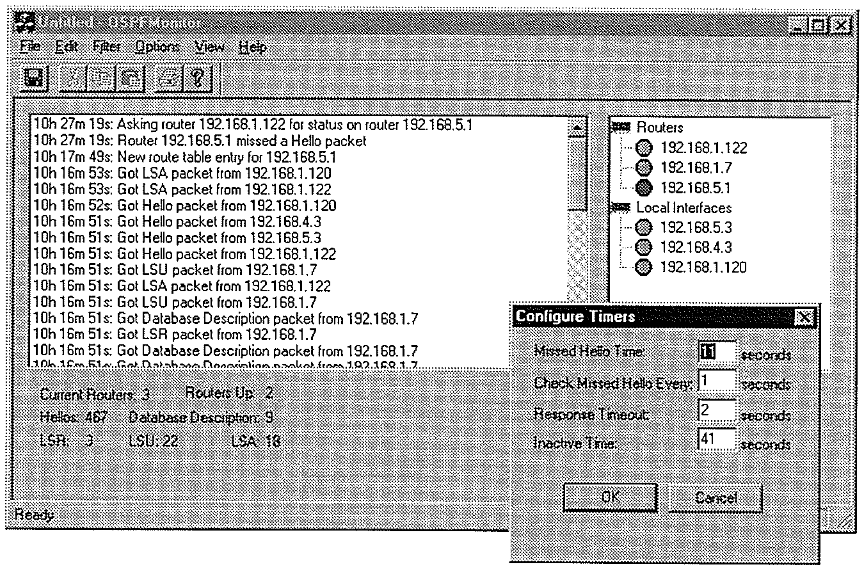
This section will look at some software and some experimental tests that were done with the OSPF protocol. The experimental test network consisted of Windows NT Servers running Routing and Remote Access Service [29] (RRAS), Cisco 3640 Router with IOS 11.3, and Linux routers. The OSPF protocol running on the NT based routers was developed by Bay Networks and bundled with RRAS. The Linux routers used an implementation developed by The Merit GateD Consortium [30]. These router were connected in various topologies for testing.

4.4.1 OSPF Network Monitor

The OSPF network monitor [31] currently resides on a Windows NT Server and monitors all OSPF packets it receives. It logs all information to the main window as well as keeping track of statistics on the number and from where they are received. The OSPF network monitor specifically targets the Hello packets, keeping track of inter-arrival times. If for any reason a Hello packet is missed, a warning is issued and the monitor contacts other monitors within the network to verify the missed Hello packet. This type of verification can be used to validate a network fault so that appropriate action can be taken to rectify it. Initially, the OSPF network monitor would reconfigure the protocol online by using a command line utility called routemon. However, due to a bug in the routemon software, on the fly configuration of the protocol has not been implemented in the network monitor. Online configurations have been done using a Cisco 3640 router as described in the next section. Figure 4.7 shows the OSPF network monitor in operation as it gathers and logs OSPF information.

4.4.2 OSPF Cost Metric

An initial concept for improving OSPF restoration time was to dynamically configure the cost parameter to force OSPF to recalculate its routing tables by using a Super-



```
10h 27m 19s: Asking router 192.168.1.122 for status on router 192.168.5.1
10h 27m 19s: Router 192.168.5.1 missed a Hello packet
10h 17m 49s: New route table entry for 192.168.5.1
10h 16m 53s: Got LSA packet from 192.168.1.120
10h 16m 53s: Got LSA packet from 192.168.1.122
10h 16m 52s: Got Hello packet from 192.168.1.120
10h 16m 51s: Got Hello packet from 192.168.4.3
10h 16m 51s: Got Hello packet from 192.168.5.3
10h 16m 51s: Got Hello packet from 192.168.1.122
10h 16m 51s: Got LSU packet from 192.168.1.7
10h 16m 51s: Got LSA packet from 192.168.1.122
10h 16m 51s: Got LSU packet from 192.168.1.7
10h 16m 51s: Got Database Description packet from 192.168.1.7
10h 16m 51s: Got LSR packet from 192.168.1.7
10h 16m 51s: Got Database Description packet from 192.168.1.7
```

```
Current Routers: 3     Routers Up: 2
Hellos: 467    Database Description: 9
LSR: 3         LSU: 22        LSA: 18
```

```
Routers
  192.168.1.122
  192.168.1.7
  192.168.5.1
Local Interfaces
  192.168.5.3
  192.168.4.3
  192.168.1.120
```

Configure Timers

Missed Hello Time:	11	seconds
Check Missed Hello Every:	1	seconds
Response Timeout:	2	seconds
Inactive Time:	41	seconds

Figure 4.7: OSPF Network Monitor

visory system. In the event of a network failure on a particular interface, the cost parameter for that interface would be reconfigured to be an arbitrarily high number such that it wouldn't be chosen in the shortest path first (spf) algorithm. In this experiment, a missed Hello packet was considered to be a network failure and reconfiguration would occur immediately after. With an spf-delay of five seconds and an off the wall time reading, the results shown in Figure 4.8 were as expected.

The figure depicts two graphs, one for the original OSPF restoration and one for the Supervisory system. It's important to note that the y-axis serves to represent that a connection exists. Any connection value greater than zero indicates that a connection exists. The fact that the supervisory graph appears lower than the OSPF graph has no meaning in this case.

The figure shows connection loss at about the 10 second mark. The OSPF Routing

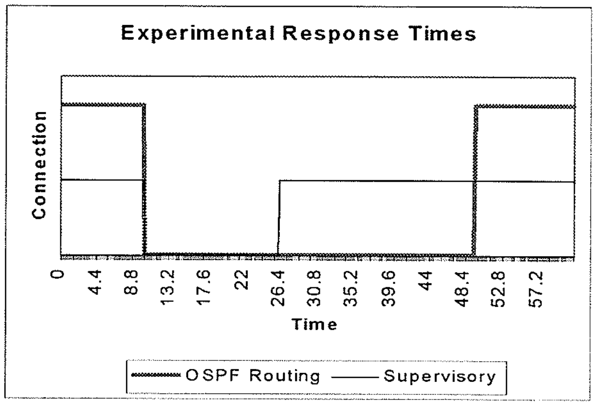


Figure 4.8: Dynamic Cost Reconfiguration

graph re-established connection 40 seconds later, which is after the RouterDeadInterval time has elapsed. As expected, the Supervisory system re-establishes communication 15-16 seconds after the fault occurred. 10 seconds accounting for the HelloInterval time, and 5 seconds accounting for the spf-delay time.

Although this result was expected, the actual experiment had many benefits. An obvious benefit is the configuration and operation of the OSPF protocol across multiple platforms. Experience gained from this can not be acquired from reading about the protocol. Another benefit comes in terms of the spf-timers and unnecessary routing spf calculations. This experiment brought to our attention the role that the spf-timers play in the overall restoration mechanism as well as the effects of short lived network failures causing unnecessary spf calculations. These issues were addressed in the previous chapter.

4.5 Summary

This chapter covered some of the computational results that were obtained by solving the integer programming problem using the opbdp package. Although the test network was small, it is more than sufficient to prove and test the formulation of the integer problem. Although the optimal solution is always desired, there are situations where a near optimal solution is considered to be better considering the trade offs involved. This near optimal solution is provided in terms of an algorithm that takes advantage of the structure of the problem.

For these solutions to be useful, there needs to be an implementation of them in an actual networking environment. The following chapter discusses issues related with the implementation of the two-phase restoration system and also includes some background concerning the technologies involved.

Chapter 5

Implementation Issues

5.1 Introduction

Issues pertaining to the implementation of the two-phase restoration system are discussed in this chapter. Multi-Protocol Label Switching [34] (MPLS) and Active Networking are proposed as potential technologies that can be used to implement the restoration system. Due to the young age of these technologies, there are a lot of outstanding issues concerning their own implementations. This chapter will not discuss implementation issues of MPLS or active networking, but will refer readers to other sources. The focus will be placed on how the two-phase restoration system, MPLS, active networking, and OSPF will function as a whole to provide fast restoration. To get readers acquainted with these technologies, a brief introduction to each technology is presented.

5.2 Multiprotocol Label Switching

5.2.1 Introduction

In MPLS a short fixed-length label is used to represent an IP packet's header. In traditional networks, packets must be processed to determine the destination address at every router or node along the path in the network. This is known as hop-by-hop routing. MPLS on the other hand will encapsulate the IP packets with its label when the packet first arrives into an MPLS routing domain. The MPLS edge router will look at the information in the IP header and select an appropriate label for it. An advantage to this is that the label selected by the MPLS edge router can be based on more than just the destination address in the IP header. This is in contrast to conventional IP routing, where the destination address will determine the next hop for the packet. All subsequent routers in the MPLS domain will then forward the packet based on the label not the IP header information. Then, as the labeled packets exit the MPLS domain, the edge router through which it is leaving will remove the label.

The routers or nodes in MPLS are called Label Switched Routers (LSRs). There are two categories of LSRs: MPLS edge routers and the core LSRs. MPLS edge routers are routers that connect an MPLS domain to a router that is outside the domain. A router that is outside the domain can mean either the router does not support MPLS, and/or the router is in a different domain. Edge routers are classified as either ingress or egress where an MPLS ingress router handles traffic as it enters an MPLS domain and MPLS egress routers handle traffic as it leaves an MPLS domain.

5.2.2 Label Switching

There are three important distinctions between label switching and conventional routing as shown in Table 5.1.

Table 5.1: Distinctions between conventional routing and label switching

	Conventional Routing	Label Switching
Full IP Header Analysis	Every node	Once at network edge
Uni/Multicast Support	Multiple complex forwarding algorithms	One forwarding algorithm required
Routing decisions	Based on address only	Based on any number of parameters, such as QoS, VPN membership

*Taken from "Multiprotocol Label Switching (MPLS)" Technology Guide [33]

One important point to make note of is that label-based forwarding is meant to complement conventional routing, not to replace it. A label can simply be thought of as a short hand notation for a packet header. This label is generated at the ingress node and is used by the core LSRs in their forwarding decisions. The value of the label usually changes at each LSR in the path as the incoming label is looked up and swapped for another. From this we can see that labels only have meaning between neighbouring nodes. With this, MPLS can be used to forward data between neighbouring nodes even if no other nodes on the network support MPLS. We can also see that when MPLS is used between more than two nodes, operations between neighbouring nodes are independent from any other pair of nodes.

Forwarding of packets proceeds as follows: a core LSR will receive a labeled packet, extract the label, and then use it as an index into the forwarding table. This will provide the LSR with a new outgoing label that will be added to the packet. The packet is then sent out the LSRs outgoing interfaces that are specified by the label. The forwarding table can be implemented at either the node level or at the interface level.

Once a labeled packet leaves the core of an MPLS domain, the egress router would simply remove the label to forward packets on an unlabeled interface.

A control component is responsible for creating label bindings, binding between a label and routes. These bindings then have to be efficiently distributed to the other LSRs. When a label is assigned, it involves the allocation and binding of the label. There are three methods of label assignment as defined by the MPLS WG, topology based control traffic driven, request based control traffic driven or data traffic driven. Topology driven schemes assign labels when routing protocol control traffic is received. Request driven schemes respond to request based control traffic. Data traffic assignment will assign labels at an LSR when data traffic is received.

5.2.3 Label Distribution

The labels that are used in MPLS need to be distributed among the MPLS domain routers so that proper routing may take place. LSRs in a domain acquire the labels through several label distribution methods.

Explicit Label Distribution

The MPLS WG is developing a protocol called the Label Distribution Protocol (LDP) which will incorporate some of the already existing proprietary solution such as TDP

(Tag Distribution Protocol: Cisco), ARIS (Aggregate Route-based IP Switching: IBM), and FANP (Flow Attribute Notification Protocol: Toshiba) [35]. Label distribution must ensure that LSRs receiving labeled packets know what to do with the packet. A question arises as to which LSR allocates the label. There are a few methods for label allocation. Downstream label allocation is where labels are assigned or allocated by the downstream LSR. The downstream LSR is the LSR that is receiving the data and using the label to index a switching table. It is most natural for the downstream LSR to allocate the label because it is the node that uses the label to make forwarding decisions. After all labels have been allocated, the downstream LSR will distribute the labels and bindings to all its neighbours. The other alternative to downstream label allocation is upstream label allocation is where the labels are allocated by the upstream LSR. In this case, the LSR allocating the label is different from the LSR that is to use the label as an index. One point to mention is that the label referred to in this section is not the one used as an index into the switching table but rather the resulting label of table lookup. Both of these methods allocate labels at every LSR in the MPLS domain. An alternative to this is to have a unique label within an MPLS domain that would only have to be allocated once.

Piggybacking on Other Control Messages

The previous section discussed the MPLS LDP protocol, but there are several protocols (OSPF, BGP, and RSVP) already in existence that can be modified to distribute labels in MPLS. What is interesting about this is that these protocols already have the underlying mechanisms that are needed for label distribution because they are the same mechanisms used for distributing routing or control information. Piggybacking is to use these mechanisms to distribute the labels without having to create a new distribution protocol. The problem arises as to how the labels are to be allocated and what will result if an invalid label has been received. For the issue of acceptable label values, refer to the [34].

Reliability

A concern that rises when discussing label distribution is the reliability. If the labels can not be distributed reliably to all LSRs, then problems will be encountered, such as invalid labels or routing loops. The reliability of label distribution will depend on the forwarding components of L2 and L3 as well as the routing protocol operation. For the case where label distribution is using the piggybacking method, it can be said that the reliability of the label distribution is the same as the reliability of the routing protocol it is piggybacking on. In the case where LDP is used, reliability could possibly be accomplished by using a reliable transport protocol such as TCP or by building features for reliable transfer within LDP itself.

5.2.4 Hierarchical operation

Networks are naturally divided into different levels or hierarchies. For example, in OSPF, there exists a two level hierarchical structure. The first level is the routing between routers within an area and the second level would be the routing between areas. With the use of label stack, MPLS allows for hierarchical operation.

For example [34], consider the scenario in the figure below. There are three routing domains (Domain #1, #2, and #3). The domain boundary routers are shown for these three domains (router R1 and R2 for domain #1, R3 and R8 for domain #2, and R9 and R10 for domain #3). Lets assume that these boundary routers are running BGP. The internal routers are also shown for domain #2 depicting a path through R4, R5, R6, and R7 between the boundary routers R3 and R8.

We can see that there are two distinct levels of routing taking place. For example, routers R3, R4, R5, R6, R7, and R8 are running OSPF within domain #2 to compute internal routes. The domain boundary routers R1, R2, R3, R8, R9, and R10 operate BGP to compute routes between routing domains. MPLS hierarchical routing can be

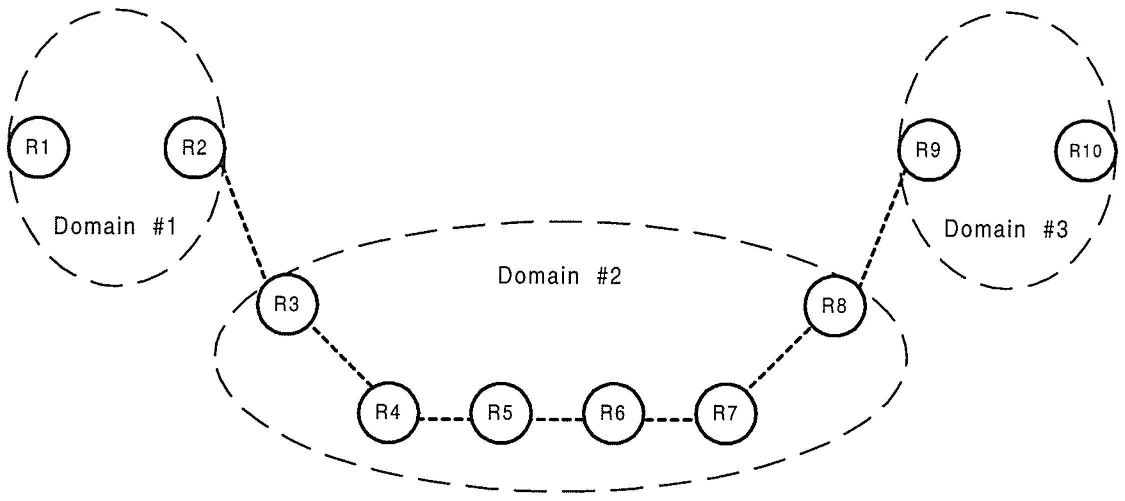


Figure 5.1: Hierarchical Routing with MPLS

used here at the BGP level between the domain boundary routers and at the OSPF level between the routers within domain #2. So as a packet traverses domain #2, it will contain two labels, encoded in a label stack. Routers R3 and R8 will use the higher level label that would be encapsulated in a header containing the lower level label used within domain #2.

5.2.5 Explicit Routing

There are two ways in which route selection can take place: hop by hop routing and explicit routing. Explicit routing is where the LSP (Label Switch Path) is determined by a single node rather than by each node along the path as in hop by hop routing. This single node is usually the ingress or egress LSR. Explicit routes in MPLS have to be specified when labels are assigned to the packets.

Establishing a point to point explicit route requires that the LDP (Label Distribution Protocol) packets used to set up the LSP contain the explicit route. From this we can conclude that the LSP is established in order either form the ingress LSR to the egress LSR or vice-versa. The explicit route can be chosen in two ways: 1) by configuration and 2) by using a routing protocol. The first option, by configuration, means that the explicit route can be configured by the administrator or by a centralized server. The second way to express an explicit route uses the routing protocol. The routing protocol would allow for the ingress and egress nodes to know the entire LSP. This implies that a link state routing protocol (e.g. OSPF) is to be used in which the topology of the network is node to all nodes of the network. The other option is a path vector routing protocol such as BGP.

5.3 Active Networking

An active network is an architecture in which customized applications are executed in the network. There are generally three main components that make up the active network, a set of active nodes, active applications, and active packets. The set of active nodes does not comprise all of the nodes in the network, there could be non-active nodes as well. These components work together in the network by transporting data and/or code encapsulated in active packets between nodes in the network. The active nodes in the network would then use the data to be processed in an active application or execute the code that was received.

With this architecture, active networking has several potential benefits. The first one is in the area of new services. By separating the services from the underlying infrastructure, it is possible for new services to be deployed incrementally. The second benefit is the customization of services for different applications. Users can implement a custom service for an application tailored to their needs. Although communication, not computation, is the main function of the active network, it is possible to use the computational power within the network to our advantage [36] [37].

For these benefits to be realized, there needs to be a way for users to direct their packets to a particular application on the active node. This is provided by the Active Network Encapsulation Protocol(ANEP). ANEP is a mechanism that can be used to transport active network data over different media. The proposed format allows the use of existing network infrastructure over the link layer.

5.3.1 Active nodes

An active node can be decomposed into two components that specify where active applications execute and communication with the operating system. The first component is the NodeOS. The NodeOS provides resource management for the active node and functionality to the execution environment, which is the second component. An active node can potentially contain multiple execution environments which are all managed by the NodeOS. These execution environments provide an interface or virtual machine that can be managed and controlled by directing active packets to them.

5.3.2 Active Applications

An active application is a program that can implement customized services for end-user applications. Active applications are executed by the virtual machine of a particular execution environment. The applications can either be stored locally on the active node or demand downloaded into the node when an active packet arrives requesting a particular application. Active applications can be used for many different tasks. Some examples are for congestion control and providing reliable multicast.

5.3.3 Active Packets

Active packets are network packets that carry an active network frame. Active packets can be used for a variety of tasks. The first and most obvious is to provide service for the end-user by executing specific active applications within the network. Another important responsibility of active packets is the ability to carry active applications in their payload. When an active node receives an active packet requesting service to a particular application, the active node can send out a request for the application to the previous active node. The previous node would then encapsulate the application in an active packet a send it across the network. In this way, active applications can be dynamically loaded into the active node by the exchange of active packets.

5.3.4 Active Network Encapsulation Protocol

The active network encapsulation protocol is used to encapsulate active data for transmission between active nodes in the network. The main reason for ANEP is to provide standardization for active packets. Active nodes must be able to quickly and uniquely identify the execution environment active packets are directed to. With a standard ANEP header, this is easily achieved. An ANEP header can also provide options for default processing when a desired active application is unavailable. An option could be to just forward the active packet to the next node or dynamically download the application.

The format of the ANEP header is shown in Figure 5.2. Details of each field are

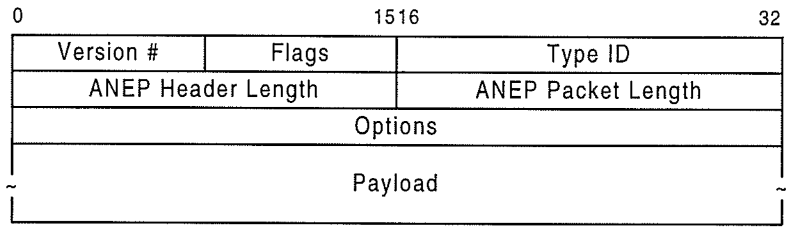


Figure 5.2: ANEP Header

described in the active network encapsulation protocol RFC draft [43].

5.3.5 Active Networking Implementations

This section quickly discusses three of the current implementations of active networking. There are undoubtedly more implementations which will help to either prove or disprove active networking as a technology for future networks. For further information pertaining to each of these implementations, see the related citations.

ANTS [37] [38]

Components

- *capsule:* A capsule replaces the traditional network packet by having information that refers to the processing that needs to be done on it's behalf.
- *active node:* Routers and end nodes that have the capability to execute user defined programs within a restricted environment limiting access to shared resources.
- *code distribution:* A mechanism by which active applications can be dynamically loaded from other active nodes. This mechanism also provides for caching of active applications to increase performance.

Main Goals

- To support dynamic deployment of new network protocols
- Simultaneous support for a variety of network protocols
- Creation of new protocols by mutual agreement - no centralized registration scheme needed

SwitchWare [39] [40]

Components

- *active packets:* A network packet with a mobile program that includes both data and code.

- *active extensions*: Node resident extensions that can be dynamically loaded from node to node by using active packets for transport. Common extensions would be available on all active routers for performance benefits.

- *active router:* A secure infrastructure in which the other two components can reside on top of.

Main Goals

- Balance the programmable network's flexibility with respect to safety and security issues.

- To maintain if not improve the network's usability

Janos [41] [42]

Components

- *Custom JVM:* A custom JVM that is built upon the OSKit++, having the ability to run different execution environments. Since it is a JVM, it is as portable as any JVM in existence today.

- *OSKit++:* Provides a framework and a set of modularized library code for the construction of OS kernels, servers, and other core OS functionality

Main Goals

- Resource management, security and performance
- Technology transfer

5.4 Implementation Issues with OSPF

This section will detail some of the interoperational issues of MPLS or active networks with OSPF. One of the main benefits of the two-phase restoration system is that it does not affect the operation of OSPF. There are four main issues when implementing the two phase restoration system. The first is distributing the restoration table information. This is followed by propagating network failure information to the necessary nodes in the network. Thirdly is the actual operation of either MPLS or active networking to provide for fast restoration (primary scheme). After the primary scheme of the two-phase restoration system is complete, control of the network must be passed back to OSPF for optimal restoration(secondary scheme).

Before tackling each one of those issues, there needs to be an understanding of the scope or restoration provided. An OSPF network is divided up into different subnetworks called areas. It is in these areas that the two-phase restoration system will reside. For each area in an OSPF network, a separate two-phase restoration system would be implemented thereby reducing the size of the preconfigured restoration tables to the size of an area. This leads to the conclusion that the MPLS and active network domain are the same as an OSPF area.

5.4.1 Issue 1: Distribution

After calculating the set of restoration tables, it is necessary that all nodes within the area are aware and have the same set of tables. A proposed method for distributing this information throughout the network is described here.

MPLS Label Distribution

With the already reliable flooding mechanism in place by OSPF, distribution of labels would be most efficient using piggybacking. After the preconfigured restoration tables are created and the corresponding labels have been assigned, the labels can be piggybacked on top of OSPF's Hello packets or any other Link State Advertisement(LSA). According to RFC 2328 [45], five out of the eight option bits in the LSA options field have meaning. One of these options could be used to indicate that the LSA is carrying MPLS label information. An alternative to this is since OSPF runs over top of IP (Internet Protocol), an option bit can be used in the IP header to indicate MPLS label distribution. Label distribution is only relevant where the preconfigured restoration tables are calculated by only one node in the area. Distributing MPLS labels is not even an issue if the restoration tables are calculated by each node in the area. As a link state routing protocol, each of the nodes in an area have knowledge of the topology of that area. With that knowledge, the restoration tables can be created by each node individually. This concept is very much the same as used by

OSPF when performing spf calculations.

Active Networks Preconfigured Routing Table Distribution

As described previously, calculation of the preconfigured restoration tables can be done by one node and flooded to all the other nodes or it can be calculated by all nodes. With all nodes executing the same calculations for obtaining the restoration tables, there is no need for distribution. However, the alternative is to have a single node perform the calculation and then distribute the restoration tables to the other nodes. The same method of piggybacking can be applied here as was done in MPLS label distribution. An alternative to piggybacking is to encapsulate these routing tables into active packets called Restoration Table Advertisement(RTA) active packets. RTA packets can be assigned their own Type ID in the ANEP header(Figure 5.2). The Type ID would be associated with an active application that would handle the processing of the restoration tables in the RTA packet. The RTA Active Application (RTA3) would handle the responsibility of storing the RTA payload in a persistent state for later use. Since RTA3 needs access to resources, possibly the filesystem, certain security measures must be in place [44]. Security measures such as access policies to resources and authentication of RTA packets must be addressed. These security measures would most likely reside in the execution environment that the RTA3 would execute in.

5.4.2 Issue 2: Fault Notification

In order for the primary scheme of the two-phase restoration system to be of any use, the nodes in the network must know about an occurrence of a fault.

MPLS

Fault notification throughout the area is necessary for MPLS. With ingress and egress nodes responsible for assigning and removing MPLS labels respectively, flooding is not necessary, but is acceptable. A more efficient alternative to flooding is to have the ingress and egress nodes participate in a Multicast group. Multicasting is a way of distributing data to multiple recipients. Where flooding is broadcasting, multicasting is a broadcast to only recipients that are associated with a certain multicast group. More information can be found on multicasting in [46] [47]. Multicasting in OSPF is defined as an extension in RFC 1584 [48]. Multicast OSPF (MOSPF) can be used as the transport for fault propagation in the MPLS domain.

Active Network

Active network nodes respond when active packets are received from other nodes. This is very similar to MPLS, where MPLS enabled nodes would respond to the attached MPLS labels. A difference between the two is that ingress and egress nodes in MPLS assign and remove labels, whereas in active networks, potentially any active node can encapsulate and send active packets. The question is which node initiates the encapsulation. Drawing from MPLS, ingress nodes can perform this initial encapsulation. Once the initial encapsulation is performed, active nodes within the area would do the routing of the active packet. This suggests that only ingress active nodes need knowledge of a fault since interior nodes respond to the initial active packet. However, egress nodes also need to know about the fault so that the outgoing packet (destined for another area) is no longer encapsulated. The same procedures for multicasting the fault to the ingress and egress nodes with MOSPF can be used here.

5.4.3 Issue 3: Primary Scheme

After receiving a fault notification, ingress nodes would either attach MPLS labels or encapsulate packets for active network use. Actual operation of MPLS or active networking was briefly described previously and is left to the reader for further investigation. At this point, data would be rerouted using the preconfigured restoration tables through the network. The primary scheme is now dependent on MPLS and active networking technology to provide reliable data transfer through the area.

During the operation of the primary scheme, OSPF is running in the background. It is still sending Hello packets, and other LSAs to its neighbours. The only difference is that all of these packets are now being routed by the primary scheme technologies and not OSPF itself. To the knowledge of OSPF, the network is has not changed because it does not know about the network failure. When OSPF realizes that a failure exists (the RouterDeadInterval time has elapsed), it will begin procedures for optimal restoration as described in the following section. However, consider the case of a short lived network failure.

A short lived network failure could exist due to a configuration error in the node. Realizing that an error in the configuration exists, the administrator corrects the error within the RouterDeadInterval time. This would cause the primary scheme to be activated (configuration error) and then hopefully deactivated when the configuration error is corrected so that OSPF can resume control. Similar to propagating a fault notification, when a short lived fault is corrected, a resume notification is issued to the ingress and egress nodes by the same node that detected the fault. A fault is considered corrected when the node that issued the fault notification hears a Hello packet across the failed link. The resume notification would cause the primary scheme to be deactivated. This would disable MPLS(do not attach labels) or active networking(do not encapsulate) so that routing of data is resumed by OSPF. These routing changes for a short lived failure are depicted in Figure 5.3.

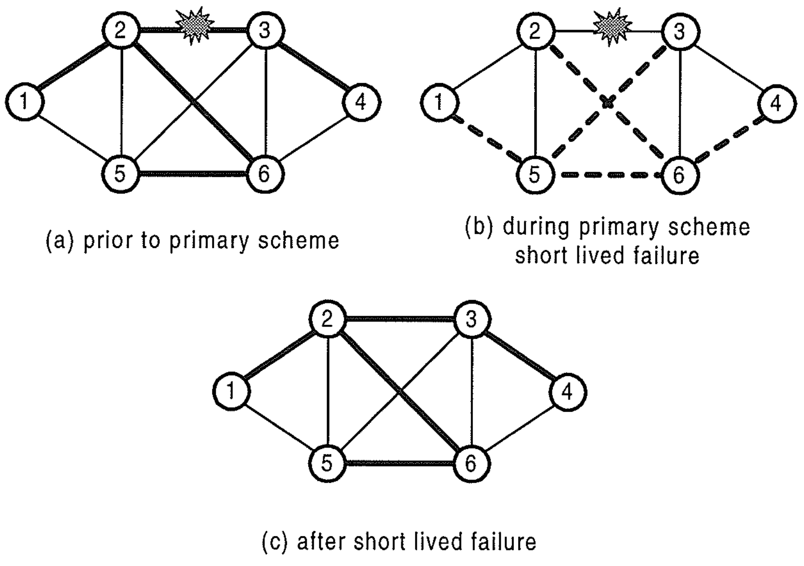

1
2
3
4
5
6
(a) prior to primary scheme
1
2
3
4
5
6
(b) during primary scheme
short lived failure
1
2
3
4
5
6
(c) after short lived failure

Figure 5.3: Routing Changes for Short Failures

5.4.4 Issue 4: Secondary Scheme

For a fault which is recognized by OSPF, OSPF would perform an spf calculation to reconfigure its routing tables optimally. At this point, the primary scheme should hand over control to the secondary scheme. The handover is merely deactivating the the primary scheme. Since OSPF recognizes a fault after the RouterDeadInterval time, the node which issued the fault notification would also listen on the failed interface for the same amount of time. If no Hello packet is seen, then the node would issue a resume notification to the ingress and egress nodes to terminate the primary scheme. The routing changes for a persistent failure is shown in Figure 5.4.

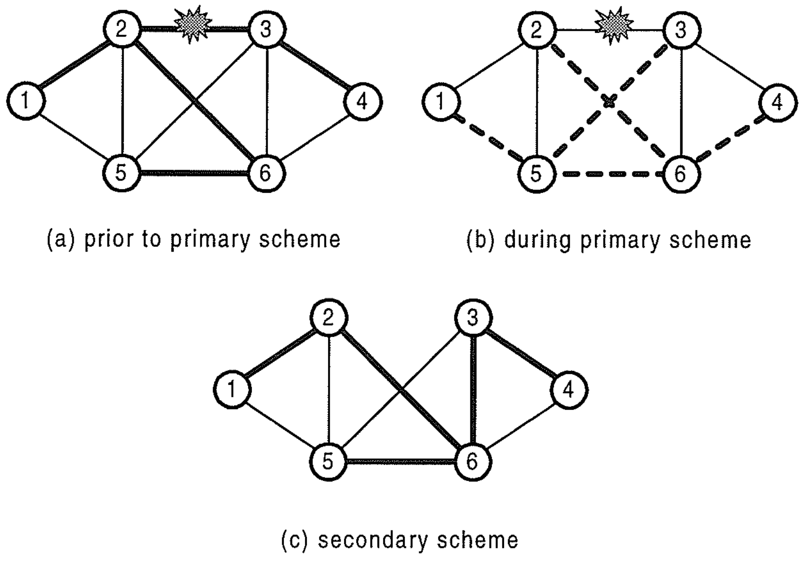


Figure 5.4: Routing Changes Persistent Failures

An issue arises here with respect to convergence. What happens if the primary scheme is deactivated and data is sent before OSPF converges to the optimal state? A simple solution to this problem is to wait until OSPF has converged before multicasting the resume notification.

5.4.5 Other Issues

This section will briefly describe some other issues concerning the operation and implementation of the two-phase restoration system.

1. *Area Border Routers(ABRs)*: An ABR is a router which is at the boundary of an OSPF area. In this case, ABRs are required to have two sets of preconfigured restoration tables, one for each area it is connected to. If there is ever an occasion where a fault occurs in the interior of each area connected to the ABR, more computation is necessary for proper data transmission.

   With the ABR having knowledge of two failures, (one in each connected area), it must first remove the attached label and swap it with a label from the other restoration table. This is of course for when MPLS is being used. When active networking is used, the ABR would have to execute the active application twice. The first time would be to decapsulate the active packet after receiving it from the first area. The second active application would be executed to encapsulate the packet for the next area. This dual processing of the packet can be avoided by putting more complexity into the active application of the ABRs. The ABR active application would be able to receive the packet from one area and directly encapsulate it for transmission into the next area.

2. *Fault Localization*: Another interesting issue comes in the area of fault localization. The ability to localize a fault has many advantages. The first is that the amount of traffic generated by a fault notification is reduced to only those local to a fault. The terms local to a fault refer to the minimum number of nodes

that are needed for there to be a restoration path around the fault. Another benefit to fault localization is that the optimal path is still being used where ever possible in the network.

In MPLS fault localization would require a few changes. The first change is that all MPLS enabled nodes be able to attach labels to any data that arrives for the purpose of restoration. In this way, the fault need not be propagated to the ingress and egress nodes, but just to those nodes local to the fault. An alternative to this would be to deploy explicit routing using loose source routing. Loose source routing is where only a portion of the entire path is specified. The portion that would be specified is the path around the fault. This leaves the remainder of the path to follow the shortest path provided by OSPF.

Active networking can deploy fault localization in a very similar manner. Only active nodes local to the fault would encapsulate and transfer data in the form of active packets.

The difficult part here is determining which nodes are local to the fault. One solution is to select the two nodes that are adjacent to the fault. If there is a common node distance one away from the two fault adjacent nodes, then these three nodes are local to the fault. If a common node does not exist, then all nodes with a distance of one from the two adjacent nodes are examined. From this set of nodes and all associated links between them, a spanning tree is formed. If a spanning tree exists, then this set of nodes is local. If a spanning tree is not found, then the search is expanded to all nodes with a distance of two from the adjacent nodes. Eventually, the edges of the area would be reached, and the fault is no longer local.

If it is not already clear, real time processing is necessary for fault localization.

5.5 Summary

This chapter provided some background into some of the issues relating to the implementation of the two-phase restoration system. The focus of the implementation was on MPLS and active networking technology. There are other issues that need to be addressed before implementation becomes feasible. One of the main issues would be security in the network.

MPLS and active networking are potential technologies that could support the two-phase restoration system. However, the restoration system is not limited to them. Other technologies are available in which further research is needed.

Chapter 6

Conclusion

This concluding chapter discusses the contributions of this work, it also proposes directions for future research.

6.1 Summary of Contributions

The contributions of this thesis are in the field of survivable networks or network restoration. With focus on the network or IP layer, this thesis distinguishes itself from other papers whose main target is Physical Layer restoration. This thesis has provided a view of current restoration techniques at both the network and Physical Layer. A two phase restoration system has been introduced to provide fast restoration. We have identified the issue of balancing the restoration tables across the network by developing an optimization model and the solution for the problem. A heuristic has also been proposed to obtain near optimal solutions to the problem. Finally, the necessary background research for implementation of the two-phase restoration system has been presented.

The first contribution of this thesis is the introduction of the two-phase restoration system to enhance the current OSPF restoration mechanism. The OSPF restoration mechanism worked very well to provide optimal path reconfiguration but at the expense of speed. Fast restoration is provided by preconfiguring a set of restoration tables that can be used immediately upon failure. The combination of the preconfigured restoration tables, and OSPF's optimal path reconfiguration leads to the two-phase restoration system that meets two important requirements. These requirements are fast restoration and optimal reconfiguration.

Another contribution is the optimization model that has been developed for the primary scheme of the two-phase restoration system. Not only does the primary scheme provide for fast restoration, but it provides it in an optimal manner in two ways. With a specified number of restoration tables, the optimization model will maximize restoration of the number of links that are protected. If full restoration is desired, the optimization model will determine the necessary configurations of the restoration tables to provide the minimum set of tables required. The problem is formulated as a non-linear integer programming (IP) problem with linear constraints. Testing has shown that the IP problem converges to an optimal state. To help ease the design process of the problem, a Spanning Tree Generator (STG) tool was created. STG aids in translating a network diagram into a set of equations that formatted correctly for the opbdp package.

A heuristic algorithm has been proposed for a near optimal solution to the primary scheme. The Restorable Spanning Tree Algorithm (RSTA) can provide for faster computation time and easier implementation. A case study of the RSTA heuristic was also presented to illustrate possible scenarios that may arise during computation.

Lastly, research has been done in dealing with the implementation of the two-phase restoration system. The interaction between MPLS, active networking and OSPF plays an important role in the overall system. This architecture is the first step in implementing the two-phase restoration system. This thesis has provided some of the background research necessary for implementing the two-phase restoration system.

6.2 Future Work

We conclude this thesis by identifying some of the important issues to be addressed as an extension to this research.

1. *Reducing the size of the IP problem*: With an increase in the number of nodes leading to an exponential increase in the number of decision variables, it is apparent that reducing the size of the IP problem is necessary. The most obvious reduction is formulating the problem in terms of links, not arcs as defined in Chapter 3. This would reduce the number of decision variables by half. The associated problem with using links is that the connectivity constraint would have to be reformulated in a fashion to account for the *od* pairs traversing the links in opposite directions. Using links would also remove the necessity for Constraint 3.9

2. *Speed-up*: Additional work is required to improve the computation time of the IP problem. This can possibly involve reducing the size of the problem as men-

tioned previously. Another possible approach is to obtain an initial feasible solution to the problem so that less nodes will have to be examined during the optimization procedure. The RSTA heuristic can be fully tested and investigated further to provide faster computation time.

3. *Network topology*: As was stated from the results in Chapter 4, network topology greatly affects the number of spanning trees that are required. Designing networks with enough redundancy to require only two spanning trees for restoration is desired. In discussing network topology, the IP problem assumed a symmetric network, but this is not always the case. Further research into asymmetric networks would be needed to develop a more general optimization model, where a symmetric network would be a special case of this general model.

4. *Dual link failures*: Although single link failures are most common, there are situations where dual link failures may exist. Although the IP problem was not formulated to protect dual links failures, it inherently does. The optimization model can be altered slightly to provide for even better dual link failure protection. Altering the formulation for this provision will increase the number of decision variables by the number of dual link permutations that are possible in the network. In the same manner that the formulation constraints checked for single link protection, they can check for dual link protection. The original model examined whether or not a link existed in a spanning tree. The new model would then examine whether or not all possible combinations of dual links existed in a spanning tree. It is apparent that the new model would also

account for all single links failures as well.

5. *Fault detection*: Fault detection always remains an important issue in restoration. With the two-phase restoration system, the validity of a fault is not as big a concern as it is in OSPF. This is because the two-phase restoration system does not affect the original routing table in the primary scheme, thereby eliminating any unnecessary spf calculations. Even so, it is still important to be able to detect faults as quickly as possible so that restoration systems in place can take action. This thesis looked only at single link faults, but node faults would be a very important event to consider in the future.

6. *Adaptive algorithm*: Currently, after a network failure, the set of preconfigured restoration tables would have to be recalculated for the new network topology. This can be greatly reduced by looking into algorithms that can update spanning trees when individual links are subject to change [49]. Research is needed in updating a set of spanning trees while still preserving the maximum restoration possible.

Appendix A

Catalog of Alarms

This appendix catalogues the variables, signals and alarms that are potentially important to the topic of network restoration.

Open Shortest Path First(OSPF)

(A list of important variables and parameters in configuring OSPF for Network Layer restoration)

- Cost
    - The cost of sending a packet on an interface.
    - Default cost: none
- Metric
    - The metric used for generating the default route.
    - Default metric: 10

- Hello Protocol
  - The Hello Protocol is used for establishing and maintaining neighbour relationships. It also makes sure that there is bidirectional communication between the neighbours.
- HelloInterval
  - The interval, specified in seconds, of time that hello packets that a router sends on the interface.
  - Default time: 10 seconds
- RouterDeadInterval
  - The interval, specified in seconds, of time that hello packets must not have been seen before neighbouring routers will declare the router down.
  - Default time: four times the HelloInterval value
- Inactivity Timer
  - A single shot timer such that when fired indicates that a hello packet has not been seen from this neighbour recently. The timer value is the RouterDeadInterval time.
- Retransmit Timer
  - The time in seconds between link state advertisement retransmissions.
- Transmit Delay

– The estimated time it takes to transmit a link state packet on an interface.

– Default time: 1 second

- spf Timers

  – The delay time between when OSPF receives a change in topology and when it starts to calculate the shortest path. You can also configure the hold time between two consecutive spf calculations.

  – Default time: spf-delay - 5 seconds; spf-hold-time - 10 seconds

- Interface States

  – A collection of states that a router can be in.

- LSAge Field

  – The amount of time in seconds that have elapsed since the LSA was first originated.

- LLDown

  – A signal to indicate that the Logical Link has lost connection.

Synchronous Optical Network(SONET)

(SONET signals and alarms for restoration)

- Section Signals

  – LOS: Loss of Signal

  - LOF: Loss of Frame
  - S-BIP: Section-level Bit-interleaved Parity error
- Line-level Signals
  - L-AIS: Line-level Alarm Indication Signal
  - L-RDI: Line-level Remote Detection Indication
  - L-BIP: Line-level Bit-interleaved Parity error
  - L-FEBE: Line-level Far End Block Error
- Path-level Signals
  - P-AIS: Path-level Alarm Indication Signal
  - P-RDI: Path-level Remote Detection Indication
  - P-BIP: Path-level Bit-interleaved Parity error
  - P-FEBE: Path-level Far End Block Error
  - LCD: Loss of Cell Delineation
  - Other Path Errors
    * TIM: Trace Identifier Mismatch
    * SLM: Signal Label Mismatch
    * UNEQ: Unequipped Signal
    * All of these signals will cause a P-RDI to be returned

Bibliography

[1] G. Malkin, *RFC 2453: RIP Version 2,* November, 1998.

[2] J. Moy, *RFC 2328: OSPF Version 2,* April, 1998.

[3] W. D. Grover, *The Selfhealing Network: A Fast Distributed Restoration Technique for Networks using Digital Crossconnect Machines,* Proc. IEEE Globecom, 1987.

[4] W. D. Grover, *Analysis of Unavailability and Resource Consumption for High Availability Paths in SONET Ring-Based Networks,* TRLabs, Technical Report TR-96-01, February, 1996.

[5] K. Murakami and H.S. Kim, *Comparative Study on Restoration Schemes of Survivable ATM Networks,* Proc. IEEE Infocom, April, 1997.

[6] M. Schwartz and T. Stern, *Routing Techniques used in Computer Communication Networks,* IEEE Transactions on Communications, vol. COM-28, No. 4, April, 1980.

[7] A. Balakrishnan, T. L. Magnanti, and P. Mirchandani, *Designing Hierarchical Survivable Networks,* Operations Research, vol. 46, no. 1, January-February, 1998.

[8] S. Ben Yahia and C. Robach, *Self-Healing Mechanisms in ATM Networks: The Role of Virtual Path Management Functions,* Proc. IEEE Int. Conference on Communications, 1997.

[9] A. Gersht and A. Shulum, *Architecture for Restorable Call Allocation and Fast VP Restoration in Mesh ATM Networks,* Proc. IEEE Int. Conference on Communications, 1997.

[10] B. Van Caenegem, N. Wauters, and P. Demeester, *Spare Capacity Assignment for Different Restoration Strategies in Mesh Survivable Networks,* Proc. IEEE Int. Conference on Communications, 1997.

[11] D. K. Hsing, B. Cheng, G. Goncu, and L. Kant, *A Restoration Methodology based on Pre-Planned Source Routing in ATM Networks,* Proc. IEEE Int. Conference on Communications, 1997.

[12] S. G. Finn, M. M. Menard, and R. A. Barry, *A Novel Approach to Automatic Protection Switching Using Trees,* Proc. IEEE Int. Conference on Communcations, 1997.

[13] M. M. Medard, S. G. Finn, R. A. Barry, and R. G. Gallager, *Redundant Trees for Preplanned Recovery in Arbitrary Vertex-Redundant or Edge-Redundant Graphs,* IEEE/ACM Transactions on Networking, Vol. 7, No. 5, October, 1999.

[14] D. Stamatelakis, W. D. Grover, *Rapid Span or Node Restoration in IP Networks Using Virtual Protection Cycles,* Proc. CCBR'99, 1999.

[15] W. D. Grover, D. Stamatelakis, *Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration,* Proc. CCBR'98, 1998.

[16] W. J. Goralski, *SONET: A Guide to Synchronous Optical Networks,* McGraw-Hill Companies, New York, 1997.

[17] W. J. Goralski, *SONET: A Guide to Synchronous Optical Networks,* McGraw-Hill Companies, New York, 1997.

[18] G. Malkin, *RFC 2453: RIP Version 2,* November, 1998.

[19] J. Moy, *RFC 2328: OSPF Version 2,* April, 1998.

[20] J.T. Moy, *OSPF: Anatomy of an Internet Routing Protocol,* Addison-Wesley Publishing, Reading, Massachusetts, 1998.

[21] T.H. Wu, N. Yoshikai, *ATM Transport and Network Integrity,* Academic Press, San Diego, California, 1997.

[22] S. Halabi, *OSPF Design Guide,* White Paper, Cisco Systems (web site), 1996.

[23] Bellcore, *Generic Requirements for Operations Interfaces Using OSI Tools - Information Model Overview: Synchronous Optical Network (SONET) Transport Information Model,* Document GR-1042-Core, Issue 2, September, 1996.

[24] R. Diestel, *Graduate Texts in Mathematics: Graph Theory,* Springer-Verlag New York, Inc., New York, New York, 1997.

[25] K. Thulasiraman, M. N. S. Swamy, emphGraphs: Theory and Algorithms, John Wiley & Sons, Inc., Toronto, Canada, 1992.

[26] W. L. Winston, *OPERATIONS RESEARCH: Applications and Algorithms,* Duxbury Press, Belmont, California, 1994.

[27] P. Barth, *OPBDP - A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization,*
http://www.mpi-sb.mpg.de/units/ag2/software/opbdp/.

[28] P.Barth, *A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization,* Max-Planck-Institut für Informatik, Germany, 1995.

[29] *Routing and Remote Access Service,*
http://msdn.microsoft.com/library/psdk/rras/rrasport_6r6t.htm.

[30] *Merit GateD Consortium,* http://www.merit.edu/ gated/.

[31] J.A. Rueda, P. Thang, A.S. Alfa, M. Komus, *Network Layer Restoration Project Update,* Telecommunications Research Laboratories technical report TR-98-04, 1998

[32] R. Callon et al., *Internet Draft: A Framework for MultiProtocol Label Switching,* December, 1999.

[33] J. Ryan, *The Technology Guide Series, Multiprotocol Label Switching,* The Applied Technologies Group, Inc., Natick, Massachusetts, 1998.

[34] S. Agrawal, *A Framework for Multiprotocol Label Switching,* Internet Draft, November, 1997.

[35] S. Agrawal, *IP Switching,* http://www.cis.ohio-state.edu/ jqain/cis788-97/ip-switching/index.htm

[36] K.L. Calvert, *Architectural Framework for Active Networks Version 1,* University of Kentucky, July 1999.

[37] D. Wetherall, *Active network vision and reality: lessons from a capsule-based system,* $17^t h$ ACM Symposium on Operating Systems Principles, Dec. 1999.

[38] D. Wetherall, J. Guttag, D. Tennenhouse, *ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols,* IEEE OPENARCH'98, April 1998.

[39] D. Alexander, et al., *The SwitchWare Active Network Architecture,* University of Pennsylvania, July 1998.

[40] D. Alexander, et al., *The SwitchWare Active Network Implementation,* University of Pennsylvania, Sept. 1998.

[41] The Janos Project, *Java-oriented Active Network Operating System,* website: http://www.cs.utah.edu/flux/janos/.

[42] OSKit++, website: http://www.cs.utah.edu/flux/oskit/.

[43] D. Alexander et al., *Active Network Encapsulation Protocol(ANEP),* Active Networks Group, RFC Draft, July, 1997.

[44] D. S. Alexander, W. A. Arbaugh, A. D. Keromytis, J. M. Smith, *Security in Active Networks,* Bell Labs, Lucent Technologies, Distributed Systems Lab, 1999.

[45] J. Moy, *RFC 2328: OSPF Version 2,* April, 1998.

[46] S. Armstrong, A. Freier, K. Marzullo, *RFC 1301: Multicast Transport Protocol,* February, 1992.

[47] R. Braudes, S. Zabele, *RFC 1458: Requirements for Multicast Protocols,* May, 1993.

[48] J. Moy, *RFC 1584: Multicast Extensions to OSPF,* March, 1994.

[49] D. Eppstein, *Offline Algorithms for Dynamic Minimum Spanning Tree Problems,* Tech. Report 92-04. Dept. of Info. and Comp. Sci., University of California, Irvine, CA, 1992.