UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

LOGICAL TOPOLOGY DESIGN FOR SURVIVABILITY IN IP-OVER-WDM NETWORKS

A Dissertation

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

MUHAMMAD S. JAVED
Norman, Oklahoma
2009

LOGICAL TOPOLOGY DESIGN FOR SURVIVABILITY
IN IP-OVER-WDM NETWORKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

THIS DISSERTATION IS DEDICATED

TO

My parents

Ghulam Muhammad and Safia Ghulam Muhammad

Acknowledgements

This thesis would have not been possible without the kind and consistent support of many people.

First and foremost, my utmost gratitude goes to my Ph.D. advisor Dr. Krishnaiyan Thulasiraman. His unflinching support, encouragement, enthusiasm, patience and expertise kept me motivated during my thesis-writing and research.

I am extremely grateful to my thesis committee members: Dr. Pramode K. Verma, Dr. Sridhar Radhakrishnan, Dr. Mohammed Atiquzzaman and Dr. Qi Cheng for their insightful questions and comments.

I am also indebted to my many student colleagues and friends: Emily Hamm, Shankar Banik, Aravind Canthadai, Tachun (Robert) Lin, Lavanya Sivakumar, Olawale Solano and Zhili Zhou for stimulating discussions and readiness to help.

I am happy to acknowledge my debt to the wonderful staff at the Computer Science department, especially Barbara Bledsoe and Chyrl Yerdon, for ensuring smooth and timely delivery of this thesis.

Lastly, I would like to thank each and everyone in my family for their boundless love, support and patience, especially my parents Ghulam Muhammad and Safia Ghulam Muhammad, and brother Shahbaz.

Contents

List of Tables

List of Figures

Abstract

IP-over-WDM networks integrate Wavelength Division Multiplexing (WDM) technology with Internet Protocol (IP) and are widely regarded as the architecture for the next generation high-speed Internet. The problem of designing an IP-over-WDM network can be modeled as an embedding problem in which an IP network is embedded in a WDM network by establishing all optical paths between IP routers in the WDM network. Survivability is considered a vital requirement in such networks, which can be achieved by embedding the IP network in the WDM network in such a way that the IP network stays connected in the presence of failure or failures in the WDM network. Otherwise, some of the IP routers may not be reachable.

The problem can be formulated as an Integer Linear Program (ILP), which can be solved optimally but is NP-complete. In this thesis, we have studied and proposed various efficient algorithms that can be used to make IP-over-WDM networks survivable in the presence of a single WDM link (optical fiber cable or cables) failure.

First we evaluate an existing approach, named Survivable Mapping Algorithm by Ring Trimming (SMART), which provides survivability for an entire network by successively considering pieces of the network. The evaluation provides much insight into the approach, which allowed us to propose several enhancements. The modified approach with enhancements leads to better performance than the original SMART.

We have also proposed a hybrid algorithm that guarantees survivability, if the

IP and the WDM networks are at least 2-edge connected. The algorithm uses a combination of proactive (protection) and reactive (restoration) mechanisms to obtain a survivable embedding for any given IP network in any given WDM network.

Circuits and cutsets are dual concepts. SMART approach is based on circuits. The question then arises whether there exists a dual methodology based on cutsets. We investigate this question and provide much needed insight. We provide a unified algorithmic framework based on circuits and cutsets. We also provide new methodologies based on cutsets and give a new proof of correctness of SMART. We also develop a method based on incidence sets that are a special case of cutsets. Noting that for some IP networks a survivable embedding may not exist, the option of adding new IP links is pursued. Comparative evaluations of all the algorithms through extensive simulations are also given in this dissertation.

Chapter 1

Introduction

1.1 Overview

Over the last decade, Internet has seen an exponential worldwide increase in the number of users. At the same time, Internet usage has shifted away from simple applications such as web surfing, email, etc., to bandwidth intensive online services and applications such as streaming video, voice, gaming etc. This phenomenon has compelled the internet service providers (ISPs), providing such services using traditional networks, to invest in new architectures and technologies that can cope with the current bandwidth demand as well as projected future requirements.

An approach that has attracted significant interest from network designers and researchers is to replace (or upgrade) legacy networks with all optical networks (AON) using wavelength division multiplexing (WDM) technology. AONs are high-speed optical fiber networks that perform the common network operations (switching, routing, amplification etc.) without optical-electrical-optical conversion (O-E-O). By integrating AONs with WDM technology, which allows simultaneous transmission of several signals through a single fiber, bandwidth of several terabits per seconds can be achieved.

Advances in optics and photonics now allow more flexible network architectures (e.g. mesh networks) that can transparently run a host of prevailing network protocols (SONET/SDH, ATM, and TCP/IP) on the top of the optical (WDM) layer. Since most of the end user communications today rely on TCP/IP protocol, the idea of implementing IP protocol directly over the WDM layer using optical cross-connects (OXCs) and IP routers has emerged as the winning combination for the new Internet. Such networks are generally called IP-over-WDM networks.

A commonly proposed approach to implement IP protocol over the WDM network is to *map* or *embed* an IP topology, referred to as the *logical topology*, in a physical WDM topology, commonly called the *physical topology.* The *mapping* involves finding a *lightpath* for an IP (logical) link in the physical topology, which connects the two end points (source and destination) of the logical link. A *lightpath* is an all-optical path in the physical topology, which is established by allocating a wavelength between the source and the destination of an IP link. A lightpath, once established, does not require processing or buffering at intermediate logical nodes and quite possibly no intermediate O-E-O conversions.

In an IP-over-WDM network, a single fiber generally carries several lightpaths simultaneously and usually several fibers are bundled together to form a cable (a physical link). Therefore, a cable cut in the network can disrupt all the lightpaths passing through the cable and degrade the network performance significantly, if the failure persists. Unfortunately, cable cuts and equipment failures have become a common occurrence due to human or natural events, drawing considerable attention to designing networks that can provide an acceptable level of service in the presence of such failures. Such networks are generally called *survivable networks.*

Protection and *restoration* are the two widely discussed mechanisms for providing survivability in IP-over-WDM networks. Protection in IP-over-WDM networks is generally provided at the physical layer at the design stage. First *primary* or *work-*

ing lightpaths are established for the logical links and then *backup* or *protection* lightpaths are calculated that do not use physical links (or nodes) already assigned to their respective primary lightpaths (i.e. their mappings are disjoint). In case of a failure, the network traffic carried by a primary lightpath is always switched to its corresponding backup lightpath. Since protection requires explicit reservation of resources, it is generally very fast but inefficient in terms of resource utilization.

Restoration is generally provided at the logical layer by provisioning the network with some additional capacity, which can be utilized by the IP routers to find backup paths after a failure. It is possible to find backup paths only if the logical links are mapped onto the physical topology in such a way that the logical topology remains connected after the failure. This can be achieved by requiring the logical and physical topologies to be at least 2-edge connected and finding link/node disjoint mappings for some or all the logical links. A mapping of the logical links in the physical network that remains connected after the failure of physical link/links is called a *link survivable mapping* and a mapping that stays connected after the failure of physical node/nodes is called a *node survivable mapping.*

The problem of finding link/node survivable mappings is known to be NP-complete [1]. Therefore, efficient algorithms to solve the problem are unlikely in full generality. Therefore, this thesis focuses on developing efficient heuristics to find *one link survivable* mappings i.e. mappings that leave the logical topology connected after the failure of a single physical link.

1.2 Thesis Organization and Contributions

The organization of the thesis is as follows. In Chapter 2, we will introduce some of the commonly proposed physical (WDM) and logical (IP, SONET, ATM) layer survivability schemes. In Chapter 3, we will formally introduce survivable IP-over-

WDM networks and discuss various approaches proposed in the literature to design such networks. In Chapter 4, we will analyze in detail the *Survivable Mapping Algorithm by Ring Trimming* (SMART) framework proposed by M. Kurant and P. Thiran [2], and suggest enhancements that can improve the success rate and performance of the approach. We will also provide a comprehensive approach to find one link survivable mappings in Chapter 4. Chapter 5 will discuss a framework that guarantees survivability in IP-over-WDM networks using a combination of protection and restoration schemes. Taking advantage of the duality that exists between circuits and cutsets in a graph, a unified algorithmic framework for the survivable logical topological design problem will be presented in chapter 6. In Chapter 7, we will provide the summary of the work presented and point to some future research directions.

Chapter 2

WDM Optical Networks and Survivability Mechanisms

This chapter provides a brief introduction to Wavelength Division Multiplexing (WDM) networks and discusses some commonly used WDM architectures. It also introduces the concept of survivable networks, networks that provide an acceptable level of service in the presence of failures, and discusses various mechanisms that can be employed to make a given network survivable.

2.1 WDM Optical Networks

Optical fibers possess several properties which made them the ideal replacement of copper cables in the traditional telephone networks. Optical fibers are not only low cost, lightweight, and difficult to wiretap, but they also offer very high bit rates (up to 160 Gbps), better signal quality (optical fibers have an approximate Bit Error Rate (BER) of 10^{-12} to10^{-14} compared to 10^{-3} to 10^{-4} for copper wires), and are immune to electro-magnetic and radio-frequency interference (EMI/RFI) [3].

Initially, optical fibers were mainly used as transmission links by phone companies to upgrade their trunk lines from copper wires. Trunk lines are always digital and employ time division multiplexing (TDM) to support several simultaneous voice connections. These pure point-to-point systems or networks are the simplest form of optical networks and are usually set up using an optical transmitter, a fiber and an optical receiver. An optical transmitter is essentially a light source that converts data into a sequence of on/off light pulses of a particular wavelength (λ), which travel through the optical fiber and arrive at the receiver. The receiver then converts the light pulses back into data. Fig. 2.1 shows one such network.

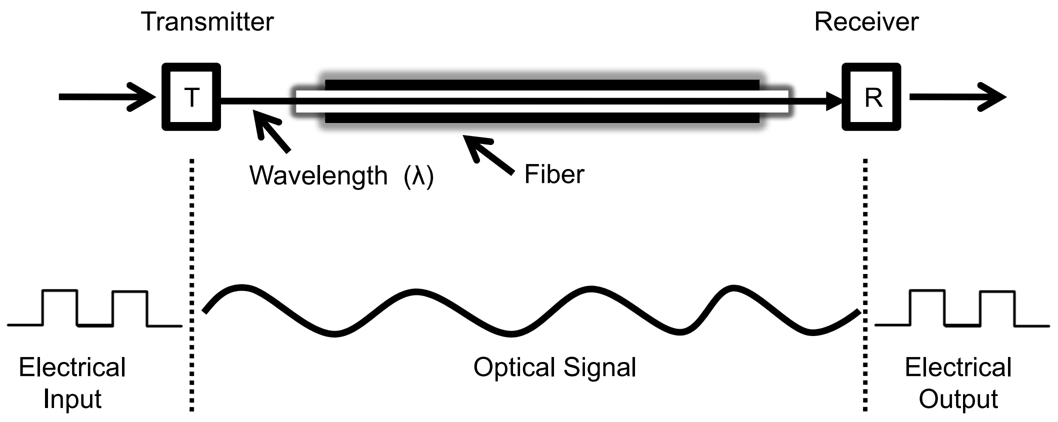


Figure 2.1: A basic optical network.

With the advent of Wavelength Division Multiplexing (WDM) technology, the phone companies switched to coarse or dense wavelength division multiplexing (CWDM and DWDM, respectively) to improve transmission speeds and capacity. WDM technology allows a single fiber to simultaneously carry multiple optical signals (channels), each modulating at a unique wavelength. A wavelength can be thought of as a different color of light in the infrared spectrum that can carry data. Since the number of wavelengths that a fiber can carry is generally limited by the end equipment (e.g. transmitters, receivers, multiplexer/de-multiplexer etc.) not by the fiber itself, optical fiber networks employing WDM technology offer unprecedented scalability. Such networks are usually called WDM optical networks or simply WDM networks.

Early commercial WDM networks were point-to-point networks that appeared in 1995. They were based on 2.5 Gbps per channel (wavelength) with 8 or 16 available wavelengths and did not require regeneration of signal up to a distance of 750 miles (1200 km) [4]. These are considered the first generation WDM networks and required manual connection set up. Fig. 2.2 shows a basic WDM optical network and Fig 2.3 shows two point-to-point WDM networks connecting three facilities. Such networks cannot perform network operations in optical domain. Therefore, optical to electrical to optical (O-E-O) conversion must be performed if signal switching is required. This phenomenon, generally called *electrical bottleneck*, limits the throughput of the network to rates compatible with the electronic circuitry of the switching equipment.

The popularity of packet switched World Wide Web (WWW) or Internet in the 1980s and 1990s created a tremendous appetite for more capacity and speed. Given the fact that it is extremely costly to install new fibers to increase transmission capacity, the telecommunication research community focused on increasing the number of wavelengths a fiber can carry and the bit rate. In 1998, the second generation of WDM networks replaced the first generation that were characterized by 10 Gbps channels, 40 channels per fiber and semi-automatic connection set up [4]. Such networks used optical add-drop multiplexers (OADMs) to provide limited networking functionality. However, OADMs can be used only in point-to-point or ring networks to add or drop signals and have the ability to remove the electrical bottleneck to some extent as shown in Fig 2.4.

Introduction of OADMs in WDM networks led to the introduction of *lightpath communications* [5]. A *lightpath* is an all-optical path in the WDM network, which is established by the allocation of a particular wavelength between a pair of facilities that may or may not be adjacent to each other. Once established, a lightpath may traverse through multiple fibers without requiring buffering, processing, and quite possibly no O-E-O conversion at the intermediate facilities [5]. However, in some cases it may not be possible to avoid O-E-O conversion. This may occur when a wavelength is not available to connect a pair of facilities.

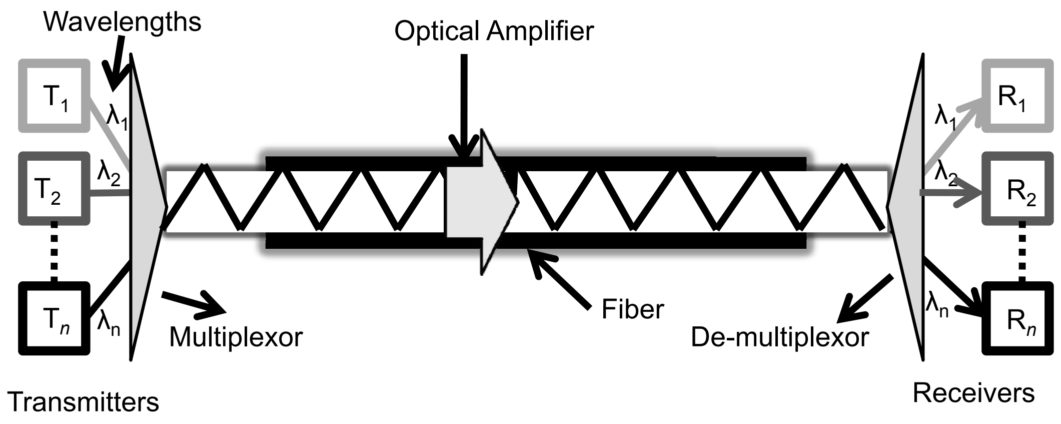


Figure 2.2: A basic WDM optical network.

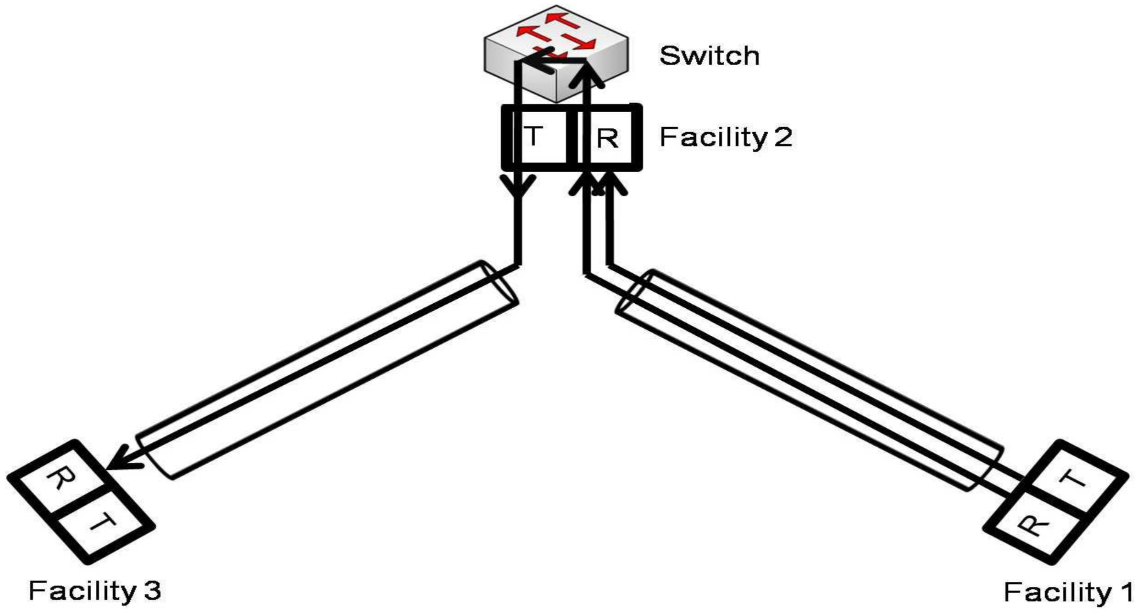


Figure 2.3: Three facilities connected by two point-to-point WDM networks.

The ever increasing demand for more bandwidth has kept the focus of the research community on developing new types of fibers and enabling networking equip-

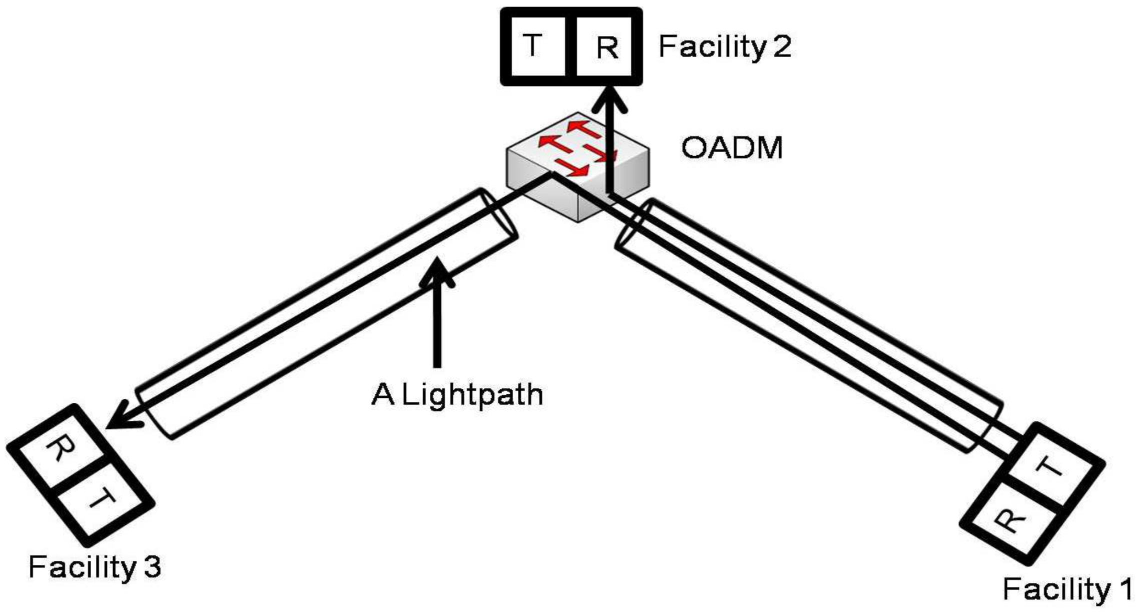


Figure 2.4: A point-to-point WDM network with OADM.

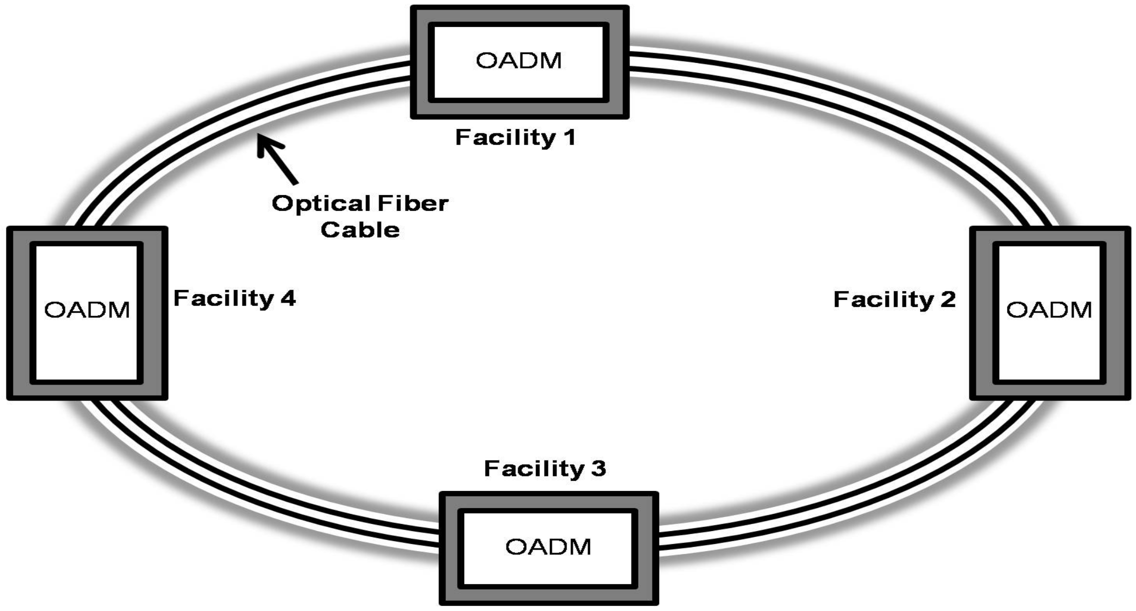


Figure 2.5: A ring WDM network.

ment. Prevailing experimental technologies allow 160 signals per fiber each modulating at 160 Gbps, providing a total bandwidth of 25.6 Tbps over three 80 km long single fiber strands without signal regeneration or amplification [6]. Since a fiber optic cable generally contains several hundred fiber strands, an aggregate throughput of several thousand terabits per second can be achieved. Additionally, the development of optical cross-connects (OXC), optical amplifiers (OA), tunable transmitters/receivers, wavelength converters (WC) etc., now allow more flexible network configurations such as ring and mesh networks. Fig. 2.5 and 2.6 show a ring and a mesh WDM network, respectively.

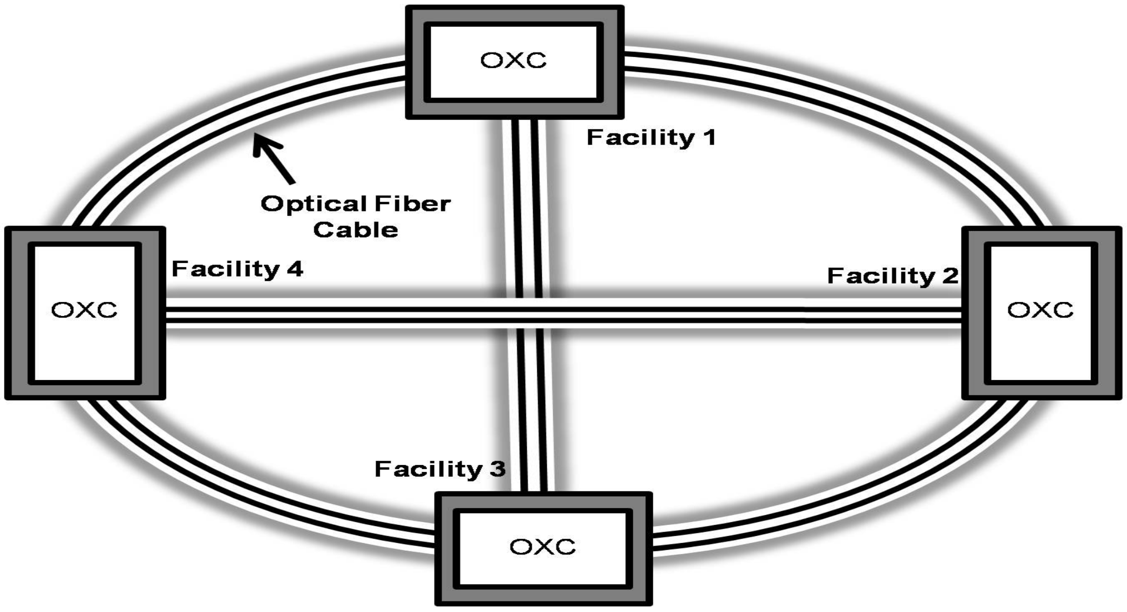


Figure 2.6: A mesh WDM network.

An optical cross-connect can be used to add and drop signals as well as to switch traffic from one fiber to another without any O-E-O conversion and a wavelength converter converts a signal at one wavelength to another without O-E-O conversion. OXCs and WCs, when used together in a WDM network, remove the need for O-E-O conversions. Such WDM networks are also called *All Optical WDM Networks* (AONs).

2.1.1 Challenges in WDM Networks

Modern WDM networks utilizing all optical networking technologies to provide tremendous bandwidth have posed several challenges to network designers and operators. One such challenge is to determine the best way to effectively utilize the tremendous bandwidth available using existing networking protocols such as Internet Protocol (IP), Asynchronous Transfer Mode (ATM), Synchronous Optical Network/Synchronous Digital Hierarchy (SONET/SDH) etc. The widespread use of these protocols makes it very difficult to modify them or add new functionalities. Therefore, the most commonly proposed approach to effectively exploit the high bandwidth WDM networks is a layered approach, in which the WDM layer is transparent or invisible to protocols such as IP, ATM, SONET/SDH etc.

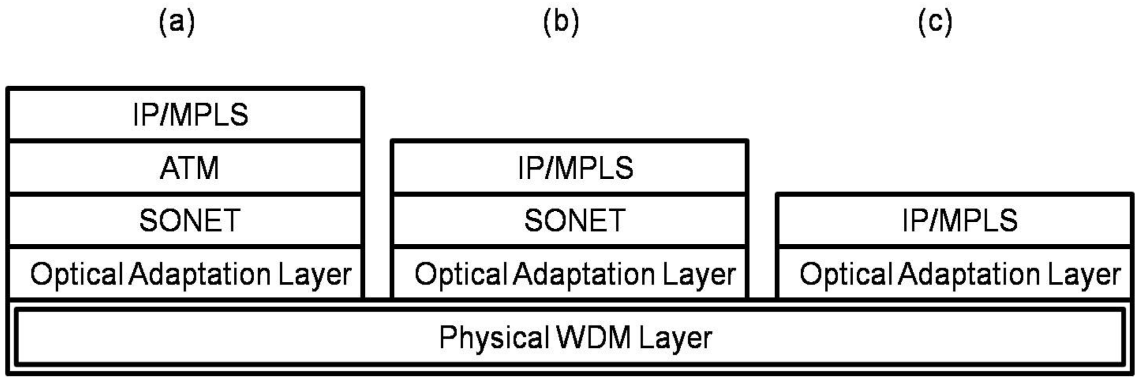


Figure 2.7: Protocol stacks for WDM networks.

In the layered approach optical fiber cables, OXCs, OADMs, OAs etc. form the *physical network* or the *WDM layer* and IP routers, ATM switches, SONET/SDH rings etc. represent the *logical network* or *layer*. In the physical network all the network operations i.e. switching, routing, amplification etc. are performed in the optical domain. However, a logical layer accomplishes the network operations using equipment that employs electronic circuitry. Fig. 2.7 shows only a few possible combinations of logical and physical layers that can be used to exploit WDM networks [7] and Fig. 2.8 shows an example of the protocol stack shown in Fig. 2.7(a).

Another challenge that the network operators and designers must address is the massive amount of data loss that may occur after the failure of a network component or components. Usually, several hundred optical fibers strands are bundled together to form a cable. Such cable or cables are then buried along with other utility

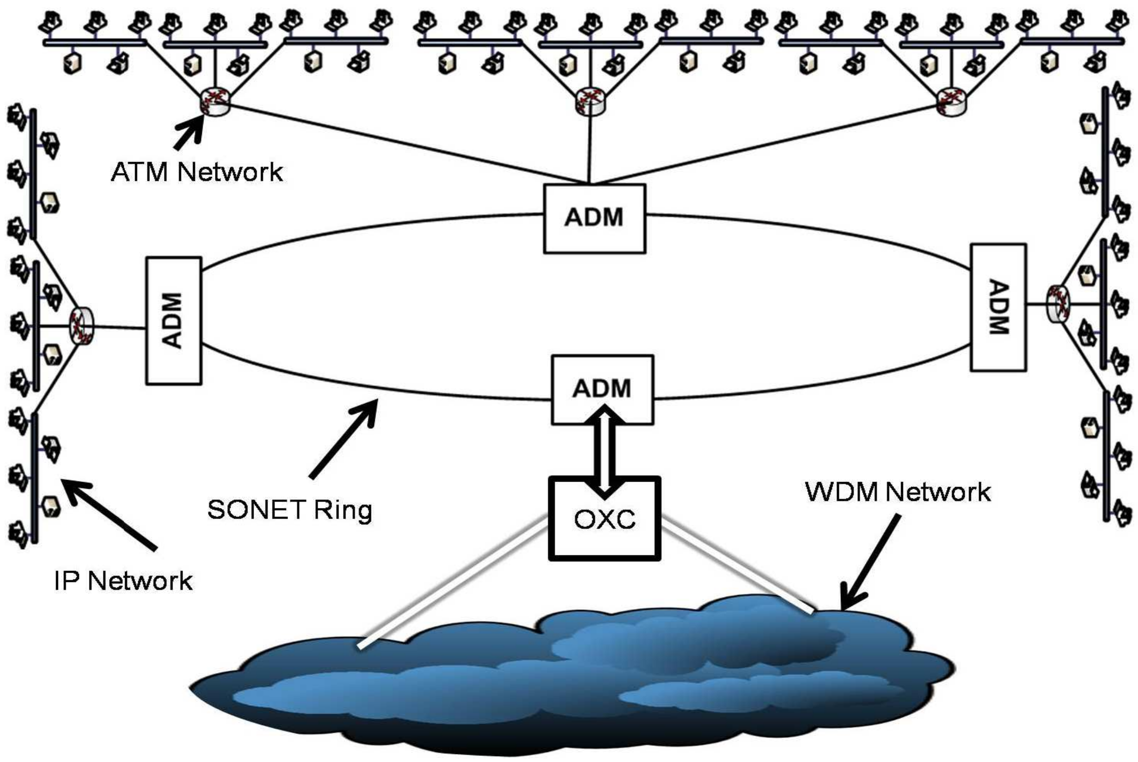


Figure 2.8: An IP-over-ATM-over-SONET-over-WDM network.

services like cable, telephone, water etc. to connect remote facilities. These utility services require continuous maintenance and upgrades, thereby exposing the fiber optic cables to damage and failure. Other networking components like OXCs, OAs etc. may also fail or malfunction but this situation can be remedied by providing redundant equipment. However, it is much more difficult to address fiber failures and consequently has drawn more attention from researchers.

The most frequent cause of fiber failures in WDM networks is a fiber cut due to human (construction/repair work, vandalism etc.) or natural (earthquakes, lightning, fire, etc.) events. Furthermore, the time required to precisely determine the location of the cut and digging up of the cable to perform the repairs is usually significant. Therefore, given the facts that a single fiber may carry enormous amount of data which is lost in the event of its failure and the time required to repair the failure, even a single fiber failure can affect the performance of the entire network.

The resulting significant degradation in performance can be mitigated by employing mechanisms that allow WDM networks to provide an acceptable level of service in the presence of a failure or failures. WDM networks with built-in mechanisms that allow them to continue to deliver an acceptable level of service in the presence of a fiber failure or fiber failures are generally called *link survivable WDM networks.* Similarly, *node survivable WDM networks* are able to provide an acceptable level of service after the failure of networking equipment such as OXCs, OAs, etc.

The focus of this thesis is to study link survivable WDM networks, more specifically *one link survivable WDM networks.* One link survivable WDM networks are able deliver an acceptable level of service in the presence of a single link failure. Henceforth, a fiber failure/cut implies that all the fibers between a pair of networking facilities are severed and one link survivable WDM networks are simply referred to as survivable WDM networks.

2.2 Link Survivable WDM Networks

Link survivability mechanisms are broadly classified into two categories, namely *protection* and *restoration* [3]. Protection is a pre-planned proactive mechanism in which network resources such as fibers, transmitters/receivers, wavelengths, routers etc. are explicitly reserved for various failure scenarios. When a fiber fails, the network traffic (wavelengths) carried by the affected fiber is simply switched to the resources reserved for this failure scenario. Protection is very fast (on the order of milliseconds) but inefficient in terms of capacity utilization as the reserved resources are idle in the absence of a failure [3].

On the contrary, restoration is a reactive mechanism in which no network resources are set aside for various failure scenarios but the network is provisioned with some extra capacity which can be used to carry the failed fiber's network traffic [3]. In restoration, the selection of network resources to use in the event of a failure is made after the failure has taken place. Restoration is generally considered slow (usually on the order of a few seconds) but more resource efficient [3].

It is important to note that restoration and protection mechanisms can be implemented only if a network has some degree of redundancy built into it.

As mentioned earlier, WDM networks can be subdivided into physical and logical layers, which allows the flexibility to implement survivability mechanisms at physical layer or logical layer or both. Protection is generally associated with the WDM layer. However, it is expected that advances in the development of WDM routers will allow the flexibility to implement restoration at the WDM layer. Logical layer can employ both protection and restoration, but restoration is the preferred mechanism for logical layer.

2.3 WDM Layer Survivability Mechanisms

Protection mechanisms are more commonly employed at the WDM layer and greatly depend on the configuration of network under consideration (e.g. point-to-point, rings, mesh etc.). This section provides a brief overview of the WDM layer survivability mechanisms [3].

2.3.1 Point-to-Point Networks

In point-to-point networks, *automatic protection switching* (APS) is the most widely accepted protection mechanism. APS requires spare fibers which must be buried along a route not being used by the main fibers. The main fibers are generally referred to as *primary* or *working links* and the spare fibers are called *protection* or *backup links.*

There are three main types of APS systems, namely 1:1, 1+1 and 1:N APS [3].

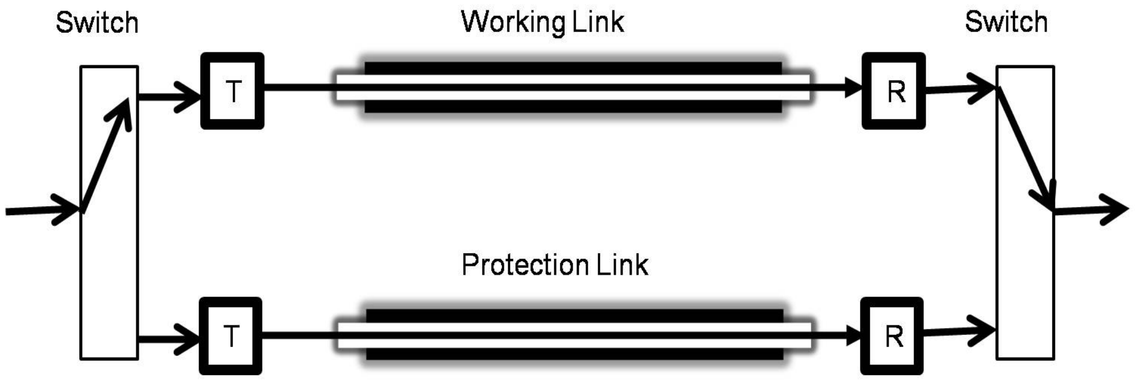


Figure 2.9: 1:1 APS point-to-point network.

1:1 Automatic Protection Switch (1:1 APS)

In 1:1 APS, the normal network traffic is carried by a primary link and after the failure the traffic is switched to the protection link. In some cases, the protection link may carry low priority traffic which is preempted when the working link fails. Figure 2.9 shows a 1:1 APS point-to-point network.

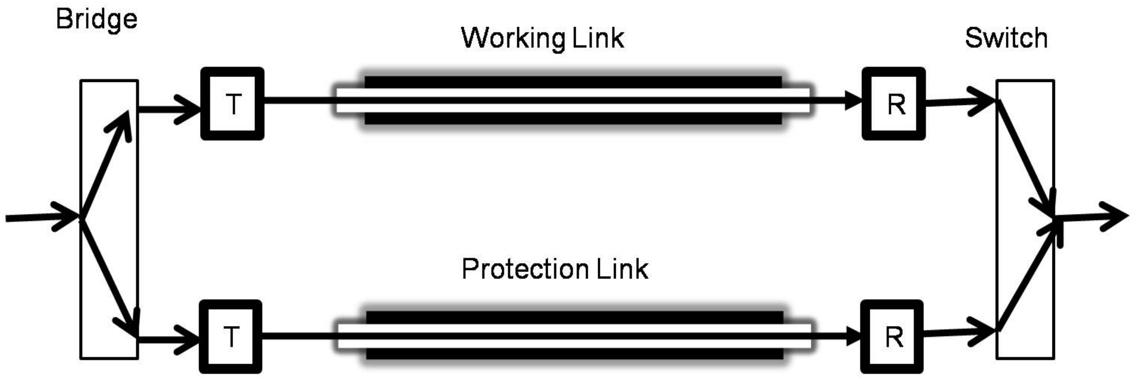


Figure 2.10: 1+1 APS point-to-point network.

1+1 Automatic Protection Switch (1+1 APS)

In 1+1 APS, the normal network traffic is carried by a primary link and an exact copy of this traffic is also carried on the protection link. The receiver chooses the signal of better quality. Figure 2.10 shows a 1+1 APS point-to-point network.

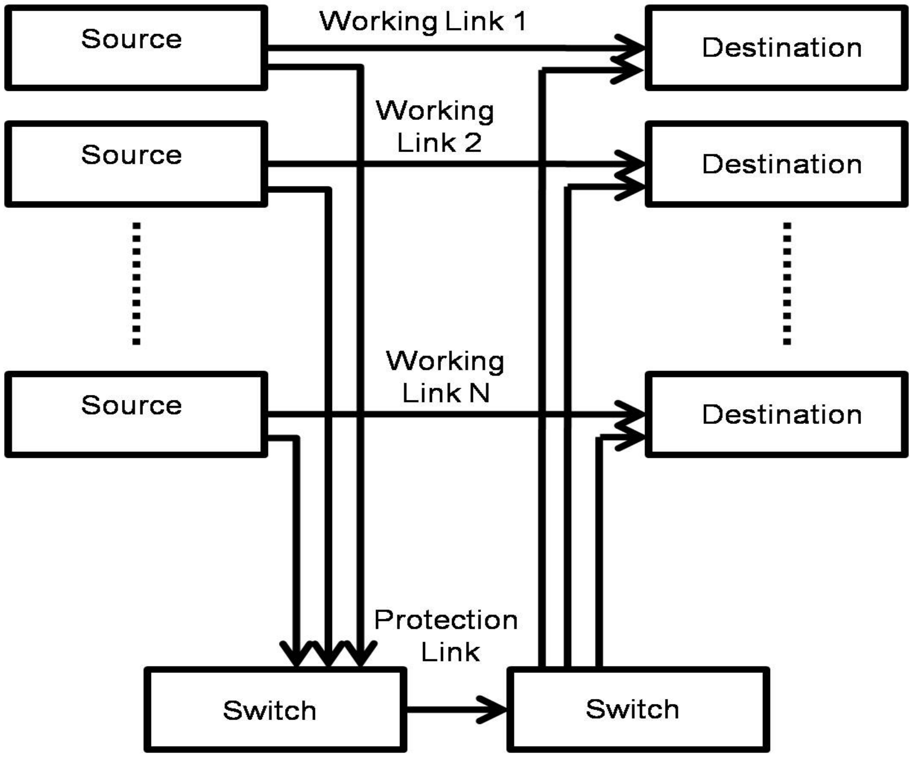


Figure 2.11: 1:N APS point-to-point networks.

1:N Automatic Protection Switch (1:N APS)

In 1:N APS, one protection link is shared by many working links that are not expected to fail at the same time. When a working link fails, the traffic carried by this link is switched to the protection link. The traffic is switched back to the working link, when it recovers from the failure. Figure 2.11 shows a 1:N APS point-to-point network.

2.3.2 Ring Networks

The concept of APS has also been used in ring networks to design *Self Healing Rings* (SHRs), which protect the networks designed in the form of a ring. Fig. 2.12 shows a SHR with two fibers. The working traffic flows in one direction (clockwise or anticlockwise) on a fiber and in case of a failure the working traffic carried by the failed link flows in the opposite direction on the protection fiber (similar to 1:1

APS). In some cases, the signal is transmitted on both working and protection fibers and the nodes decide which signal to choose (similar to 1+1 APS).

These methods require that the spare capacity reserved must be equal to the working capacity of the network, which makes the network very inefficient but the time to recover from a failure is negligible, usually on the order of 50 *ms* [3].

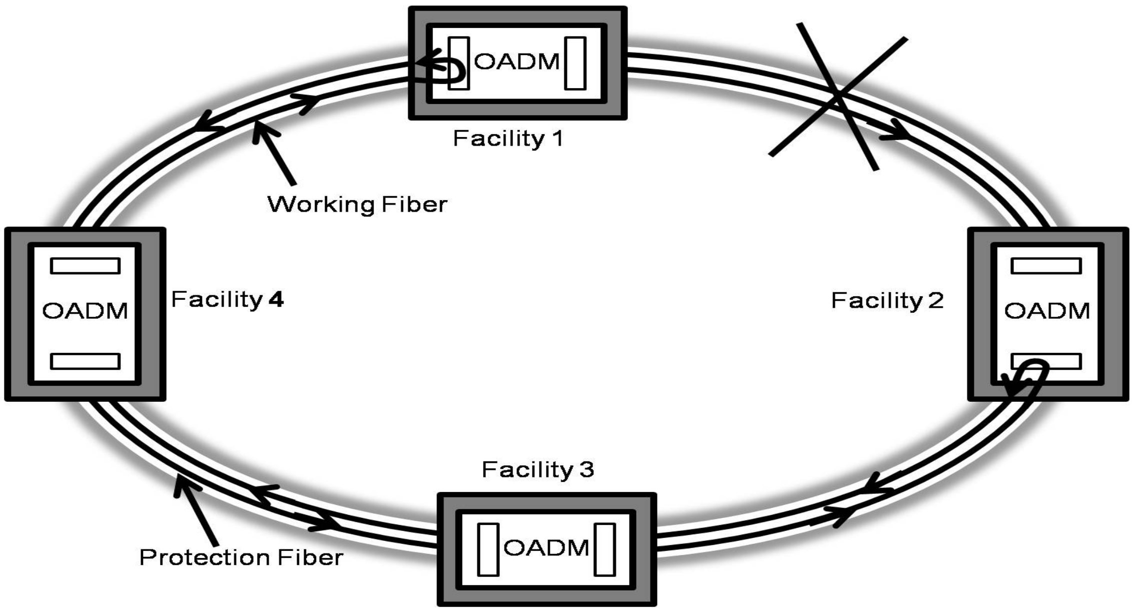


Figure 2.12: Protection in a ring network using a self healing ring (SHR).

2.3.3 Mesh Networks

Mesh networks have higher link diversity that could lead to lower bandwidth redundancy but the problem of designing fast protection mechanisms becomes more difficult. Various preplanned protection techniques have been proposed for mesh networks. One such technique is *Pure Ring Covers* (PRCs) that finds multiple logical rings to cover all the links which then work as a collection of SHRs. Fig. 2.13 provides an example of PRC. However, PRCs require at least 100% redundancy but in real networks it is sometimes more than 200% [3].

To reduce redundancy, Grover and Stamatelakis introduced the concept of *Pre-*

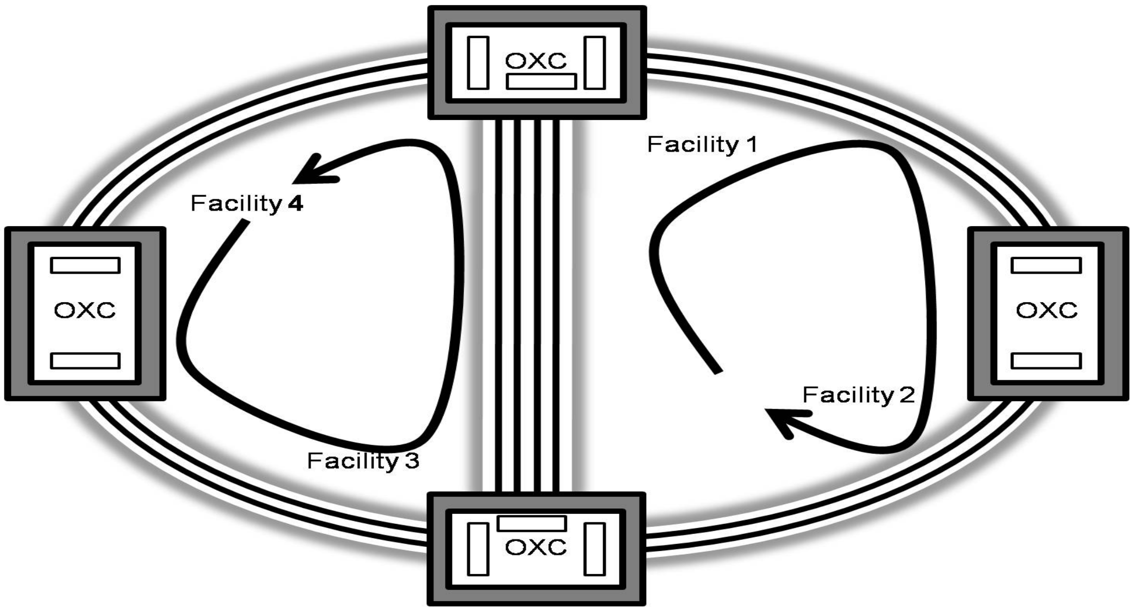
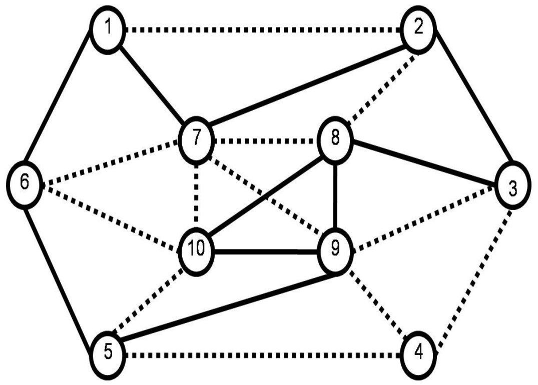


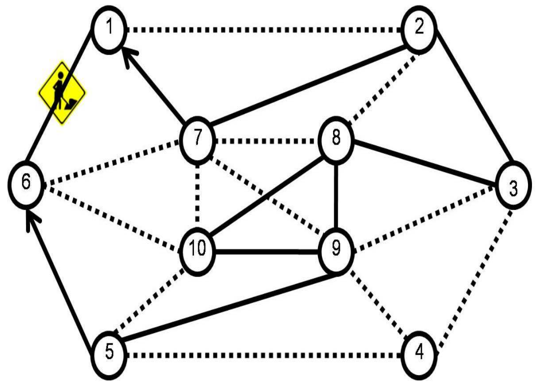
Figure 2.13: A mesh network protected by two rings (Pure Ring Covers).

configured Protection Cycles (P-cycles) to protect mesh networks against a single link failure [8]. P-cycles require significantly less spare capacity than the other protection mechanisms for mesh networks. P-cycles typically require a redundancy of 50-70% in well-connected physical networks. P-cycles are based on the ability of a ring to protect not only the links that form the ring but also any possible *straddling links*. A *straddling link* is a link which is not part of the ring but its end points lie on the ring. Figure 2.14(a) shows a P-cycle that provides a single protection path for 9 on cycle links and two paths for the six straddling links. Fig. 2.14(b) shows the behavior of the P-cycle when a link on the cycle fails. In this case, the P-cycle behaves like a SHR. Fig. 2.14(c) and 2.14(d) show the behavior of the P-cycle when a straddling link fails. In these cases two paths are available to protect the failed links.

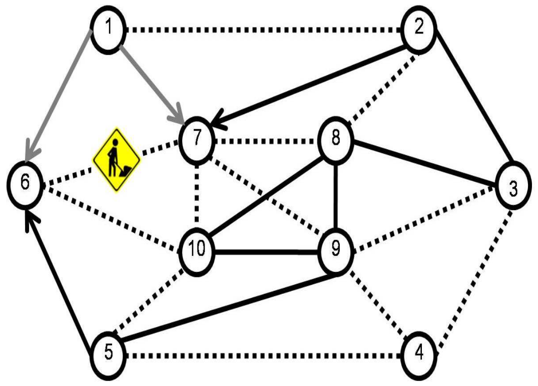
By using a set of carefully designed P-cycles, it is possible to protect all the links in a mesh network. To minimize the spare to working resources ratio, most of the methods proposed in the literature to design P-cycles are based on Integer Linear Programs (ILPs). In fact the problem of finding a P-cycle or a set of P-cycles that minimize the spare to working resource ratio is NP-complete [9].



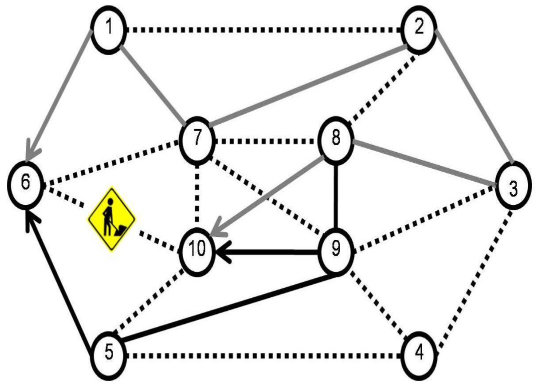
(a) A P-cycle (bold line)



(b) A span on the cycle fails (1-6), the P-cycle contributes one restoration path (BSHR like behavior).



(c) A span off P-cycle fails (6-7), the P-cycle contributes two restoration paths (mesh like).



(d) A span off P-cycle fails (6-10), the P-cycle contributes two restoration paths (mesh like).

Figure 2.14: P-cycle behavior.

2.4 Logical Layer Survivability Mechanisms

As shown in Fig. 2.7, the logical layer may consist of several different protocols, which have well developed survivability mechanisms. Some of these mechanisms are discussed below.

2.4.1 Transmission Control Protocol/Internet Protocol (TCP/IP)

TCP/IP networks have a built-in mechanism to reroute traffic around a failed network component through use of various routing protocols such as *Routing Information Protocol* (RIP), *Open Shortest Path First* (OSPF), etc. In TCP/IP networks, after detecting a failure, the IP routers compute the alternate routes for affected traffic based on the network topology after the failure (restoration).

2.4.2 Asynchronous Transmission Mode (ATM)

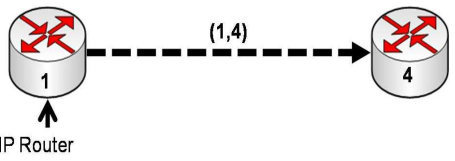
ATM is a connection oriented protocol. Therefore, a connection must be established before data transmission could begin. Such a connection is usually called the *primary* or *working virtual path* (VP). Restoration is provided in an ATM network by calculating a *backup* VP for the failed working VP after a failure [10]. The backup VP is selected in such a way that it avoids the failed network component. It is also possible to provide protection in ATM networks by pre-computing the backup VPs for the working VPs such that a failure does not affect both the working and the backup VPs [11].

2.4.3 Synchronous Optical Network (SONET)

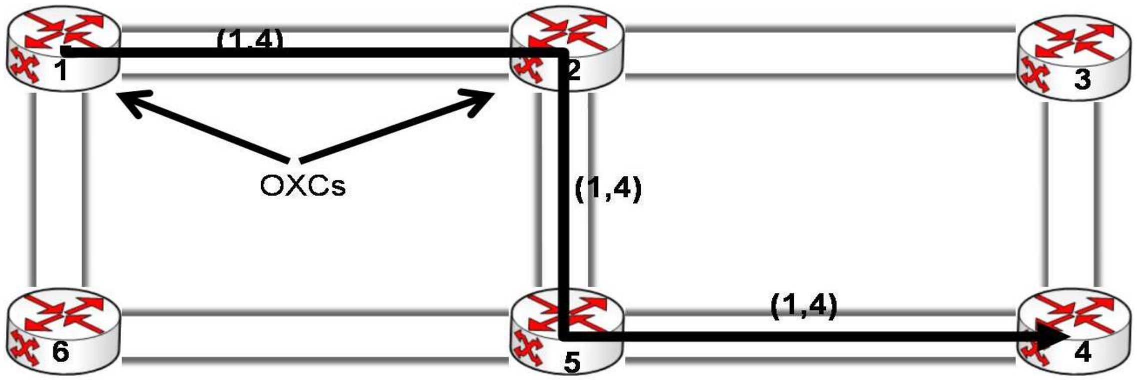
A SONET is an optical network that is set up using digital cross-connect switches (DCS). The DCSs are responsible for switching, failure detection and restoration in SONET. In case of a failure, the DCSs dynamically establish alternate paths for the traffic on the failed link by utilizing the available spare capacity (restoration).

2.5 WDM Layer Survivability or Logical Layer Survivability?

In WDM networks, the logical layer and the WDM layer both are capable of providing survivability mechanisms. The decision to choose the appropriate layer to provide survivability depends on several factors such as the cost involved in terms of network resources, desired speed of recovery from a failure, overhead involved in terms of signaling, etc [12] [13].



(a) A logical link (IP Link).



(b) A lightpath corresponding to the logical link (1, 4) in physical topology.

Figure 2.15: Concept of a lightpath.

WDM layer survivability mechanisms are extremely fast, usually on the order of 50 milliseconds. However, they require greater link diversity that may necessitate installation of new fibers which is expensive and difficult to achieve due to right of way limitations, geographical restrictions etc. It is expected that the availability of all optical routers will allow better utilization of resources at the expense of recovery speed.

As shown in Fig. 2.7, a logical layer may consist of several different protocols, each with its own survivability mechanism. Since it is unreasonable to expect changes in logical layer protocols due to their widespread deployment, unnecessary signaling between different layers may be unavoidable in the event of a failure. For example, an ATM switch may detect a failure and initiate the protocol to find backup paths. But before the process completes, the IP router on top of the ATM switch may time out and start its own protocol to find backup paths. Also, adding more layers can significantly increase the set up cost of the network e.g. the WDM network shown in Fig. 2.8 requires IP routers, ATM switches and SONET rings. Removing the ATM switches and implementing the network according to the protocol stack shown in Fig. 2.7(b), network operators can significantly reduce the setup cost.

The protocol stack that has gained significant attention from the research community is shown in Fig. 2.7(c). TCP/IP is currently the most commonly deployed end user communication protocol. By running TCP/IP directly over the physical layer, setup costs and signaling overheads can be considerably reduced. Such networks are termed *IP-over-WDM networks*. IP-over-WDM networks and some of the challenges they pose are discussed below.

2.6 IP-over-WDM Networks

In an IP-over-WDM network, Internet Protocol (IP) is implemented directly over the WDM optical network using IP routers, optical cross-connects (OXCs) and optical fibers [14]. A commonly proposed method to implement IP-over-WDM is to map an IP network, referred to as *logical topology*, into a physical WDM topology, commonly called the *physical topology* [2]. An IP router is also referred to as a *logical node* (vertex) and an OXC is also called a *WDM* or *physical node* (vertex). A pair of logical nodes is connected to each other via a logical connection called

IP or *logical link* (or edge). A pair of physical nodes is connected through actual optical fibers bundled together to form a cable; the cable is called a *physical link*. It is also common to assume that a logical node is always a node in the physical topology and all the physical nodes may not be present in the logical topology.

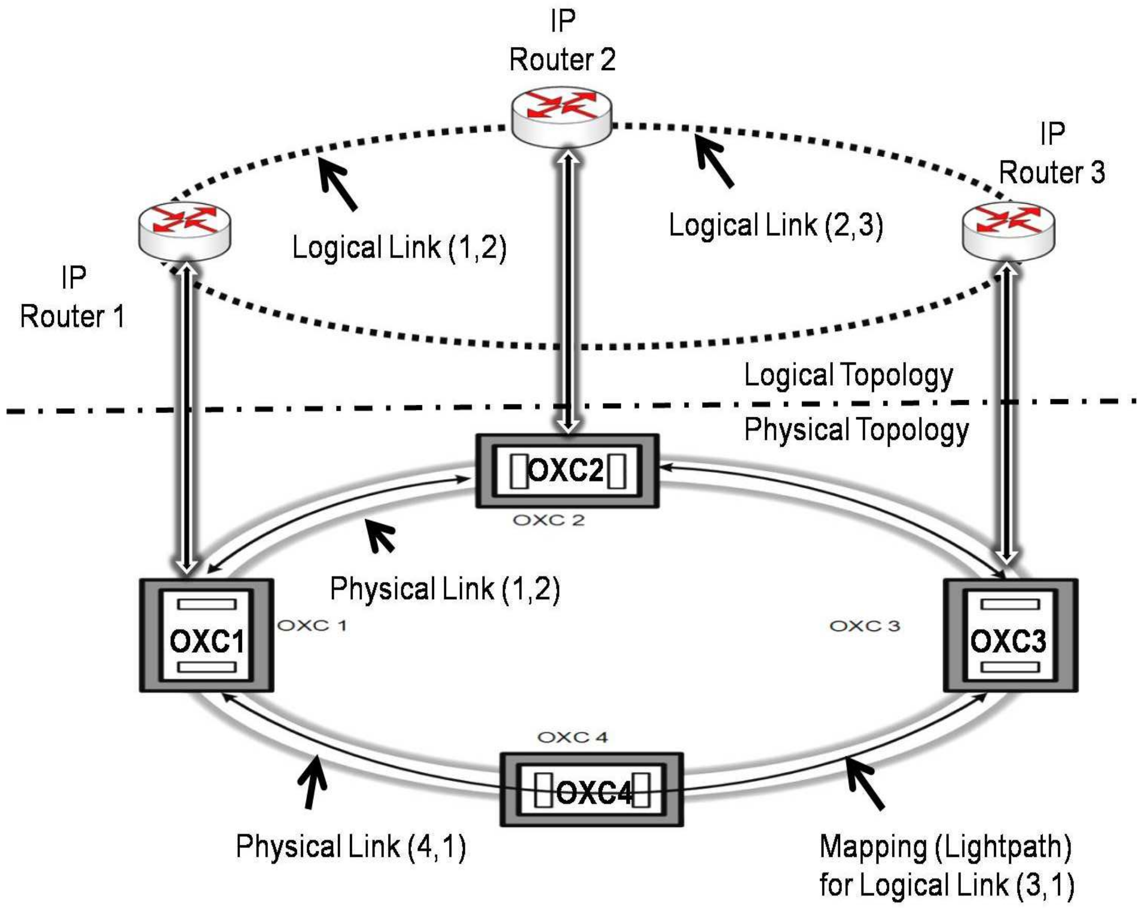


Figure 2.16: An IP-over-WDM network.

The *mapping* of a logical topology into the physical topology involves finding a *lightpath* for each IP (logical) link in the physical topology [2]. A *lightpath* is an all-optical path in the physical topology, which is established by the allocation of a wavelength between the source and the destination of an IP link, as shown in Fig. 2.15 [5]. Fig. 2.16 shows a typical implementation of an IP-over-WDM network.

In IP-over-WDM networks, a single fiber usually carries multiple lightpaths and all of them get disconnected if the fiber carrying them fails. Even a single fiber failure can result in enormous loss of data, usually in the order of terabits per second. If the failure persists for a longer duration, the entire network may suffer severe service degradation. Therefore, adding survivability mechanisms to IP-over-WDM is crucial to ensure timely and uninterrupted delivery of information.

2.6.1 Survivable IP-over-WDM Networks

In IP-over-WDM networks, survivability can be provided at the physical layer or at the IP layer. At the physical layer, protection mechanisms discussed in section 2.3 can be used to make the network survivable. Protection can also be provided at the logical (IP) layer by allocating a backup lightpath for every primary (or working) lightpath such that a failure does not affect both the primary and the backup lightpaths [13]. The traffic on a primary lightpath is always routed on its corresponding backup lightpath after a failure. This may require investment in installation of additional fibers to provide diverse routes and additional capacity to accommodate all the primary and backup lightpaths. However, providing diverse routes is difficult and expensive due to geographical and right of way constraints. Hence, to make the network more resource efficient, a single backup lightpath may be shared by several primary lightpaths (which are not expected to fail simultaneously).

Providing restoration at the IP layer does not require dedicated backup lightpaths. Instead, the network can be provisioned with some additional capacity, and backup paths can be found for the failed lightpaths within this additional capacity at the time of failure. Restoration utilizes the available resources more efficiently and reduces the need for additional fibers to provide survivability to some extent [13].

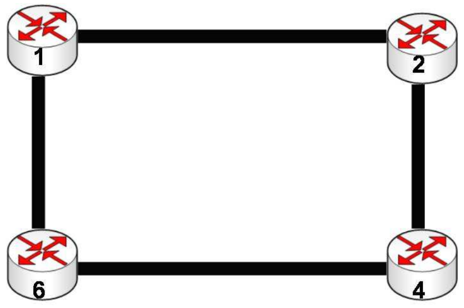
An IP-over-WDM network can be made survivable by either using protection or restoration mechanisms, however, it is not clear which would be a better choice [12] [13]. This report focuses only on one link survivable mechanisms that can be implemented at the IP layer due to their efficient resource utilization and low setup cost.

In IP-over-WDM networks, IP (logical) links are embedded in the physical WDM topology as lightpaths. When a fiber fails, all the lightpaths passing through this fiber are interrupted. The IP routers adjacent to the failed fiber can detect the failure and initiate the protocol to find alternate routes for the failed lightpaths. The routers can find alternate routes for all the affected lightpaths, only if the logical links are mapped in the physical topology in a way that the logical topology stays connected after a failure [1]. Otherwise, some of the IP routers may not be reachable.

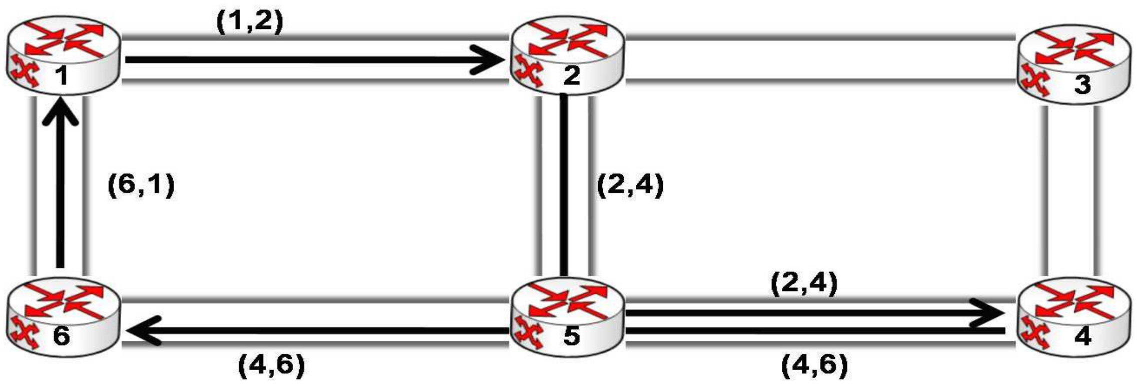
A necessary pre-condition for a network to be able to survive a link failure is that both physical and logical networks are at least 2-edge connected. A connected graph is 2-edge connected, if the removal of a link from the network does not disconnect it. If this condition is fulfilled, then the logical topology can be made survivable by requiring that some or all of the logical links are mapped in the physical topology in a *disjoint* manner (i.e. paths followed by the mapped links do not have a physical link in common) [1] [15]. This requirement ensures that a single physical link failure will not disconnect the logical topology. And since the logical topology is 2-edge connected, it is always possible for the IP routers to find an alternate path between the affected nodes as illustrated in the following example.

In Fig. 2.17, the logical topology consists of a ring $1-2-4-6$ as shown in Fig. 2.17(a). Fig. 2.17(b) shows a possible mapping of the logical topology in the physical topology. Here the logical links $(1,2),(2,4),(4,6)$ and $(6,1)$ are mapped to physical paths $1-2, 2-5-4, 4-5-6$ and $6-1$ respectively. However, this mapping is not survivable because if the physical link $(4,5)$ fails, then the logical links $(2,4)$ and $(4,6)$ fail, isolating node 4. Fig. 2.17(c) shows another possible mapping for the same logical and physical topologies. If the logical links $(1,2),(2,4),(4,6)$ and

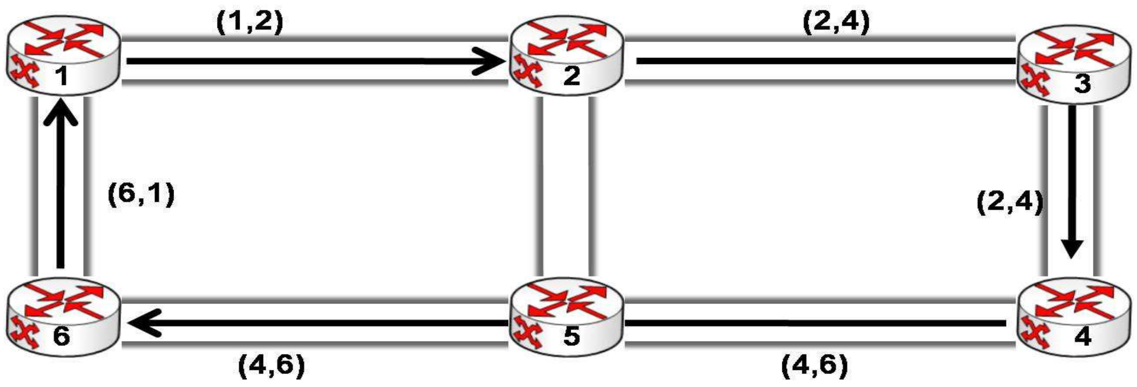
$(6,1)$ are mapped respectively to physical paths $1-2, 2-3-4, 4-5-6$ and $6-1$ then the routing is survivable, that is, any physical link failure does not disconnect the logical topology because the paths are link disjoint. For example, if the physical link $(2,3)$ fails then the logical link $(2,4)$ fails but the topology remains connected.



(a) Logical topology.



(b) An unsurvivable mapping.



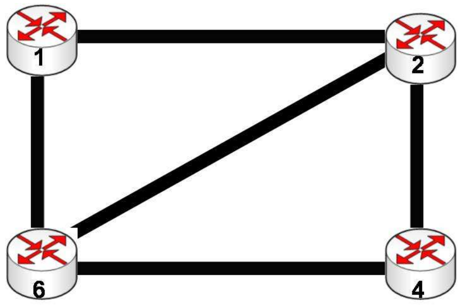
(c) A survivable mapping.

Figure 2.17: A ring topology and survivable mapping.

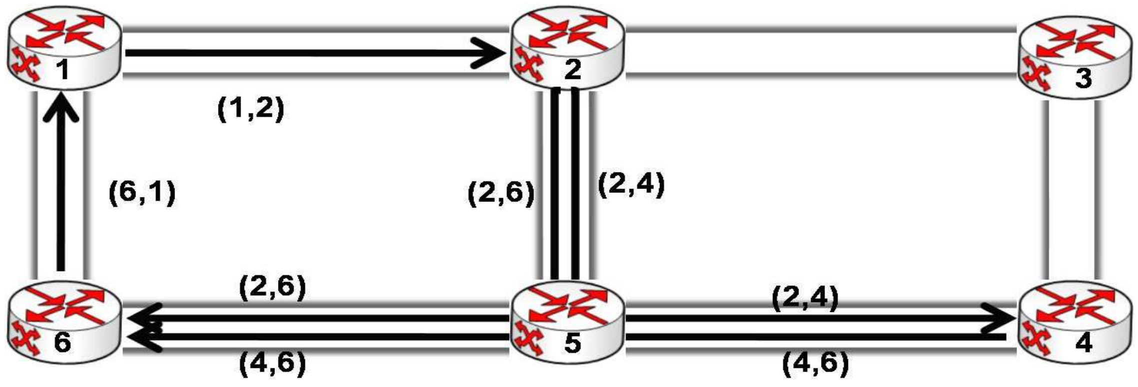
In Fig. 2.18(a), a more general logical topology is considered and Fig. 2.18(b) shows a possible mapping for the logical topology in the physical topology. Here the logical links $(1,2),(2,4),(4,6),(6,1)$ and $(2,6)$ are mapped to physical paths $1-2, 2-5-4, 4-5-6, 6-1$ and $2-5-6$ respectively. However, this mapping is not survivable because the physical link $(4,5)$ is being used by logical links $(2,4)$ and $(4,6)$ and if the physical link $(4,5)$ fails, then the logical links $(2,4)$ and $(4,6)$ fail, isolating node 4. Fig. 2.18(b) also shows that the physical link $(5,6)$ is common to the mapping of logical links $(4,6)$ and $(2,6)$. But it can be observed that when the physical link $(5,6)$ fails, logical links $(4,6)$ and $(2,6)$ fail but the logical topology still remains connected.

Fig. 2.18(c) also shows a possible mapping of the logical topology. Here logical links $(1,2),(2,4),(4,6),(6,1)$ and $(2,6)$ are mapped respectively to physical paths $1-2, 2-3-4, 4-5-6, 6-1$ and $2-5-6$ and it can be observed that, even though not all the paths are link disjoint, any single physical link failure does not disconnect the logical topology. For example, if the physical link $(5,6)$ fails then the logical links $(4,6)$ and $(2,6)$ fail but the topology remains connected.

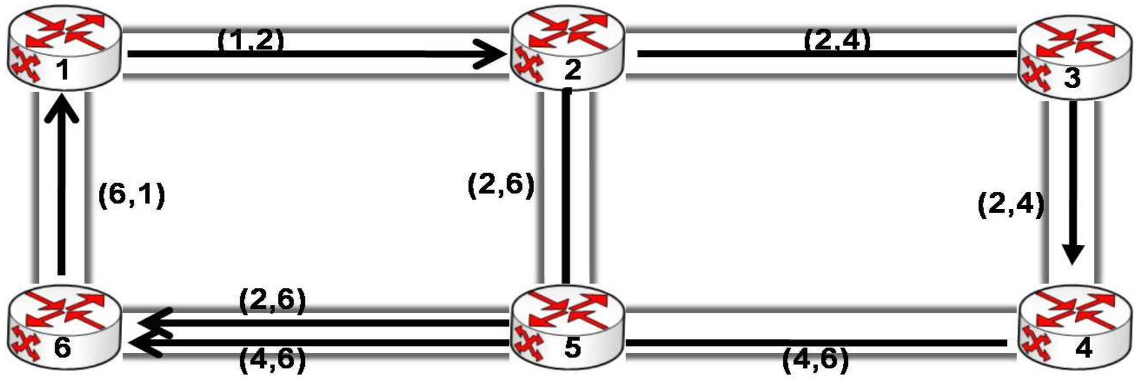
The above examples illustrate some of the difficulties involved in finding survivable mappings of logical topologies. If the logical topology is a cycle, then the problem is reduced to finding edge disjoint mappings (paths) for all the logical links in the physical topology, which is a well known NP-complete problem [1]. For general logical topologies, it may not be necessary to find edge disjoint mappings for all the logical links in the physical topology. Finding mappings for a subset of the logical links in a disjoint manner would be sufficient, but again the problem of finding disjoint paths in arbitrary topologies is NP-complete [1] [15]. Due to the NP-completeness of the problem, a large number of heuristics have been proposed in the literature.



(a) Logical topology.



(b) An unsurvivable mapping.



(c) A survivable mapping.

Figure 2.18: Mesh topology and survivable mapping.

2.7 Chapter Summary

In this chapter, we provided a brief overview of the different types of WDM networks and the survivability mechanisms employed in these networks. We introduced the concept of IP-over-WDM networks and survivable IP-over-WDM networks. We also discussed some of the issues involved in realizing such networks. In the next chapter, we will formally introduce the problem of designing survivable IP-over-WDM networks and provide a review of some of the common approaches proposed in the literature to design such networks.

Chapter 3

Survivable IP-over-WDM Networks Design: Problem Description and Literature Review

This chapter provides a formal description of the survivable IP-over-WDM network design problem and a review of various approaches proposed in literature.

3.1 Problem Description

Assume that we are given a 2-edge connected undirected physical topology $G(V, E)$ where V is the set of physical nodes (vertices), and E is the set of physical edges (links). A physical edge e is a pair of terminals, (i, j) such that $i, j \in V$. Each physical link $e \in E$ has an associated non-negative cost c_e, which represents the cost of using the link e. Also associated with each link $e \in E$ is a non-negative capacity u_e, which represents the maximum units of traffic e can carry.

The logical topology is defined by a set of source-destination (s, t) pairs where $s, t \in V$, such that together they form a 2-edge connected undirected logical topology $G_L(V_L, E_L)$. A source-destination pair (s_k, t_k) will also be referred to as a commodity

k. Let K be the set of all such (s, t) pairs i.e. $K = \{(s_1, t_1), (s_2, t_2), ..., (s_k, t_k)\}$. Also associated with each $(s_k, t_k) \in K$ is a non-negative demand d_k, which represents the units of traffic that must be sent from s_k to t_k in G without violating the capacities of all $e \in E$.

Both physical and logical topologies are assumed to be bi-directed i.e. if there is a link between node i and node j in the given topology then there is a link from node j to node i also.

A *mapping* M_k of a pair $(s_k, t_k) \in K$ is a path P_k that connects s_k to t_k in G. Let A be a set of source-destination pairs such that $A \subseteq K$, then M_A is a set of mappings for the source-destination pairs in A.

Given G and K, the survivable logical topology design problem then is to find mappings for some or all $(s_k, t_k) \in K$ $(1 \leq k \leq |K|)$ in G such that the removal of an edge $e \in E$ does not disconnect the logical topology G_L. Such a mapping is called *link-survivable*. If G_L is a ring, then the problem of finding link-survivable mappings is equivalent to finding disjoint paths for all the $(s_k, t_k) \in K$. However, in the case of a general topology, disjoint paths need to be found for only a subset of $(s_k, t_k) \in K$.

3.2 Literature Review

The problem of designing survivable IP-over-WDM networks has been widely studied in the literature. This section discusses some of the approaches presented in the literature.

3.2.1 Protection Interoperable Design

In [16], Crochat et. al. provide a general but comprehensive framework to the problem of finding survivable mappings, called *Protection Interoperable Design* (PID).

This paper also defines a *protected group with protection level k* as a group of logical links that can support up to k logical links failures. Such groups are assumed to have the ability to reroute traffic on broken links as long as the number of broken links is k or less. [16] suggests that a solution to PID should respect the following three constraints:

- *The capacity constraint*: A mapping should respect the capacity constraints of the physical links.
- *The bottleneck constraint*: One physical link/node failure should not cause more than k logical link failures.
- *The connectivity constraint*: The logical topology must remain connected for any single link/node failure in the physical topology.

The objective is to find a mapping for a logical topology in the physical topology, which satisfies the above constraints. Noting that the problem is NP-complete, [16] suggests relaxing some constraints to find a feasible solution e.g. capacity constraint can be relaxed to get a solution. It also provides an algorithm to solve the protection interoperability problem called the *Protection Interoperability for WDM* (PIW), based on *Taboo* search metaheuristic. PIW starts by finding a random mapping M for the logical topology and operates in iterations. In every iteration, the algorithm slightly modifies M by modifying lightpaths for some of the logical links. The process is repeated until a survivable mapping is found or a specified maximum number of iterations has been reached without any improvement in M.

3.2.2 Cutset Based Methods

In [1] and [15], Modiano and Narula-Tam establish conditions that are necessary and sufficient to design link survivable networks and formally show that the problem of finding survivable mappings in an undirected graph is NP-complete.

[1] and [15] provide an Integer Linear Program (ILP) to map arbitrary logical topologies in arbitrary physical topologies. The ILP is developed based on the observation that a single physical link failure can disconnect the logical topology only if it carries the entire cutset of the logical topology.

Given a partition $(S, V_L - S)$ of the node set V_L, the set of edges with one node in S and the other in $V_L - S$ is called a *cutset*. This cutset is denoted by $CS(S, V_L - S)$. In Fig. 3.1(a), the edges $\{a, e, g\}$ form a cut. Simultaneous removal of these edges partitions the graphs into two sets of nodes $\{1, 6\}$ and $\{2, 3, 4, 5\}$. Let M_a, M_e and M_g be the mappings of the edges $\{a, e, g\}$ in a physical topology. Now, if $\exists x \in E$ such that $(x \in M_a) \wedge (x \in M_e) \wedge (x \in M_g)$, then the removal of x would disconnect the graph. If there is no such $x \in E$, then the failure of a single link in the physical topology will not disconnect the graph. Fig. 3.1(b) provides another example of a cutset, here the edges associated with the cutsset are $\{a, g, h, i, d\}$.

Let $f_{ij}^k = 1$ if the mapping of the logical link k (a pair (s_k, t_k)) uses the physical link (i, j), otherwise $f_{ij}^k = 0$.

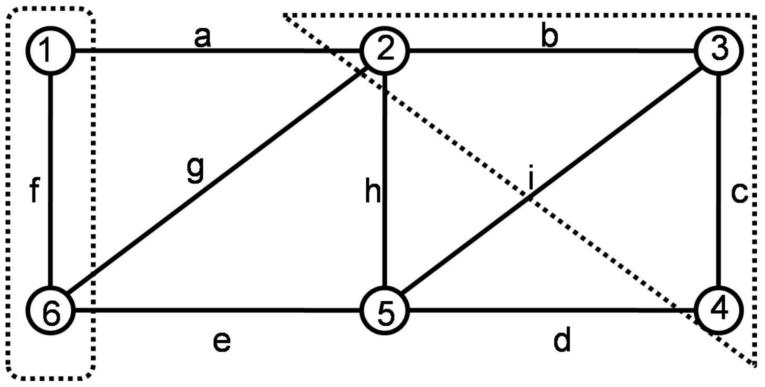
Following theorems are due to Modiano and Narula-Tam [1] [15].

Theorem 3.1. *A routing is survivable if and only if for every cut-set $CS(S, V_L - S)$ of the logical topology the following holds. Let $E(s, t)$ be the set of physical links used by the logical link (s, t), i.e., $E(s, t) = \{(i, j) \in E$ for which $f_{ij}^k = 1\}$. Then for every cut-set $CS(S, V_L - S)$,*

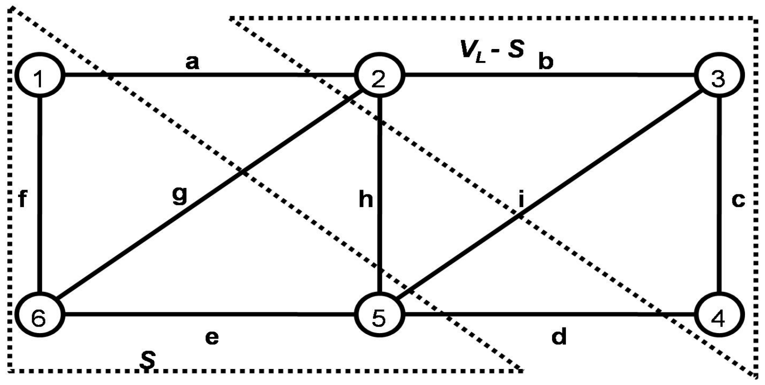
$$\bigcap_{(s,t) \in CS(S, V_L - S)} E(s, t) = \phi$$

Necessity: If there is a physical link $e \in E$ which is common to the routings of all the logical links belonging to a cut, then removal of such e will disconnect logical topology.

Sufficiency: To prove that the above condition is sufficient, it can be observed that the removal of a physical link e will leave at least one logical link in each cut



(a) $S = \{1, 6\}$, $V_L - S = \{2, 3, 4, 5\}$, and $CS(S, V_L - S) = \{a, g, e\}$.



(b) $S = \{1, 5, 6\}$, $V_L - S = \{2, 3, 4\}$, and $CS(S, V_L - S) = \{a, g, h, i, d\}$.

Figure 3.1: The Concept of a Cutset.

connected. Since a graph is connected if and only if every cut has at least one edge, it follows that the logical topology remains connected even if e fails. □

The following theorem has been proved in [1] and [15].

Theorem 3.2. *The problem of finding survivable mapping is NP-complete.* □

References [1] and [15] provide an Integer Linear Programming formulation based on Theorem 3.1. The formulation is shown in Fig. 3.2. If the formulation has a feasible solution, then a survivable mapping exists for the logical topology. And infeasibility implies that no such mapping exists.

References [1] and [15] also consider the special case where the logical topology is restricted to a ring. In this case, the capacity of each physical link is fixed at 1, since no two logical links can share a physical link. Now an attempt is made to find

$$Minimize \sum_{\substack{k \in K \\ (i,j) \in E}} f_{ij}^k$$

$$\begin{array}{lcr}
\text{Connectivity Constraint} & \displaystyle\sum_{j\ s.t.\ (i,j)\in E} f_{ij}^k - \sum_{j\ s.t.\ (i,j)\in E} f_{ji}^k = \begin{cases} 1 & \text{if } i = s_k \\ -1 & \text{if } i = t_k \\ 0 & \text{otherwise} \end{cases} & \forall\ i\ \in\ V,\ k\ \in\ K \\
\text{Survivability Constraint} & \displaystyle\sum_{k\in CS(S,V_L\setminus S)} (f_{ij}^k + f_{ji}^k)\ <\ |CS(S, V_L \setminus S)| & \forall (i,j) \in E,\ k \in K,\ \forall S \subset V_L \\
\text{Capacity Constraint} & \displaystyle\left(\sum_k f_{ij}^k + \sum_k f_{ji}^k\right) \le (u_{ij} + u_{ji}) & \forall (i,j) \in E,\ k \in K \\
\text{Integer Flow Constraint} & f_{ij}^k \in \{0,1\} & \forall (i,j) \in E,\ k \in K
\end{array}$$

Figure 3.2: ILP to find survivable mappings for arbitrary logical topologies.

$$Minimize \sum_{\substack{k \in K \\ (i,j) \in E}} f_{ij}^k$$

$$\begin{array}{lcr}
\text{Connectivity Constraint} & \displaystyle\sum_{j\ s.t.\ (i,j)\in E} f_{ij}^k - \sum_{j\ s.t.\ (i,j)\in E} f_{ji}^k = \begin{cases} 1 & \text{if } i = s_k \\ -1 & \text{if } i = t_k \\ 0 & \text{otherwise} \end{cases} & \forall\ i\ \in\ V,\ k\ \in\ K \\
\text{Capacity Constraint} & \displaystyle\left(\sum_k f_{ij}^k + \sum_k f_{ji}^k\right) \le 1 & \forall (i,j) \in E,\ k \in K \\
\text{Survivability Constraint} & f_{ij}^k \in \{0,1\} & \forall (i,j) \in E,\ k \in K
\end{array}$$

Figure 3.3: ILP to find survivable mappings for ring logical topologies.

paths for the logical links of the ring. If a feasible integer solution is obtained, then a survivable mapping for the ring has been found. Otherwise, survivable mapping of the ring in the physical topology is not possible. The ILP formulation is shown in Fig. 3.3.

The ILP formulation in Fig. 3.2 and 3.3 can be solved exactly by using commercially available software packages (such as CPLEX), but due to the large number of constraints involved, the running time to solve the ILP may be excessive.

Also, the formulations in Fig. 3.2 and 3.3 require examination of all the cutsets in the logical topology, which is a very large set. Therefore, [1] and [15] also explore some relaxations to the ILP formulation. One proposed relaxation is to consider only those cuts that prevent a single node from being isolated, in case of a single fiber failure. Another proposed relaxation is to consider cuts with size equal to or less than the degree of the topology plus one. These relaxations significantly reduce the time required to solve the problem but may lead to infeasible solutions.

In [17], Todimala and Ramamurthy apply the work in [1] and [15] to *Shared Risk Link Groups* (SRLGs). SRLG is a set of links that share the risk of failing simultaneously. [17] proves that it is sufficient to consider only primary cuts in the ILP to design survivable networks. A *primary cut* is defined as a cut that partitions the graph into two or more components such that the subgraph on the nodes in each component is connected. Otherwise, it is called a *secondary cut*. The substitution of all the cutsets with primary cuts in the ILP is based on the fact that every secondary cut is a superset of at least one primary cut. Fig. 3.1 provides examples of primary cuts and Fig. 3.4 is an example of secondary cut.

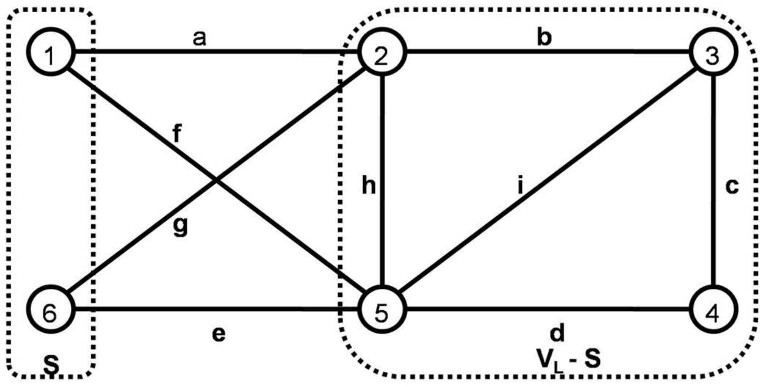


Figure 3.4: A secondary cut $S = \{1, 6\}$ and $V_L - S = \{2, 3, 4, 5\}$.

An undirected connected graph G has 2^{n-1} possible cuts, which is exponential in n, the number of nodes in G_L. The number of primary cuts in G_L is less than the number of cuts, but still exponential in number. Therefore, [17] applies the ILP to *planar cycles* and *hierarchical planar cycles* that have polynomial number of primary cuts. A planar cycle is defined as a planar cyclic graph with chords that do not cross and a hierarchical planar cycle is a collection of planar cycles such that a pair of planar cycles has at most one node in common. [17] also proves that the number of primary cuts in a planar cycle is equal to the number of primary cuts in a simple cycle of the same size.

References [18] and [19] propose two heuristics, named *FastSurv* and *extended FastSurv*, based on Theorem 3.1. FastSurv does not consider capacity constraints and extended FastSurv does. However, the heuristics do not directly use cutsets but use a probability function as an estimate of the cutsets.

The algorithms start with an initial mapping, which is constructed by arbitrarily finding lightpaths for the logical links in a random order using a shortest path algorithm. The cost function used for the shortest path algorithm depends upon the number of lightpaths currently being carried by a physical link. After obtaining the initial solution, it is tested for survivability. If the mapping is not survivable, then all the lightpaths which when routed together make the mapping unsurvivable are rerouted. The rerouting is done based on an estimate of the probability that two lightpaths when routed together make the mapping unsurvivable and using this as the cost function.

In [20], Ruan and Liu propose a heuristic *Map and Fix*, which like [17], [18] and [19] utilizes Theorem 3.1 and tries to find a mapping for a logical topology that is survivable or close to survivable. If the mapping is not survivable then new mappings are calculated for some of the logical links. However, the algorithm does not take into account capacity constraints.

The algorithm has three-stages. In the first stage, called *Simple-Mapping* (SM), an initial mapping is computed by finding a shortest path between the end nodes of each logical link in the physical topology. If the initial mapping is not survivable then a new mapping is computed in the next stage by using *Load-Based-Mapping* (LBM). In LBM each physical link e is assigned a cost equal to one plus the number of mappings using e. New mapping for the logical topology is then computed by applying Dijkstra's algorithm [21] using this new cost. If the new mapping is not survivable then FIX algorithm is applied.

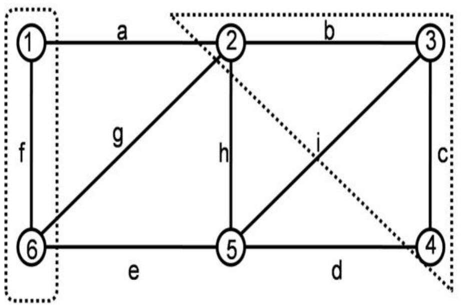
FIX algorithm accepts an unsurvivable mapping M and remaps some of the logical links to obtain a new mapping M'. FIX first finds physical links, called critical links (M'), which carry an entire cut of the logical topology. Let $e \in E$ be a physical link that carries an entire cut CS ($CS \subseteq E_L$) of the logical topology. Failure of e would disconnect the logical topology and transform it into m ($m \geq 2$) components ($C_1, C_2, ..., C_m$). An edge $b \in CS$ is called a *bridge* as its end nodes lie in two different components. The FIX algorithm chooses at most one bridge from the set of bridges that connect a pair of components and remaps it without using e. Each time a bridge is remapped, the number of components is reduced by 1. After remapping $m - 1$ bridges, the new mapping M' is tested for survivability. If M' is not survivable, then a new set of bridge links are chosen and remapped. The process is repeated until a survivable mapping or the maximum number of iterations is achieved. The maximum number of iterations is set to $K \times |M'|$, where K is a constant.

3.2.3 Circuit Based Methods

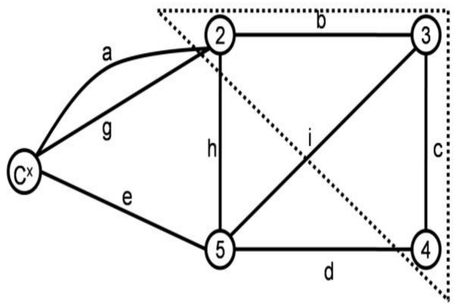
In [2] [22] and [23], Kurant and Thiran provide a unique framework to find node/edge survivable mapping for a general logical topology by successively mapping only a subset of the logical links. It is accomplished by introducing the concept of *piecewise survivability*, which proceeds by successively finding survivable mappings for *pieces* of the logical topology rather than for the entire logical topology.

The concepts of *contraction of an edge* in a graph G_L and *Origin*(.) function are key to understanding the framework. Let x be an edge such that $x \in E_L$. The process of contraction involves deleting the edge x and merging its end nodes. In

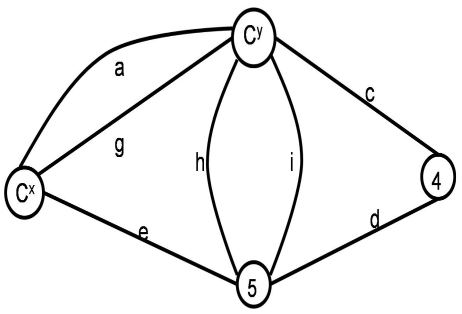
Fig. 3.5(a), the edge to be contracted is f with end nodes 1 and 6. The contraction is done by deleting the edge f and merging the end nodes 1 and 6 to form a single node C^x. The edges which were incident on nodes 1 and 6 are now incident on the contracted node C^x and self loops are deleted. Fig. 3.5(b) shows the graph after contraction of edge f. The concept of contraction of an edge is then extended to the contraction of a set of edges $Y \subseteq E_L$. This is accomplished by successively contracting the edges belonging to Y in the graph G_L. In Fig. 3.5(a), $Y = \{b, c\}$, Fig. 3.5(c) shows G_L after the contraction of the edge b and Fig. 3.5(d) shows G_L after contraction of edges b and c.



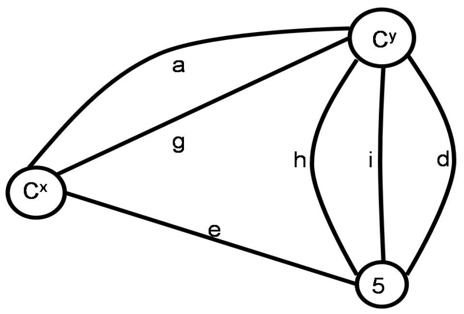
(a) A graph with two subgraphs $\{f\}$ and $\{b, c\}$.



(b) Graph after contraction of edge f.



(c) Graph after contraction of edge b.



(d) Graph after contraction of edge c.

Figure 3.5: The concept of contraction of an edge and a subgraph.

A link e in contracted graph G^C is always a link in G_L. But a node in G^C may or may not be a node in G_L. In Fig. 3.4(b) node labeled 5 is a node in the G_L. However, the node C^x "*originates*" from a connected component of G_L consisting of nodes 1 and 6 and is not in G_L. To formally define *Origin*(.), let A

be a set of edges belonging to a graph G_L and let G^C be the new graph obtained after contracting the edges in A. Now, pick a subgraph G^C_{sub} in G^C. $Origin(G^C_{sub})$ is the maximal subgraph of G that transforms into G^C_{sub} by the contraction of A in G. The framework assumes that the given logical topology G_L and physical topology G have some degree of redundancy built into them. [2] proves the following Theorem.

Theorem 3.3. *(Expansion of survivability): Let M_A be a mapping of a set of logical edges $A \subset E_L$ in the physical topology G, such that the pair $[G_L, M_A]$ is piecewise node/link-survivable. Let G^C be the graph obtained after contracting A in G_L. Take any subgraph of G^C, call it $G^C_{sub} = (E^C_{sub}, B)$. Let M_B be a mapping of the set B of edges of G^C_{sub} on G. If the pair $[G^C_{sub}, B]$ is node/edge-survivable then the pair $[Origin(G^C_{sub}, M_A \cup M_B)]$ is also node/edge-survivable.* $\square$

Theorem 3.3 is based on the following property. If a subgraph G_1 is routable in a survivable manner in the physical topology and a subgraph G_2 in the contracted subgraph is also routable in a survivable manner, then the subgraph of the logical topology containing the links of G_1 and G_2 is also routable in a survivable manner. If true, this approach greatly simplifies the process of finding survivable mappings for fairly large networks. The solution is applicable to arbitrary logical and physical topologies and can be applied to find link survivable mapping as well as to find node survivable mappings. However, we prove in [24] that the above claim is true only if the selected subgraph meets some connectivity requirements.

Based on Theorem 3.3, [2] provides an algorithm which is called SMART (Survivable Mapping Algorithm by Ring Trimming). The algorithm starts with the entire logical topology, and then tries to find a subgraph for which a node/edge-survivable mapping can be found in the physical topology. If such a subgraph exists, the algorithm *contracts* this subgraph by collapsing the edges in the subgraph to create a contracted logical topology. See Fig. 3.5 for an example of the process of contracting a subgraph. After contracting the subgraph, the algorithm proceeds by finding another subgraph that can be mapped in a survivable manner. The algorithm terminates, if at any step there is no such subgraph in the contracted logical topology that can be mapped in survivable manner or the logical topology is reduced to a single node. If the algorithm terminates unsuccessfully, it returns a *piecewise node/edge-survivable mapping*, which consists of survivable *pieces* and the remaining contracted logical topology. The algorithm is shown in Fig. 3.6.

```
INPUT: A physical topology $G(V, E)$, a logical topology $G_L(V_L, E_L)$.
OUTPUT: A survivable mapping or a piecewise survivable mapping of $G_L$.
Step 1: Start from the full logical topology, $G_C = G_L$, and an empty mapping $M_A = \emptyset$, $A = \emptyset$.
Step 2: Pick a subgraph $G^C_{sub} = (V^C_{sub}, B)$ in $G^C$, and find a mapping $M_B$ for the links in the subgraph such that the pair $[G^C_{sub}, M_B]$ is node/edge-survivable.
   If no such (subgraph, mapping) pair is found then
      Return the mappings that have been obtained so far and the
      remaining Contracted Topology.
    END
   End If
Step 3: Update the mapping by merging $M_A$ and $M_B$ i.e. $M_A = M_A \cup M_B$.
Step 4: Contract $G^C$ by collapsing the edges included in the subgraph $B$ and merging the nodes to create a single node.
Step 5:
   If the contracted logical topology $G^C$ is reduced to a single node then
      RETURN $M_A$.
      END
  End If
Step 6: GOTO Step 2, REPEAT THE STEPS FOR THE CONTRACTED TOPOLOGY ($G^C$).
```

Figure 3.6: Survivable Mapping Algorithm by Ring Trimming (SMART) algorithm.

SMART framework, however, does not provide a method for finding survivable mapping for a subgraph in the physical topology for the general case. But [2] does provide a simple heuristic to find mappings for logical links when the subgraph is always a cycle (ring). The heuristic uses a variant of shortest path algorithm, therefore does a local search. The algorithm begins by assigning a weight of 1 to each physical link and finds a shortest path for each logical link in the selected cycle. If the paths obtained are edge disjoint, the algorithm proceeds by contracting the subgraph under consideration and picking a new subgraph. If some or all the paths are not edge disjoint, the weights assigned to the shared physical edges are incremented by 1 and the process is repeated for a predetermined number of times. In a more recent work by authors of [2], the framework is named SMART and the proposed heuristic is referred to as SMART-H [23]. [23] also considers a version of SMART-H with capacities. Henceforth, we will refer to the SMART framework as SMART and the heuristic as SMART-H.

Most approaches in literature consider the entire logical topology while designing survivable networks. [2], [22] and [23] provide a different approach by considering only pieces of the logical topology. Nonetheless, it has certain drawbacks. The next chapter contains a detailed analysis of the SMART and SMART-H and some of enhancements possible.

3.2.4 Multicommodity Flows Approach

The problem of finding edge disjoint paths bears a very close resemblance to the *multicommodity flow problem* (MCF) in operations research [25]. Given a set of source-destination pairs and corresponding demands, the MCF involves simultaneously routing the different commodities from their sources to their respective destinations, such that the total flow on each edge is not greater than its capacity while meeting certain objectives. There are several variants of the MCF, namely, *maximum multicommodity flow* (MMCF), *maximum concurrent flow* and *minimum cost multicommodity flow* problems.

The aim of MMCF is to maximize the total flow, summed over all the commodities, with each link flow equal to or less than its capacity without any fixed demands. In maximum concurrent flow problem each commodity has an associated non-negative demand. The goal then is to satisfy the maximum possible proportion of demand for each commodity. In case of minimum cost multicommodity flow problem, each edge is assigned a non-negative cost and a demand is associated with each commodity. The objective is to find a multicommodity minimum cost flow.

MMCF formulation can be used to find pair-wise disjoint paths by setting the link capacities to 1 and requiring flows to be integers only. In case of a ring, a survivable mapping of a ring logical topology in an arbitrary physical topology is possible only if a feasible solution to its integer MMCF formulation exists. For arbitrary graphs, a feasible solution to the Integer MMCF implies that a survivable mapping exists but an infeasible solution does not imply that such a mapping does not exist. This is due to the fact that, as we have shown in section 2.6.1, it is enough to find disjoint paths for some rather than all the logical edges. MMCF can be formulated as an ILP (similar to Fig. 3.3) and solved optimally. However, integer version of MMCF is NP-complete.

3.2.5 Other Approaches

Reference [26] does not directly address the problem of finding survivable mappings but provides a solution to the problem of finding disjoint paths. In the case of ring networks, this algorithm can be directly employed to find survivable mappings. But it can be easily extended to the case of general mesh networks by checking for survivability rather than for disjoint paths. Given a set of source-destination pairs $K = \{(s_1, t_1), (s_2, t_2), ..., (s_k, t_k)\}$ the objective is to determine disjoints paths among as many source-destination pairs as possible. The same problem is considered in [27] and an iterative greedy algorithm called *Bounded Greedy Algorithm* (BGA) is provided. BGA starts off with an empty solution S, picks a pair $(s_i, t_i) \in K$ and routes it using a shortest path P of length L or less. After routing, the edges along P are removed from the graph and the algorithm proceeds by picking the next pair (s_{i+1}, t_{i+1}) from K. An extension of BGA is also proposed in [27] called *Multi-start Greedy Algorithm* (MSGA). In an iteration of MSGA, S_i is the solution under consideration and S_{best} is the best solution achieved so far. Also, in each iteration of MSGA, a random permutation of K is used.

The main contribution of [27], however, is the development of an algorithm based on *Ant Colony Optimization* (ACO). A solution S is constructed by assigning an *ant* to each (s_i, t_i) pair, which finds a path for the assigned (s_i, t_i) pair. Initially the paths in S may not be mutually disjoint. Edge disjoint paths are then obtained by iteratively removing the path which has the most edges in common with other paths in S. The algorithm continues until edge disjoint paths are found or the termination condition is met.

3.3 Chapter Summary

In this chapter, we formally introduced the problem of designing survivable IP-over-WDM networks and discussed some of the commonly proposed approaches to solve the problem. This chapter discussed two approaches in detail, namely, the approaches by Kurant-Thiran and Modiano-Narula-Tam, which will form the basis for our work.

In the next chapter, we will provide a detailed analysis of SMART framework and its heuristic version (SMART-H). We will also discuss some of the enhancements that can improve the performance of original algorithms.

Chapter 4

SMART: Evaluation and Enhancements

SMART (*Survivable Mapping Algorithm by Ring Trimming*) framework, introduced in Chapter 3, is a unique approach to find survivable mappings for logical topologies. Most approaches in the literature usually consider the entire logical topology and attempt to find a survivable mapping for it. In contrast, SMART proceeds by picking pieces of the logical topology and finding survivable mappings for these pieces. This chapter discusses some of the shortcomings of SMART framework and proposes some enhancements to improve the performance of SMART.

4.1 SMART Evaluation

The step 2 of SMART approach (Fig. 3.6) requires selecting a subgraph which is connected. However, as shown below with an example, that this requirement is not sufficient to guarantee survivability.

Consider the logical and physical topology shown in Fig. 4.1(a) and Fig 4.1(b), respectively. Assume that the step 2 of the algorithm first selects a connected component consisting of logical links $\{a, f\}$, which are mapped to paths $1-2-3-4$ and

$1-6-5$ in the physical topology (shown in Fig 4.1(d)). It can be observed that the mapping for this subgraph is survivable. Therefore, the subgraph is contracted into a single node to obtain contracted topology shown in Fig. 4.1(e). The algorithm then proceeds by selecting another subgraph consisting of edges $\{b, g, e\}$. This subgraph can be mapped to paths $4-3-2, 2-6$ and $6-5$ (Fig. 4.1(f)), which are edge disjoint and form a survivable mapping. b, g and e are then contracted to obtain contracted topology shown in Fig. 4.1(g) and a new subgraph with edges $\{c, d\}$ is chosen, which is mapped to paths $3-2$ and $3-6$ (Fig. 4.1(h)). Contraction of subgraph c and d reduces the logical topology to a single node. Therefore, SMART terminates and returns the mapping shown in Fig. 4.1(i) as the survivable mapping for the logical topology.

The mapping shown in Fig. 4.1(i), however, is not survivable. It can be seen that the failure of physical link $(5, 6)$ brings down logical links $\{e, f\}$ and logical topology gets disconnected, isolating logical node 5. Similarly, if the physical link $(2, 3)$ fails, the logical links $\{a, b, c\}$ fail and logical node 4 is isolated.

The above example stresses that it is important to pick the subgraph in step 2 carefully. We now prove the following which is the correction to Theorem 3.3.

Theorem 4.1. *A 2-edge connected logical graph is survivable if and only if SMART algorithm picks in step 2 a 2-edge connected subgraph and terminates successfully.*

Proof:

Necessity: *SMART terminates successfully* $\longrightarrow$ *Graph is survivable.* This follows from Theorem 3.3.

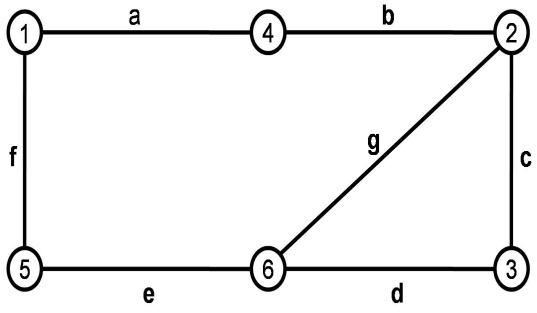
Sufficiency: *SMART terminates unsuccessfully* $\longrightarrow$ *Graph is not survivable.* Proof is by contradiction. Assume that the SMART algorithm terminates unsuccessfully and the logical topology G_L is survivable. Thus there exists a mapping of the links of G_L that guarantees survivability of G_L for any single link failure. But at some iteration, step 2 did not find a survivable 2-edge connected subgraph of the contracted graph G_C. Since G_L is survivable removing a physical link will result in G'_L that is connected. If we now contract those edges in G'_L corresponding to the edges defined by the contracted nodes in G_C then the resulting graph G'_C will be connected. This means that removal of a physical link does not disconnect G_C. So G_C is survivable and SMART would terminate successfully at this step. Hence, a contradiction. □

Note that the above theorem holds even if the subgraph chosen in step 2 of SMART is not connected as long as every connected component of the selected subgraph is 2-edge connected. The algorithm can be easily extended to k link failures by requiring logical and physical topologies to be at least $(k + 1) -$ *connected* and picking subgraphs that are at least $(k + 1) -$ *connected* in step 2 of SMART.

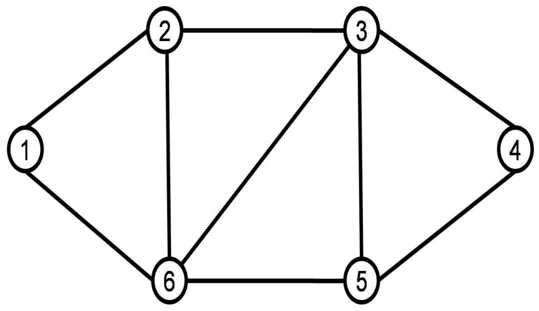
The step 2 of the algorithm also requires finding a mapping for the selected 2-edge connected subgraph that is survivable. Since a survivable mapping requires that at least some of the logical links are routed on disjoint paths in the physical topology, a problem that is NP-complete in general [26], a heuristic must be employed.

The algorithm shown in Fig. 3.6 terminates, when in step 2, it is not possible to find any subgraph in the contracted logical topology that could be mapped in survivable manner. This would force the algorithm to examine a very large number of subgraphs before termination, especially if a survivable mapping of logical topology in the physical topology does not exist. If the subgraph is restricted to a cycle in each iteration, then an undirected complete graph has $\frac{1}{2}\sum_{i=3}^{|V|}\binom{|V|}{i}(i-1)!$ cycles, a number which grows faster with $|V|$ than the exponential $2^{|V|}$ [28]. This is a computationally expensive task. Therefore, any implementation of SMART will be limited to considering a small number of subgraphs, and may terminate unsuccessfully, if none of these selected subgraphs is found to be survivable.

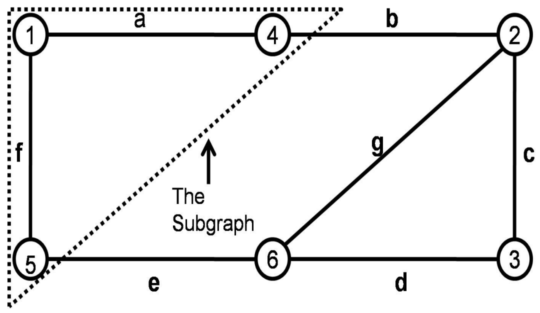
Given the above drawbacks of SMART framework, a heuristic version of SMART called SMART-H is given in [2] [23]. SMART-H considers only a small number of



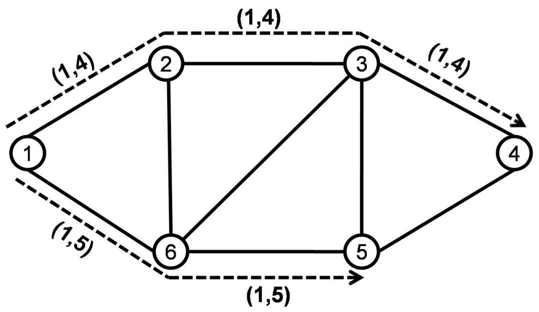
(a) A logical topology.



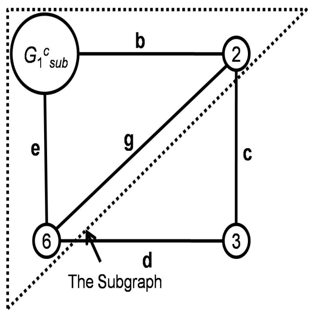
(b) A physical topology.



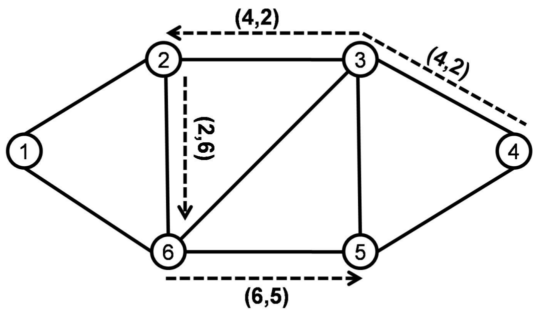
(c) The selected subgraph consists of logical edges $\{a, f\}$.



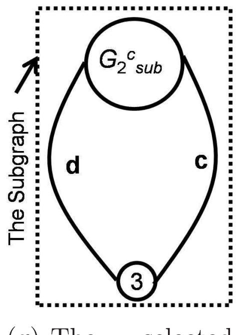
(d) A mapping of edges $\{a, f\}$ in the physical topology.



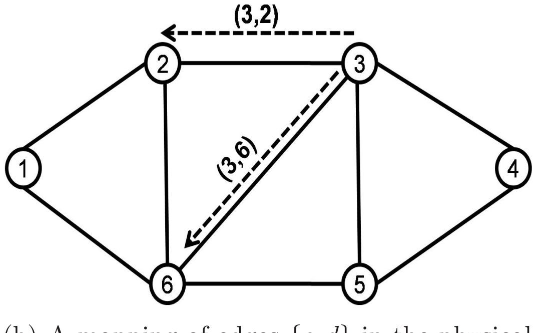
(e) The selected subgraph consists of logical edges $\{b, e, g\}$.



(f) A mapping of edges $\{b, e, g\}$ in the physical topology.

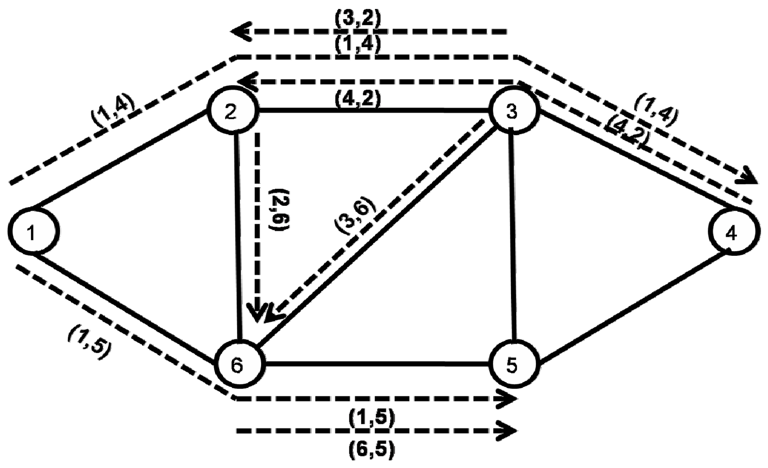


(g) The selected subgraph consists of logical edges $\{c, d\}$.



(h) A mapping of edges $\{c, d\}$ in the physical topology.

Figure 4.1: The effect of subgraph selection on survivability.



(i) Complete mapping of the logical topology in the physical topology.

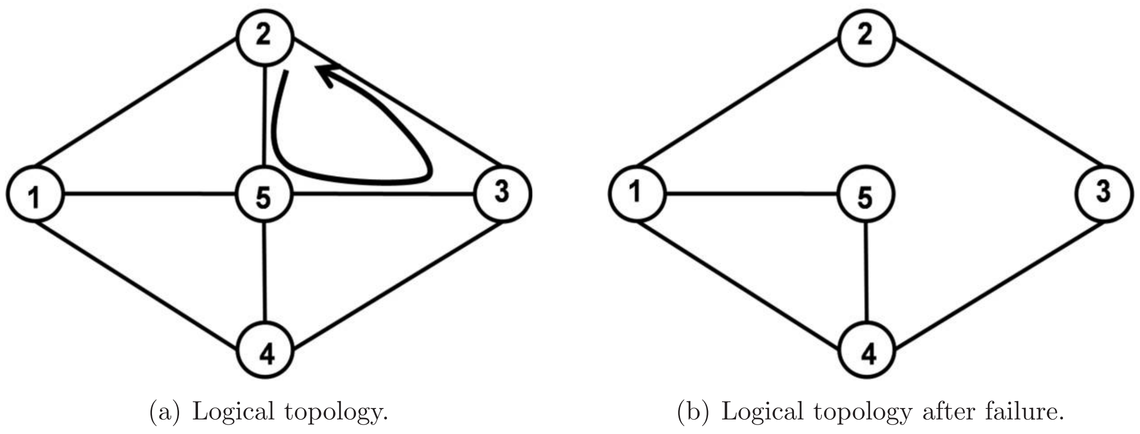
Figure 4.1: The effect of subgraph selection on survivability. (contd.)

subgraphs before termination. However, this does not mean that survivable mapping of the entire logical topology does not exist if survivable mappings for all the considered subgraphs are not possible. As an example consider Fig. 4.2. Assume that all the links in the subgraph $2-3-5$ cannot be mapped in a disjoint fashion. Also assume that links $(2,5)$ and $(3,5)$ share a physical link. However, if we are able to map all the remaining links in a disjoint manner then the logical topology will still remain connected. This leads us to the following:

Fact 1: If a survivable mapping cannot be found for a particular subgraph then accepting an unsurvivable mapping for this subgraph and continuing with SMART-H may still yield a survivable mapping for the entire logical topology.

The requirement that a 2-edge connected graph must be selected in step 2 can easily be met by choosing a cycle (or ring), which can be easily picked using a shortest path or breadth first search algorithm. The question then is whether we should select a longer or shorter cycle. A short cycle can be picked using a shortest path algorithm

Fact 2: A shorter cycle has an advantage over a longer cycle in the sense that the chances of successfully mapping a small number of links is higher than mapping



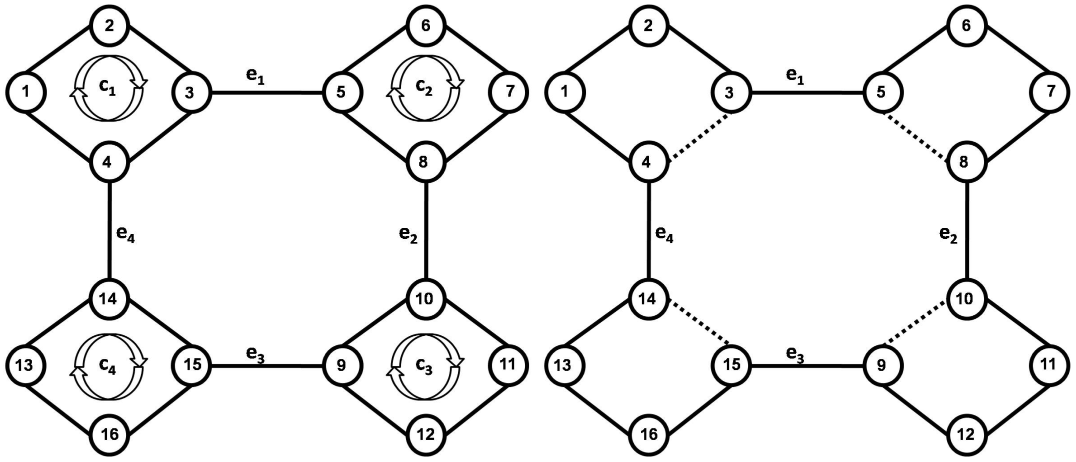
(a) Logical topology. (b) Logical topology after failure.

Figure 4.2: Illustration of Fact 1.

a larger number of links. However, a longer cycle has the advantage of having one or more number of straddling links that can be mapped in an arbitrary manner.

Picking shorter cycles provides an additional advantage. When the SMART-H algorithm terminates successfully, it gives a mapping which remains connected for any single physical link failure. This is true even if a single physical link failure causes more than one logical link to fail as long as these logical links are in different cycles constructed by the SMART-H algorithm. Therefore, picking smaller cycles to map at each step of SMART-H can keep the logical topology connected even if a physical link failure results in a large number of logical link failures. This leads us to definition of the concept of *robustness of a routing*.

A survivable routing is considered *robust* if the logical topology remains connected for a large number of pairs of logical link failures. So, the larger the number of cycles considered by SMART-H the higher will be the robustness of the routing. For example, consider the logical topology in Fig. 4.3(a). Consider cycles in the following order: c_1, c_2, c_3, c_4, and cycle with edges e_1, e_2, e_3, e_4 then the resulting survivable mapping will remain connected for 160 pairs of logical link failures. Also it can survive 45 5-link failures. If SMART-H picks the cycle $\{1, 2, 3, e_1, 5, 6, 7, 8, e_2, 10, 11, 12, 9, e_3, 15, 16, 13, 14, e_4, 4, 1\}$ then the resulting mapping will survive only 16 1-logical link failures (Fig. 4.3(b)).



(a) Shorter cycles. (b) Longer cycle (bold lines).

Figure 4.3: Robustness of a routing.

Fact 3: Always selecting a short cycle at step 2 of Fig. 3.6 makes the network more robust.

SMART-H uses a variant of shortest path algorithm to find a survivable mapping for subgraph and therefore performs a local search. Each time the algorithm fails to find a disjoint mapping, it increments the weights of physical links used more than once by 1. Increments of 1 may require a large number of applications of shortest path algorithm before disjoint mappings are found. As an example, consider the topology shown in Fig. 4.3 (a) as the physical topology. Assume that SMART-H is trying to find a mapping between logical nodes 5 and 7, and the discovery of path $5-3-4-14-15-9-10-8-7$ (length 8) is required so that the mapping for a given subgraph is survivable. The algorithm will alternate between paths $5-6-7$ and $5-8-7$ several times before choosing the correct path. However, penalizing the repeated links heavily, may force the algorithm to converge faster.

Fact 4: To find survivable mapping for a subgraph using a cost function that increments the cost of links, which appear in the mapping more than once, at a higher rate can increase the probability of finding a survivable mapping.

4.2 SMART Enhancements

In light of the above mentioned facts, we propose following enhancements to SMART-H heuristic.

4.2.1 Enhancement 1:

To improve robustness, when selecting a subgraph effort should be made to not include contracted nodes as long as possible.

As pointed out in *fact* 1, even if a survivable mapping cannot be found for a particular subgraph then accepting this mapping and continuing with SMART-H may still yield a survivable mapping for the entire logical topology.

4.2.2 Enhancement 2:

If a survivable mapping cannot be found for any of the subgraphs considered then accept the unsurvivable mapping for the current subgraph under consideration and continue with SMART-H.

At the end of the SMART-H algorithm some of the logical links may be left unmapped. In fact, these logical links can be mapped in an arbitrary manner but by a judicious mapping of these links one can achieve a high success rate for obtaining survivable mappings. This is similar to taking advantage of DON'T CARE conditions in logical function minimization. Furthermore, if a subgraph contains a contracted node, then disjoint mappings are calculated for only two of the multiple links incident on it. The remaining links are not mapped. However, even if these links are mapped in an arbitrary manner (or disjoint manner, if possible) it may allow some of the logical topologies that were previously declared unsurvivable to become survivable. So we propose the following enhancement to SMART-H.

4.2.3 Enhancement 3:

At the end of SMART-H, map unmapped links in an appropriate manner to reduce the degree of unsurvivability of the final mapping.

The mapping algorithm used by SMART-H can also be modified to use a new cost function. As an alternative to incrementing by 1, each physical link can be assigned a cost of 2^{α} where α is the number of times a link appears in unsuccessful attempts. Initially, $\alpha = 0$, however in the next iteration α is incremented by 1 and the links used more than once have cost 2. Similarly, in the next iteration if any of the links repeated in previous iteration appear again their cost is incremented to 4.

4.2.4 Enhancement 4:

Before finding shortest paths, assign each physical link a cost of 2^{α}, where α is the number of times the link appears in unsuccessful attempts.

4.3 Simulation Study of SMART-H and Enhancements to SMART-H

To study the behavior of SMART-H and the proposed modifications, they were implemented using LEDA [29] and VC++ 7.0. For simulation two kinds of physical topologies with a varying number of nodes and edges were generated. Regular topologies of n nodes were generated with an average degree of 6. The regular topologies were constructed using a procedure originally given by Harary [30] and described in [31], and random topologies were generated with LEDA using a procedure given in [32] with n nodes and $m = 3 \times n$ edges. The values of n were $\{20, 50\}$. Logical topologies were generated randomly with number of nodes being $f \times n$, where $f = \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and the number of edges were $1.5 \times (f \times n)$.

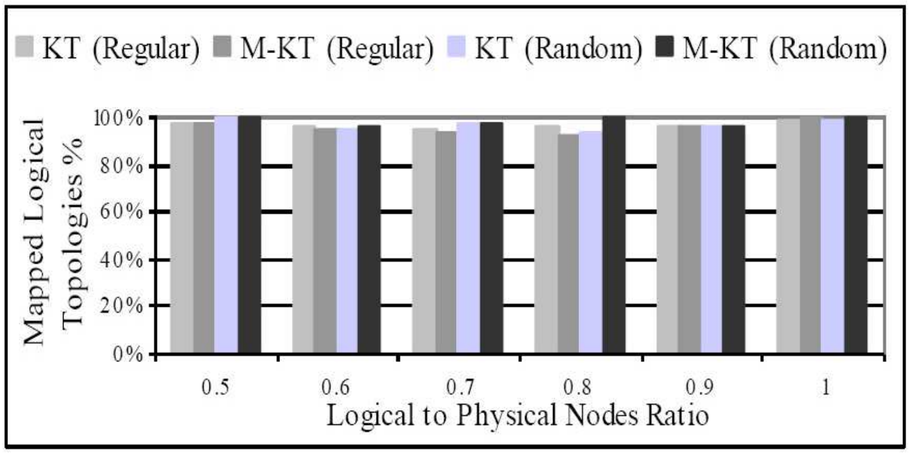
The logical nodes were always a random subset of the physical nodes. For each f, 100 random logical topologies were generated. After generating the topologies, they were admitted for further processing only if they were at least 2-edge connected. Each logical-physical topology pair was subjected to following two methods:

Method 1(KT): The first method was an implementation of SMART-H. A subgraph was chosen randomly and then attempts were made to map it in disjoint manner. If a disjoint mapping could not be obtained after applying the mapping method 50 times, a new subgraph was selected and SMART-H terminated after unsuccessfully examining 25 subgraphs.

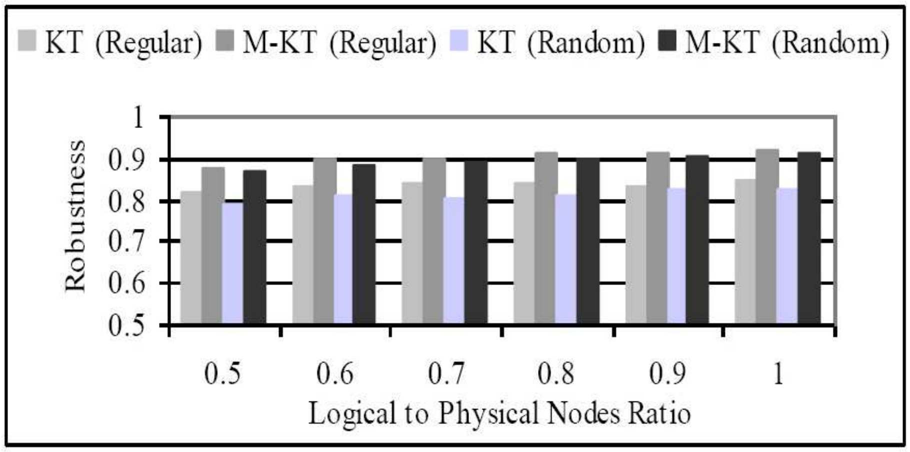
Method 2(M-KT): This method implemented SMART-H approach with enhancements 1, 2, 3, and 4; a) an attempt was made to find a subgraph without a contracted node; b) an unsurvivable mapping for a subgraph was accepted; c) all the links not previously mapped, were mapped; d) the cost function described in enhancement 4 was used. e) At the end of the execution, a test was conducted to see if the logical topology remained connected in case of a single physical link failure.

The statistics of interest were the percentage of successfully (survivable) mapped logical topologies and robustness. Methods 1 and 2 were applied on the same logical-physical pairs, and tests were conducted with $n = \{20, 50\}$. Fig. 4.4 (a) and (b) show the ratio of survivable topologies found to the number of logical topologies considered. Fig. 4.4(a) and Fig. 4.4(b) compare both the methods for $n = 20$ and 50, respectively. It can be seen that both methods are comparable for different values of f. Both methods are able to map majority of the logical topologies in survivable manner for $n = 20$. However, in case of $n = 50$ success rate is much lower especially for the regular graphs. This may be due to the fact that there are more subgraphs to map, hence, more number of physical links are likely to be common to mappings of logical links.

Figure 4.5 (a) and (b) show comparison of robustness for $n = \{20, 50\}$. It can



(a) $n = 20$.



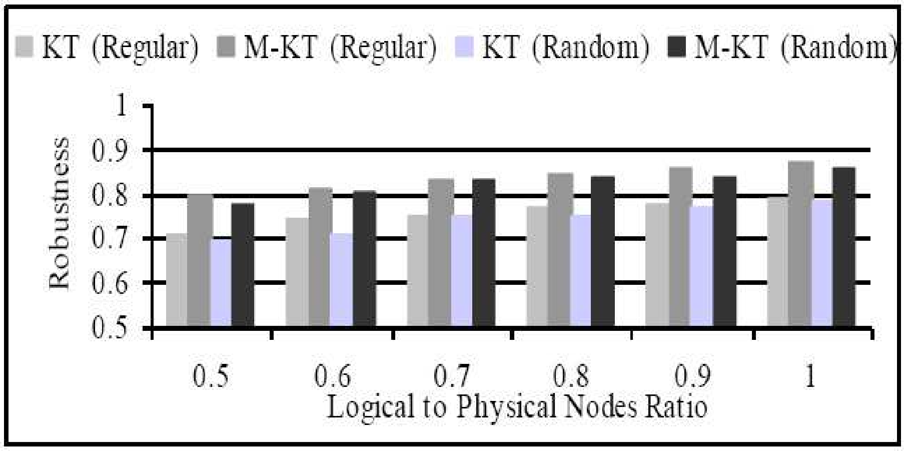
(b) $n = 50$.

Figure 4.4: KT vs. M-KT: Percentage of mapped logical topologies.

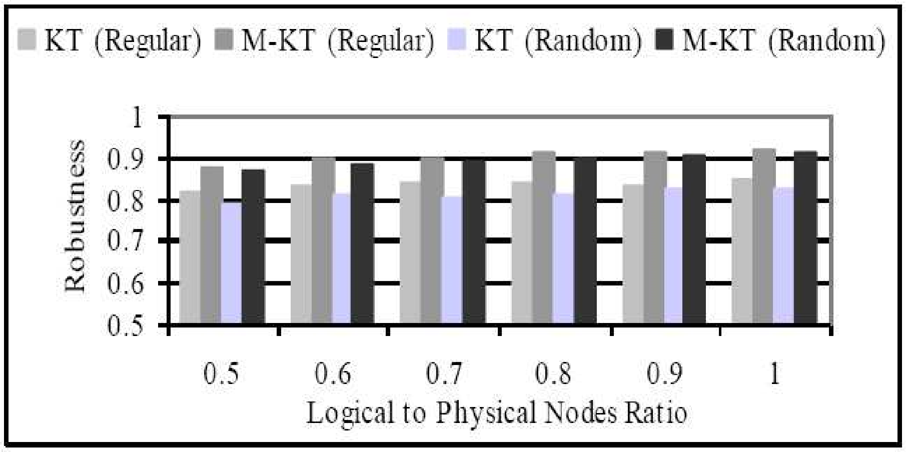
be seen that both methods provide a high degree of robustness. However, M-KT performs better due to the implementation of enhancement 1 and 3. This work was presented in [24].

4.4 Modify and Map: A Robust Survivable Topology Mapping Algorithm

Building upon the work in section 4.2, this section presents a more robust survivable topology design algorithm called *Modify and Map* (MM). The new approach gives



(a) $n = 20$.



(b) $n = 50$.

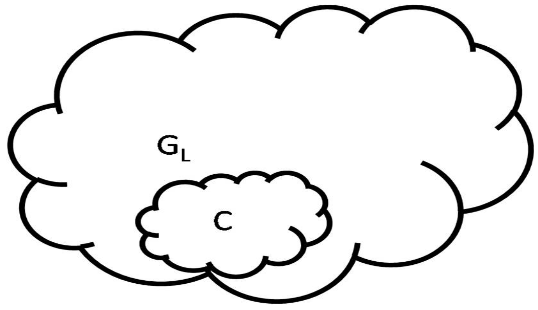
Figure 4.5: KT vs. M-KT: Comparison of robustness.

a more systematic approach to finding subgraphs (cycles) to map in the logical topology and a mapping algorithm based on the concept of randomized rounding [33] utilizing a fractional multicommodity flow approximation algorithm [34].

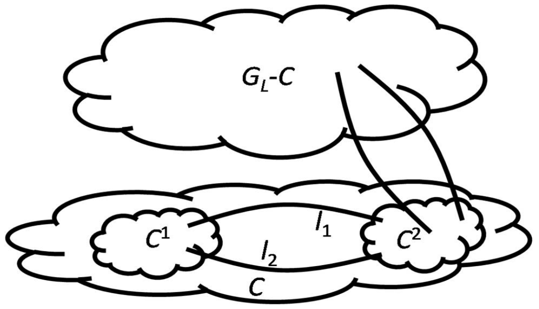
4.4.1 Subgraph Selection:

Assume that a physical topology $G(V, E)$ and a logical topology $G_L(V_L, E_L)$ are given, where V is the set of physical nodes, E is the set of physical edges, V_L is the set of logical nodes such that $V_L \subseteq V$, and E_L is the set of logical links. Also assume that both physical and logical topologies are at least 2-edge connected. The algorithm proceeds by picking a subgraph C in the form of a cycle in the logical topology and attempts to map it in the physical topology. If C can be mapped in a disjoint manner, then the logical topology is contracted by collapsing the edges and merging the nodes in C. However, if C cannot be mapped in a disjoint manner, then there is a set of links $\psi \subseteq C$, such that they share some physical link/links. In this case C will be split into $|\psi|$ components $\{C^1, C^2, ..., C^{|\psi|}\}$. Figure 4.6 shows the case when the cycle C is split into 2 components C^1 and C^2 i.e. $|\psi| = 2$. When C is split into two or more components and a component gets isolated i.e. the component is no longer connected to any other component or the graph $G_L - C$, then it is not possible for the entire logical topology to be survivable without finding a cycle, which contains the links that connect the isolated component to the other components. This scenario is shown in Fig. 4.6(b). In this case, it is necessary to do an exhaustive search to verify that no such cycle exists. Fig. 4.6(c) shows another possibility. In this case none of the components get isolated i.e. there exists a path or paths in $G_L - C$ that connect one component to the other component. In that case we proceed as follows:

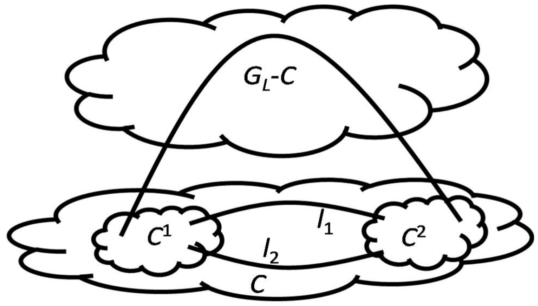
Step 1: Pick a link $l_i \in \psi$ $(1 \leq i \leq |\psi|)$, and find the two components C^1 and C^2 it connects. Let n_1 be the set of degree 3 nodes in component C^1 and n_2 be the set of such nodes in the component C^2. Now pick a node $n' \in n_1$ and a node $n'' \in n_2$. Let P' be the path between n' and n'' on the cycle C. After finding such nodes, remove all the edges belonging to the cycle C in the logical topology and find a shortest path P between n' and n''. The edges are then restored and a cycle is formed by concatenating P' and P and an attempt is made to map this cycle. If this cycle can be mapped in disjoint manner, ψ is updated by removing l_i and C is updated by collapsing the edges and merging the nodes in P'. Otherwise, this procedure is repeated by pairing other nodes from n_1 and n_2. If for a link $l_i \in \psi$, no such cycle can be found we then proceed with $l_{i+1} \in \psi$.



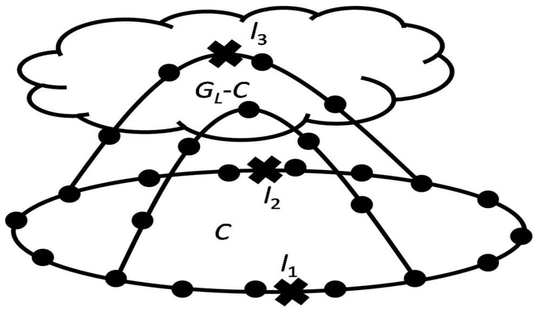
(a) A logical topology G_L and cycle C.



(b) An unsurvivable cycle C is split into two components (C^1 and C^2) and one of the components is isolated.



(c) An unsurvivable cycle C is split into two components (C^1 and C^2) and the components are not isolated.



(d) New cycles formed using the nodes in C^1 and C^2.

Figure 4.6: Illustration of Modify and Map.

Suppose that survivable mappings can be found for $|\psi| - 1$ links, then it can be easily shown that the edges in the $|\psi| - 1$ cycles along with those in C form a graph which will be connected for any single physical link failure. So we contract these edges and proceed with the remaining edges in the logical topology.

Step 2: If we are unable to find survivable cycles for $|\psi| - 1$ links i.e. there are some $l_i \in \psi$ for which survivable cycles could not be found. Let C' be the cycle after step 1. Now we pick a pair of nodes $\{x, y\}$ with degree ≥ 3 in C' and find a cycle, which includes the path on C' between x and y and a path between x and y that does not include any edge from C'. If this cycle can be mapped in survivable manner, the edges in this cycle are collapsed. This procedure is repeated until all the edges have been collapsed to create a single node or until all the nodes with degree ≥ 3 have been processed.

Step 3: Let C'' be the cycle after steps 1 and 2. If C'' is not a single node, we apply the mapping algorithm to C''. If the mapping is survivable, the edges are contracted and nodes are merged. However, if C'' cannot be mapped in a survivable manner, then the unsurvivable mapping is accepted and the logical topology is contracted but a flag is set to indicate that an unsurvivable mapping was accepted. This requirement guarantees the termination of the algorithm. The algorithm then proceeds by selecting a new cycle C.

Step 4: If the flag was set in step 3, then after the termination of the algorithm, logical edges that were not mapped earlier are mapped in an arbitrary manner and a test is conducted to see, if the mapping of the entire logical topology is survivable.

Since the problem of finding disjoint mappings is NP-complete, therefore, any algorithm other than the ILP formulation cannot guarantee that if a solution is not found then a solution does not exist. The reasoning for accepting unsurvivable mapping for the cycle and continuing is that the cycles selected in subsequent iterations may aid in making an unsurvivable subgraphs survivable (section 4.1, fact 1). The entire algorithm is shown in Figure 4.8, 4.9, 4.10 and 4.11.

4.4.2 A Rigorous Mapping Algorithm:

One key assumption in the above procedure is that there exists a method to find disjoint mappings for a given set of links. As mentioned earlier, the problem of finding mutually disjoint paths for a given set of links is NP-complete in general [26]. The problem can be formulated as an Integer Linear Program (ILP) and solved exactly [1]. Since ILP exhibits excessive runtimes, it may not be suitable for large networks. Due to the fundamental role played by disjoint paths in designing telecommunication networks, a significant amount of literature is dedicated to solving this problem without solving the ILP formulation. Most of the proposed approaches assign weights to the physical links which are then used to find paths, usually using a shortest path algorithm. After finding a path the weights are usually updated for the links in the path. The process is repeated until disjoints paths are found or a termination condition is met. Such approaches work well in practice but an unsuccessful termination does not necessarily imply that disjoint mappings are not possible.

In this section we propose a variation of the randomized rounding technique proposed by Raghavan and Thompson to find disjoint paths in an undirected network [33]. The algorithm transforms the optimal solution of a relaxed $0-1$ integer formulation into a "*provably good*" solution to the original $0-1$ formulation. The algorithm relaxes the integer constraint of the multicommodity flow problem (MCF) to obtain fractional solutions, which are then used to obtain integer solutions. [33], however, does not provide a method to solve the relaxed problem and assumes that fractional flows are available to the algorithm. [34] and [35] provide fast algorithms to solve the fractional MCF problem in polynomial times, which can then be used to get integer solutions by applying the approach in [33].

To get fractional solutions we use a modified version of the multicommodity flow approximation algorithm in [34]. To use the algorithm, the links belonging to the subgraph (a cycle C) are viewed as commodities $K = \{(s_1, t_1), (s_2, t_2), ..., (s_k, t_k)\}$. All the physical links are assigned a capacity of 1 ($u_{ij} = 1$) and the objective is to find flows f_i from s_i to t_i without violating the capacity constraints of the physical edges.

Algorithms in [34] and [35] are based on the path-flow multicommodity flow formulation to provide ϵ-approximate solutions. The formulation is given in Fig. 4.7.

Let $x(P)$ is the amount of flow sent along a path P (initially 0), then dual of the formulation in Fig. 4.7 can be formed by assigning lengths to the edges of the graph. The length l_{ij} of an edge (i, j) is related to the amount of flow it carries and

$$\begin{array}{lll} & Maximize \sum_{p\ \in\ P} x(P) & \\ \text{Capacity Constraint} & \left(\sum_{P\ :\ p \in P} x(P)\right) \leq u_{ij} & \forall p \in E \\ \text{Flow Constraint} & x(P) \geq 0 & \forall P \end{array}$$

Figure 4.7: Multicommodity flow formulation.

represents the marginal cost of using an additional unit of capacity of (i, j). The maximum flow computations are done only on this length function. The objective is to minimize $\sum_{(i,j) \in E} u_{ij} l_{ij}$, such that the length of the any shortest path between any (s_i, t_i) pair is at least 1, for all $(s_i, t_i) \in K$ $(1 \leq i \leq |K|)$. The length of each edge is initially set to $\delta = \frac{1+\epsilon}{((1+\epsilon)L)^{\frac{1}{\epsilon}}}$, where L is the length of the longest path between any $(s_i, t_i) \in K$ and ϵ is the desired accuracy.

The algorithm proceeds by finding shortest paths for all $(s_i, t_i) \in K$, using the length function. Let P_j be the set of all such paths in an iteration j. The algorithm then picks the shortest path $P \in P_j$, and updates the flows (primal variables) and lengths (dual variables) of the edges along P. The flow is updated by finding the capacity u of the minimum capacity edge (bottleneck edge) along P and adding it to $x(P)$ i.e. $x(P) = x(P) + u$. The lengths of the edges along P are then updated as $l(e) = l(e)\left(1 + \frac{\epsilon u}{u(e)}\right)$. This update ensures that the length of the bottleneck link is increased by a factor of $(1+\epsilon)$. It can be noted that the update of the primal variables may violate the capacity constraints while satisfying the non-negativity constraints, making the solution infeasible. However, a feasible solution can be determined at any stage by finding the most violated capacity constraint and scaling all the primal variables by dividing them by an appropriate scalar. Similarly, a dual feasible solution can be obtained by finding the most violated dual constraint and multiplying the dual variables by the proportion of violation.

In an iteration j, the above algorithm finds shortest paths for all $(s_i, t_i) \in K$, which requires $|K|$ applications of the shortest path algorithm. The algorithm then always chooses the commodity with the shortest path $P \in P_j$ to push flow, which may starve some commodities i.e. they may not get any flow. This is undesirable when finding disjoint paths using randomized rounding.

To circumvent this problem we modify of the above algorithm as follows. Instead of picking the commodity with the shortest path, we go through the commodities in a round robin manner to provide each commodity a chance to send flow. To introduce further fairness, the commodities are permuted in each iteration. In addition to the existing variables, we also introduce two new variables M_{Best} and $|M_{Best}|$ for the subgraph under consideration. Initially $M_{Best} = \emptyset$ and $|M_{Best}|$ is set to infinity. In each iteration, the algorithm tries to finds a shortest path for each commodity and updates $x(P)$ and $l(e)$ as described above. If shortest paths could be found for all the commodities, the algorithm checks that if all the paths are disjoint. If all the paths are edge disjoint, M_{Best} is updated and returned as the survivable mapping for the subgraph. If all the paths are not disjoint, then the number of commodities whose mappings are not disjoint is computed. If this number is less than the $|M_{Best}|$ in the previous iteration, then $|M_{Best}|$ is set to this number and the mapping obtained in this iteration are stored in M_{Best}. If shortest paths could not be found for all the commodities then flows and lengths are updated but no changes are made to M_{Best} and $|M_{Best}|$. The procedure is repeated until the length of all the shortest paths for every commodity becomes at least 1. The modified algorithm is given in Fig. 4.10.

Once fractional flows have been obtained for all $(s_i, t_i) \in K$, the randomized rounding technique proposed in [33] can be applied to get integer solutions. Let $f_k(e)$ be the fractional flow on link $e \in E$ for commodity k. Now a directed graph $G_k(V, E_k)$ is constructed for each commodity, here $E_k \subseteq E$. An edge $e \in E_k$ is assigned a direction, which is the direction of flow of commodity k in e. An edge with $f_k(e) = 0$ is removed from G_k. After constructing $G_k (1 \leq k \leq |K|)$, the next step in [33] is to find directed paths from s_k to t_k in G_k using *depth first search* (DFS) approach. Using DFS to get paths results in very long paths, which makes it harder to get integer solutions. Therefore, we propose that instead of using DFS approach, a shortest path algorithm should be used. Using a shortest path algorithm will yield paths with fewer links, which increases the probability of finding integer solutions in fewer iterations.

After finding a path P, the link with minimum flow f_m is found along P. Now f_m is subtracted from the flows of the links along P. If the flow on any link becomes zero, it is removed from G_k. The path P and flow f_m associated with it are added to a set π_k. Using the updated flows, the shortest path algorithm is applied again to find another path in G_k and the link with the minimum flow. Both are added to π_k after updating the flows. The process is repeated until there is a path from s_k to t_k in G_k. This process is called *path stripping*. The algorithm then proceeds by applying path stripping to the remaining commodities.

After obtaining paths for all the commodities, a die is cast for each π_k $(1 \leq k \leq |K|)$ with number of faces equal to $|\pi_k|$ and the face probabilities equal to the flows for the paths in π_k. Now the dice is rolled for a particular commodity and path whose face comes up is chosen as the mapping for that commodity. The process is repeated for all commodities. After obtaining mapping for all the commodities, it is checked that whether the mappings are disjoint. If the mappings are disjoint, they are accepted as the integer solution to the fractional flow problem. Otherwise, the process is repeated until disjoint mappings are found or the maximum number of iterations allowed is reached. The algorithm also keeps track of the best mapping encountered so far, in a manner similar to the multicommodity flow approximation algorithm. The complete algorithm is given in Fig. 4.11.

```
INPUT: An at least 2-edge connected physical topology $G(V, E)$
and a logical topology $G_L$ ($V_L$, $E_L$).
OUTPUT: One link survivable mapping $M$ of $G_L$ in $G$.
While the termination condition is NOT met do
  $C \leftarrow$ a short cycle in $G_L$
  Call the mapping procedure.
  If mapping procedure returns a survivable mapping for $C$ then
    Contract $G_L$ by collapsing the edges and merging nodes in $C$.
  Else
    call modify and map.
  End if
End While
If Survivable = $false$ then
  Map the unmapped edges.
  For all physical edges do
    Remove the physical edge from the physical topology and remove
    the logical links that use this edge from the logical topology.
    If the logical topology remains connected then
      continue.
    Else
      Logical topology cannot be mapped in survivable manner.
      END
    End If
  End For
End If
```

Figure 4.8: Main routine.

4.4.3 Simulation Study and Results

To study and compare the behavior of the proposed algorithm, and SMART-H, they were implemented using LEDA [29] and VC++ 8.0. For simulation, physical and logical topologies with varying number of nodes and edges were generated. Physical topologies were regular topologies with 500 and 1000 nodes ($|V|$) and degree 6 and 8. The regular topologies were constructed using a procedure originally given by Harary [30] and described in [31]. Logical topologies were random topologies generated using the procedure described in [32]. The logical nodes ($|V_L|$) were a random subset of physical nodes ($0.8 \times |V|$). The number of logical links (E_L) were 1.5 and 2.0 times

```
**INPUT: An at least 2-edge connected physical topology $G(V, E)$ and a logical topology $G_L$ ($V_L$, $E_L$), a cycle $C$.**
**OUTPUT: Return success if the mapping is survivable.**
**Procedure modify and map:**
$\psi \leftarrow$ logical links belonging to $C$ for which disjoint mappings could not be found.
For all logical link $l_i \in \psi$ do
  $C_i^1 \leftarrow$ a component that $l_i$ connects.
  $C_i^2 \leftarrow$ other component that $l_i$ connects.
  $n_1 \leftarrow$ list of nodes with degree $> 2$ in $C_i^1$.
  $n_2 \leftarrow$ list of nodes with degree $> 2$ in $C_i^2$.
  If $n_1 = \emptyset$ OR $n_2 = \emptyset$ then
    continue.
  End If
  For all $n' \in n_1$ do
    For all $n'' \in n_2$ do
      $P' \leftarrow$ path between $n'$ and $n''$ on the cycle $C$.
      $P \leftarrow$ path between $n'$ and $n''$ in $G_L$ - $C$.
      $C_i \leftarrow P \cup P'$.
      If mapping procedure returns a survivable mapping for $C_i$ then
        Contract $G_L$ by collapsing edges and merging nodes in $C_i$.
        Contract $C$ by collapsing edges and merging nodes in $P'$.
        Remove $l_i$ from $\psi$.
        Continue with the next $l_i$ in $\psi$.
      End If
    End For
  End For
  If $|\psi| < 2$ then
    Contract $G_L$ by collapsing remaining edges and merging nodes in $C_i$.
    Return *success*.
  End If
End For
$D$ = list of nodes with degree $> 2$ on cycle $C'$.
While $|D| > 1$ do
  $\gamma_1 \leftarrow$ remove a node from $D$.
  For all nodes $\gamma_2 \in D$ do
    $P' \leftarrow$ path between $\gamma_1$ and $\gamma_2$ on the cycle $C'$.
    $P \leftarrow$ a short path between $\gamma_1$ and $\gamma_2$ in $G_L$ - $C'$.
    $C_i' \leftarrow P \cup P'$.
    Call the mapping procedure.
```

Figure 4.9: Modify and Map.

```
      **If** mapping procedure returns a survivable mapping
      for $C'_i$ **then**
        Contract $G_L$ by collapsing edges and merging nodes in $C'_i$.
        Contract $C'$ by collapsing edges and merging nodes in $P'$.
      **End if**
    **End For**
**End While**
**If** subgraph is reduced to a single node **then**
  **Return** *success.*
**Else**
  **If** mapping procedure returns a survivable mapping for $C''$ **then**
    Contract $G_L$ by collapsing the edges and merging nodes in $C''$.
    **Return** *success.*
  Else
    Contract $G_L$ by collapsing the edges and merging nodes in $C''$.
    Survivable = *false.*
    **Return** *success.*
  **End if**
**End If**
```

Figure 4.9: Modify and Map. (Contd.)

the logical nodes. After generating the logical topologies, they were admitted for further processing only if they were at least 2-edge connected. SMART-H and the proposed algorithm were then applied to each logical-physical topology pair. The total number of such logical-physical topology pairs was 1200 for each case (40 physical and 30 logical topologies). To make the SMART-H practical, if a disjoint mapping could not be obtained after applying the mapping method 100 times for $|V| = 500$ and 200 for $|V| = 1000$, a new subgraph was selected and SMART-H terminated after unsuccessfully examining 100 subgraphs for $|V| = 500$ and 200 for $|V| = 1000$. For the proposed algorithm, ϵ was 0.15 and the number of randomized rounds were set to 50. To find a subgraph, two nodes were randomly picked and two link disjoint paths were found between these nodes using a maximum flow implementation [25]. The two paths were then concatenated to get the subgraph.

The results are summarized in Table 4.1 and 4.2. It can be seen that the proposed

```
INPUT: A network $G$, capacities $u(e)$, commodity pairs $K = \{(s_k, t_k)\}$, $1 \leq k \leq |K|$, and accuracy $\epsilon$.
OUTPUT: Best mapping possible for pairs in $K$, solve fractional multicommodity flow problem.
Procedure mapping algorithm:
Initialize $l(e) = \delta$.
$\forall e, x \equiv 0$.
$K' \leftarrow K$.
Best Mapping $M_{Best} \leftarrow \emptyset$.
$|M_{Best}| \leftarrow \alpha$.   /*number of non-disjoint links*/
Commodities $\leftarrow |K'|$.
While there is a path $P$ between a $(s_k, t_k) \in K'$ such that $l(P) < 1$ do
  Mapping $M \leftarrow \phi$.
  For all $(s_k, t_k) \in K'$ do
    Find the shortest paths $P$ using $l$.
    $M \leftarrow M \cup P$.
    If $l(P) < 1$ then
      $u \leftarrow min_{e \in P} u(e)$.
      $x(P) \leftarrow x(P) + u$.
      $\forall e \in P,\ l(e) \leftarrow l(e)\left(1 + \frac{\epsilon u}{u(e)}\right)$.
    else
      $K' \leftarrow \{K' - (s_k, t_k)\}$.
    End If
  End For
  If Commodities $= |K'|$ and $M$ is edge disjoint then
    Return success.
  Else If commodities $= |K'|$ then
    If number of non-disjoint links in $M < |M_{Best}|$ then
      $M_{Best} \leftarrow M$.
    End If
  End If
  permute $K'$.
End While
Call Randomized Rounding.
```

Figure 4.10: Mapping algorithm using fractional multicommodity flow approximation algorithm.

algorithm can map more logical topologies in survivable manner then SMART-H (Table 4.1). As one would expect, the proposed algorithm does more exploration and so performs more number of shortest path computations, leading to increased

```
INPUT: A network $G$, capacities $u(e)$, flow matrix $f$, commodity pairs $K = (s_k, t_k)$, $1 \leq k \leq |K|$, the Best Mapping from mapping algorithm $M_{Best}$.
OUTPUT: Disjoint paths.
Procedure randomized rounding:
$\pi_k \leftarrow$ set of paths for commodity $k$.
For all commodities $k$ in $K$ do
  $G_k(V, E_k) \leftarrow$ A directed graph with $E_k \subseteq E$ such that $e \in E_k$ iff $e$
  has a non-zero fractional flow in solution to MMCF for commodity $k$.
  While there is non-zero flow leaving $s_i$ do
     $P \leftarrow$ a directed path in $G_k$ for commodity $k$, using Dijikstra [21]
     shortest path algorithm.
    $\pi_k \leftarrow \pi_k \cup P$.
    $f_m \leftarrow$ flow along the bottleneck link (minimum flow) in path $P$.
    Add the path $P$ and the flow $f_m$ to $\pi_k$.
    Reduce flows for all $e \in P$ for commodity $k$ by $f_m$.
    Remove the edges from $G_k$ for commodity $k$ for which $f_e = 0$.
  End While
End For
While no of rounds is less than threshold do
  Mapping $M \leftarrow \emptyset$.
  For all commodities $k$ in $K$ do
    Cast a die with number of faces = $|\pi_k|$ with the face probabilities
     equal to flows for the paths in $\pi_k$.
    Roll the dice and assign the path $P$ to commodity $k$ whose
    face comes up.
    $M \leftarrow M \cup P$.
  End For
  If $M$ is edge disjoint then
    Return success.
  Else If number of non-disjoint links in $M < |M_{Best}|$ then
    $M_{Best} \leftarrow M$.
  End if
End While
```

Figure 4.11: Randomized Rounding.

execution times (Table 4.2). But the increased execution time is compensated by the increased number of survivable mappings generated. To understand why SMART-H finds fewer survivable mappings, first recall that when a mapping of a subgraph is not survivable, only the weights of the shared physical edges are updated in SMART-H.

$\|V\| = 500, \quad \|V_L\| = 400$					
	Degree	$\|E_L\|$	No. of Survivable Mappings	Ave. subgraph size (edges)	Ave Time per Topology (min)
MM	6	600	884	4	5.074
SMART-H	6	600	800	4	4.286
MM	8	600	1056	3	5.735
SMART-H	8	600	932	3	5.081
MM	6	800	1061	3	3.895
SMART-H	6	800	1004	3	3.027
MM	8	800	1101	3	3.344
SMART-H	8	800	1020	3	2.875

Table 4.1: Simulation Results for $|V| = 500$ and $|V_L| = 400$.

$\|V\| = 1000, \quad \|V_L\| = 800, \quad$ Degree=8, $\quad \|E_L\| = 1600$				
	No. of Survivable Mappings	Ave. Subgraph Size edges	Ave Time per Topology (min)	Ave. No of Shortest path applications
MM	1104	3	12.88	3598
SMART-H	991	3	9.286	3085

Table 4.2: Simulation Results for $|V| = 1000$, $|V_L| = 800$, Degree = 8, and $|E_L| = 1600$.

This may modify the paths obtained in the next iteration only slightly. This means that the mapping generated in the next iteration may still not be survivable. Since the number of unsuccessful attempts is set to a predetermined value, SMART-H may not be examining as many mappings as one would have expected. Since the subgraph is selection is random, therefore, it may take several unsuccessful attempts before a survivable subgraph can be found. Also SMART-H does not have a mechanism to keep track of the subgraphs that have already been considered so it may pick an unsurvivable subgraph several times, thereby again not exploring as many subgraphs as one would have expected.

To explain why the proposed algorithm takes more execution time, note that if a subgraph is not survivable, then subgraph modification procedure in section 4.4.1 is applied, which may map the chosen subgraph and/or some subgraphs surrounding it. However, finding such subgraphs requires extra overhead. Also, keeping track of the best mapping is another overhead involved. The increased time required by the overheads is alleviated to some extent by the efficient randomized rounding approach. Also, in the randomized rounding approach the weights (length) of all the links that are part of the current mapping are updated, thereby reducing the possibility of an unsuccessful mapping being considered more than once. It can also be seen in Table 4.1 and 4.2 that when the topologies (logical or physical) are sparse, fewer logical topologies can be mapped in survivable manner. However, making the topologies dense increases the number of survivable mapping. This is because dense logical topology implies that more subgraphs are available. Also if the physical topology is dense a larger number of paths will be available. This work was presented in [36].

4.5 Chapter Summary and Conclusions

In this chapter, we analyzed SMART and SMART-H in detail. The analysis pointed out certain shortcomings of the SMART and SMART-H, which allowed us to propose several enhancements that improved their performance.

In this chapter, we also introduced a new embedding algorithm based on randomized rounding and fractional multicommodity flow approximation. The new embedding algorithm combined with a new subgraph finding algorithm was able to find survivable mappings for a larger number of IP-over-WDM network pairs. However, the simulation results showed that for some logical-physical topology pairs even the new algorithm might not be able to find survivable mappings. This could be due to the fact that the mapping algorithm terminated prematurely or a survivable mapping for this particular was not possible. Therefore, in the next chapter we will present a new approach that uses a combination of protection and restoration to find survivable mappings for all the logical-physical topology pairs.

Chapter 5

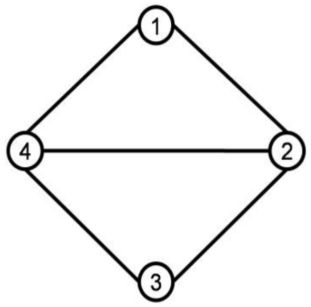
A Hybrid Approach to Survivability

In this chapter a new approach is presented that uses a combination of restoration and protection strategies to find survivable mappings for logical topologies. Protection is provided by establishing additional lightpaths in such a way that the entire logical topology becomes survivable. The approach, called hybrid approach, is able to find survivable mappings for all the logical-physical topology pairs as long as physical topologies are at least 2-edge connected.

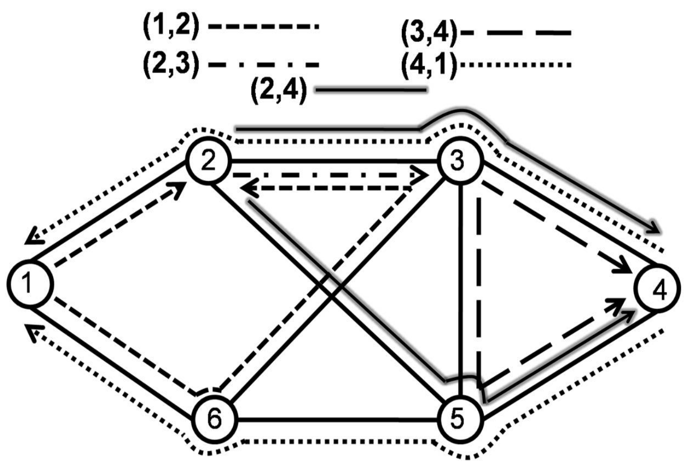
5.1 Motivation and Background

A simple way to provide survivability in an IP-over-WDM network is to establish two edge disjoint lightpaths between the end nodes of all the IP links, which is similar to 1:1 APS protection. One of the lightpaths is called a *primary lightpath* (or *working lightpath*) and carries normal network traffic under normal circumstances. The other lightpath is called a *backup lightpath.* The traffic carried by the primary lightpath is switched to the backup lightpath only if a physical link failure disconnects the primary lightpath. Two disjoint paths can be found using maximum flow techniques

[25] or some of the other algorithms available in literature (e.g. Suurballe algorithm [37]).



(a) A logical topology.

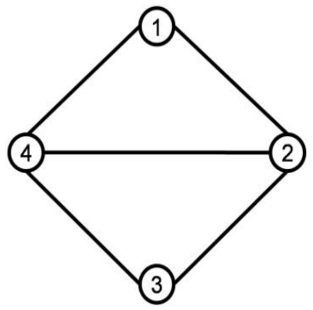


(b) Two lightpaths established for each logical link in the physical topology (similar to 1:1 APS protection).

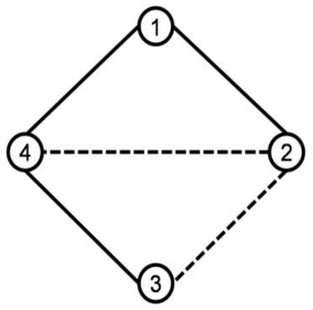
Figure 5.1: Illustration of 1:1 protection.

Fig. 5.1 shows an example of this approach. The main drawback of this approach is that the total number of lightpaths to be established is $2 \times |E_L|$, where $|E_L|$ is the number of logical links. However, this approach guarantees survivability, if the number of physical link failures allowed is restricted to exactly one.

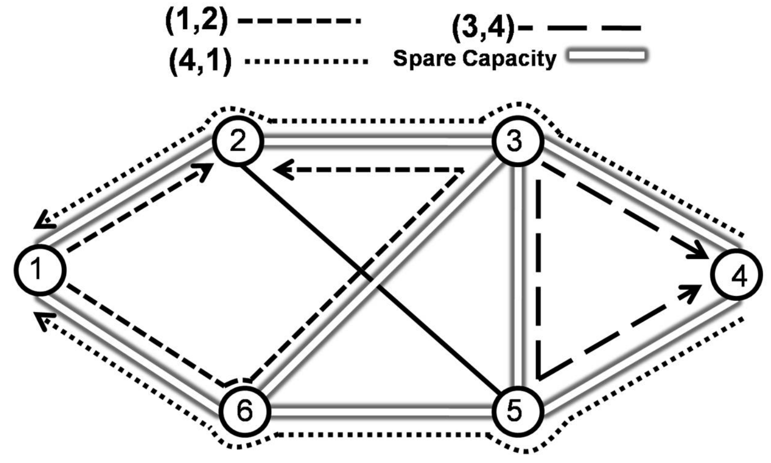
In the above approach, the number of backup lightpaths to be established can be reduced by picking a spanning tree in the logical topology and setting up two disjoint lightpaths for tree edges only. For non-tree edges, only a single lightpath is established arbitrarily. The network is then provisioned with some spare capacity. Now, a single physical link failure may cause multiple logical link failures. If a tree edge or edges fail, the traffic carried by the failed link is switched to its corresponding backup lightpath. However, if a non-tree edge or edges fail the spare capacity available in the network is used to find backup path/paths for the failed lightpaths. It can be observed that the total number of lightpaths to be reserved in this case is $2(|V_L| - 1) + (|E_L| - |V_L| + 1)$, where $|V_L|$ is the number of logical nodes and $|E_L|$ is the number of logical edges.



(a) A logical topology.



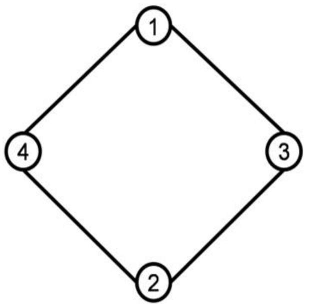
(b) Tree (bold) and non-tree (dashed) edges.



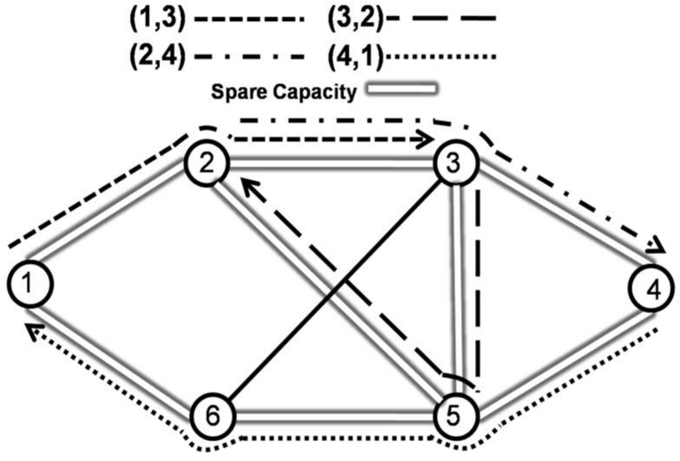
(c) Two lightpaths established for each tree edge in the physical topology (similar to 1:1 protection), one lightpath for the non-tree edge (not shown here).

Figure 5.2: Illustration of hybrid protection.

An example of this approach is given in Fig. 5.2. Fig. 5.2(a) shows the logical topology. Fig. 5.2(b) shows a spanning tree of the logical topology with bold lines and non-tree edges with dashed lines. Fig. 5.2 (c) shows the physical topology with two lightpaths established for each tree edge. It can be observed that the failure of a single physical link will not disconnect the logical topology i.e. it is still possible to reach all the logical nodes using the spare capacity available. Fig. 5.2(c) does not show the lightpaths that must be established for the non-tree edges. These lightpaths will be established arbitrarily. For example, a lightpath for the logical link $(2, 4)$ can be set up along the physical path $2-3-4, 2-3-5-4$, or $2-1-6-5-4$ etc. Similarly, a lightpath for the logical link $(2, 3)$ can be set along path $2-3, 2-1-6-5-3$ or $2-1-6-3$ etc.



(a) A logical topology.



(b) A mapping of the logical topology in the physical topology.

Figure 5.3: An unsurvivable logical-physical topology pair.

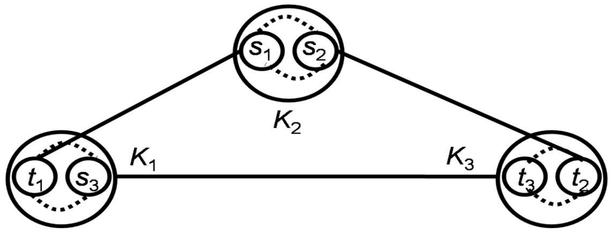
In the next section, we provide a hybrid survivability approach to design survivable IP-over-WDM network that utilizes the SMART framework to find links that must be protected i.e. links for which two disjoint paths must be provided in the physical topology. Our goal is to minimize the number of protection edges to be added.

5.2 A Hybrid Survivable Mapping Algorithm

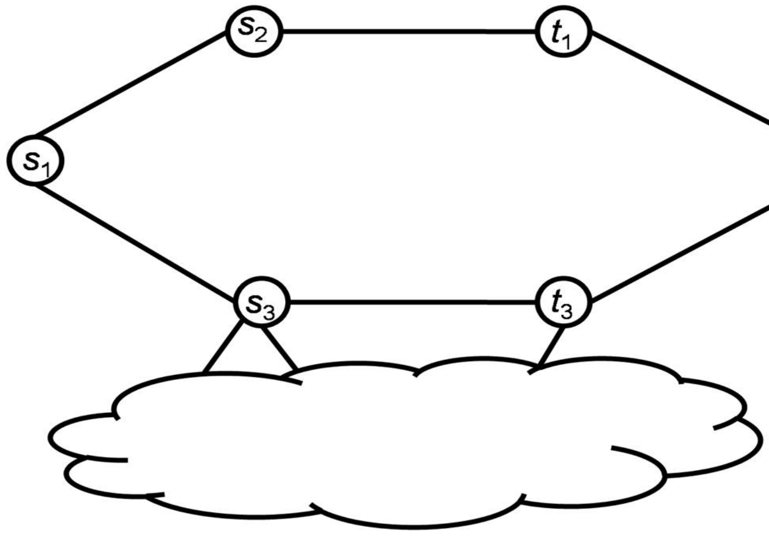
Assume that we are given a ring logical topology G_L, as shown in Fig. 5.3(a). Now, according to [1], all the logical links in G_L must be mapped to lightpaths that follow mutually disjoint paths in the physical topology. But for G_L, shown in Fig. 5.3(a), such a mapping does not exist in the physical topology shown in Fig. 5.3(b). Therefore, such a logical topology must be rejected by the network operators which may result in loss of revenue.

An obvious alternative to rejecting a logical topology, if a survivable mapping for it cannot be found in a physical topology, is to add additional logical or physical links. Adding new physical links by laying new fibers is usually prohibitively expensive and in some cases impossible due to right of way and geographical limitations. However, new logical links can be added to the logical topology simply by establishing additional lightpaths that can make an unsurvivable logical-physical topology pair survivable. However, given the number of choices available the pair of nodes between which additional lightpaths can be established could be very large. Also, the number of links to be added may not be known in advance.

[2] suggests a simple procedure to add an additional link. The procedure requires an application of SMART-H (the heuristic version of the algorithm given in Fig. 3.6). If the procedure terminates unsuccessfully, it returns a contracted logical topology and a set of mappings for the contracted components (subgraphs).



(a) A logical topology.



(b) A mapping of the logical topology in the physical topology.

Figure 5.4: An unsurvivable logical-physical topology pair.

At this stage an additional logical link is added to the logical topology by randomly picking two nodes in the contracted logical topology and finding the corresponding logical nodes in the logical topology. A new logical link is established between the two nodes, if one does not already exist. Otherwise, a different pair of nodes is chosen. After adding the new logical link, SMART-H is applied again. If the algorithm terminates successfully, then a survivable mapping exists for the new logical mapping. Otherwise the new logical topology is deemed unsurvivable and rejected.

It is possible to keep on adding more links to the logical topology until a survivable mapping is found or no more edges can added i.e. the contracted logical topologies becomes a complete graph. However, even this does not guarantee that a survivable mapping for the logical topology is possible in general. For example, consider the physical topology G shown in Fig. 5.4(b) and contracted logical topology G_C shown in Fig. 5.4(a). It is easy to verify that two edge disjoint paths, P_1 from s_1 to t_1 and P_2 from s_2 to t_2, do not exist in G. Therefore, a survivable mapping for the subgraph does not exist and it is also not possible to add additional links to the subgraph. However, it is possible to protect G_C against a failure in the physical topology by finding two disjoint paths for each edge in G_C or by finding a spanning tree and finding two disjoint paths for the tree edges.

In the following, as an alternative to adding a logical link, we develop a hybrid survivability approach that uses a combination of restoration and protection to find survivable mapping for a given logical topology in a physical topology.

5.2.1 Hybrid Algorithm 1 (HA-1)

Assume that we are given a ring logical topology G_L (shown in Fig. 5.3(a)) and a physical topology G (shown in Fig. 5.3(b)). An application of SMART-H will terminate unsuccessfully since a survivable mapping for G_L in G does not exist. After adding a logical edge, as suggested in [2], between nodes 1 and 2 or between 3

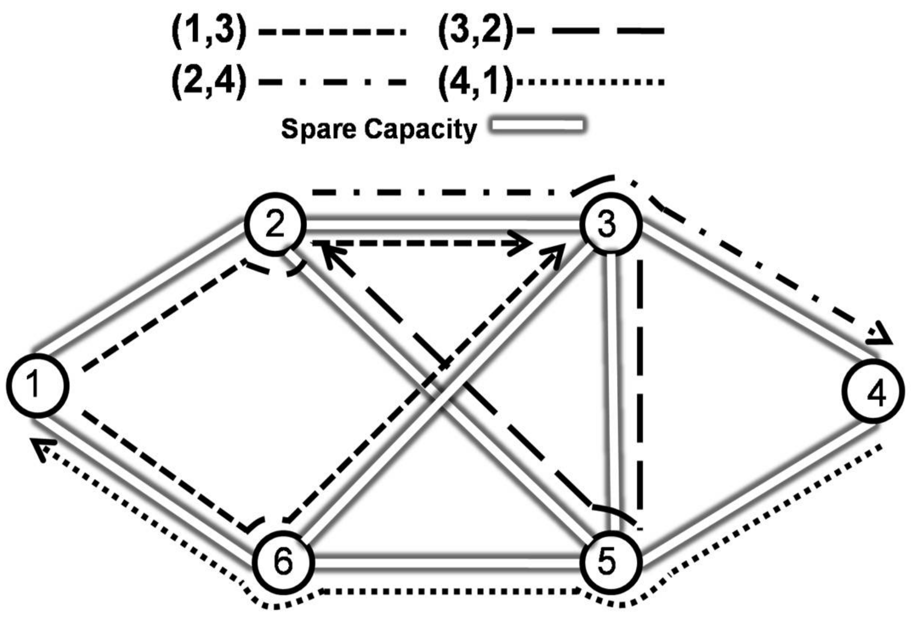


Figure 5.5: Illustration of hybrid approach, two lightpaths are established for the logical link $(1, 3)$ i.e. $1-2-3$ and $1-6-3$.

and 4, a survivable mapping can be found for G_L. However, we suggest an alternate approach.. If SMART-H terminates unsuccessfully then there is a set of links in G_L for which disjoint mappings could not be found. We call such a set of links *critical links* (π). For example, in Fig. 5.3 logical links $(1, 3)$ and $(2, 4)$ are critical links because their mappings are not disjoint i.e. $\pi = \{(1, 3), (2, 4)\}$. Failure of physical link $(2, 3)$ can disconnect both $(1, 3)$ and $(2, 4)$ and the logical topology is no longer connected. Therefore we propose that for such links protection should be provided by finding two link disjoint paths for all or some of the critical links. Two disjoints paths are easier to find using a maximum flow algorithm [25], Suurballe algorithm or any other appropriate algorithm [37]. In contrast to finding mutually disjoint mappings (NP complete) finding two disjoint paths in a two connected topology is fairly simple.

Fig. 5.5 illustrates the new approach. Since $\pi = \{(1, 3), (2, 4)\}$, Fig. 5.5 shows the mappings when protection is provided for $(1, 3)$ by setting up two lightpaths rather than just one. Now it can be seen that the failure of any single physical link will not disconnect the logical topology. For example, if physical link $(2, 3)$ fails lightpaths $(1, 3)$ and $(2, 4)$ get disconnected but logical link $(1, 3)$ remains connected through the alternate lightpath for $(1, 3)$ via path $1 - 6 - 3$. Hence the entire logical topology remains connected. Fig. 5.5 also illustrates that there is no need to provide protection for $(2, 4)$, since routing $(1, 3)$ and $(2, 4)$ together no longer makes the topology unsurvivable. In fact, if there are $|\pi|$ critical links then we have to provide protection for only $|\pi| - 1$ critical links in the worst case.

In case of a general logical topology, finding an optimal set of critical links in a topology may not be possible. Therefore, we propose utilizing SMART-H as an efficient heuristic to find such links. The set of critical links obtained using SMART-H may not be optimal i.e. some of these links may not be critical. Some of these links may have been declared critical because the heuristic used by SMART-H to find disjoint mappings terminated unsuccessfully, although disjoint mappings were possible using an ILP or some other algorithm. Therefore, we call the set of links deemed critical by SMART-H, *pseudo-critical links* (η).

To find pseudo-critical links using SMART-H, a simple modification to SMART-H is made by introducing a new variable, best mapping (M_{Best}). M_{Best} for a given subgraph (a cycle), is a set of mappings that has the fewest number of non-disjoint mappings. The logical links that have non-disjoint mappings are the pseudo-critical links. Also, let $|M_{Best}|$ be the number of links in the chosen cycle that have non-disjoint mappings in M_{Best}. The modified mapping algorithm is shown in Fig. 5.6.

Here, SMART-H proceeds by recursively picking cycles. When survivable mappings for a particular cycle C are not returned by the mapping algorithm, it proceeds by picking a new cycle. However, when the modified mapping algorithm (shown in Fig. 5.6) fails to find survivable mappings for a cycle C, it returns a set of pseudo-critical links (η). By providing protection for $|\eta| - 1$ links in η and accepting the mappings returned by the mapping algorithm for the remaining links in C for which protection was not provided, the entire cycle becomes survivable. The modified algorithm called *Hybrid Algorithm-1* (HA-1) is shown in Fig. 5.7.

The algorithm in Fig. 5.7 (HA-1) determines pseudo-critical links on a per cycle basis that would result in a large number of logical links that are protected. To reduce the number of such links, we propose three variants of the HA-1 in Fig. 5.8, 5.9, and 5.10.

5.2.2 Hybrid Algorithm 2 (HA-2)

The algorithm shown in Fig. 5.8 (HA-2) also proceeds by picking cycles. If a cycle cannot be mapped in disjoint manner, the mapping algorithm returns η, a set of links for which disjoint mapping could not be found. Now pick a link $l \in \eta$ and provide protection for such an l. After providing protection, l is contracted and the algorithm proceeds normally by picking another subgraph from the contracted topology.

5.2.3 Hybrid Algorithm 3 (HA-3)

Fig. 5.9 shows a variant (HA-3) that waits for the SMART-H to terminate unsuccessfully. An unsuccessful termination returns η and M_{Best} for the last subgraph considered by SMART-H. The algorithm then provides protection for $|M_{Best}| - 1$ links in η.

5.2.4 Hybrid Algorithm 4 (HA-4)

Figure 5.10 shows another variant (HA-4) that also waits for the SMART-H to terminate unsuccessfully. An unsuccessful termination returns η in the last subgraph considered. The algorithm picks a link $l \in \eta$, provides protection for l and contracts it. The algorithm then proceeds by choosing another subgraph from the contracted logical topology.

```
INPUT: An at least 2-edge connected physical topology $G(V,E)$, a cycle $C$.
OUTPUT: Best mapping $M_{Best}$, $|M_{Best}|$ and $\eta$.
Procedure find mappings:
Best mapping $M_{Best} \leftarrow \emptyset$.
Number of non-disjoint mappings in $M_{Best}$, $|M_{Best}| \leftarrow \propto$.
Pseudo-critical links $\eta \leftarrow \emptyset$.
$\forall e \in E$, $u(e) = 1$.
While the termination condition is not met do
   Mapping $M \leftarrow \emptyset$.
   For all links $l_i \in C$ do
     $s \leftarrow$ source of link $l_i$.
     $t \leftarrow$ destination of link $l_i$.
     $M[i] \leftarrow$ a shortest path between $s$ and $t$ in $G$ using $u$.
   End For
   If $(M[0] \cap M[1] \cap .... \cap M[i]) = \emptyset$ then
     Disjoint mappings have been found.
     $M_{Best} \leftarrow M$.
     $\eta \leftarrow \emptyset$.
     Break.
   Else
   $\beta \leftarrow$ Physical links belonging to more than one mapping in $M$.
   $\forall e \in \beta, u(e) = u(e) + 1$.
     $\delta \leftarrow$ number of non-disjoint mappings in $M$.
     If $\delta < |M_{Best}|$ then
       $M_{Best} \leftarrow M$.
       $|M_{Best}| \leftarrow \delta$.
       $\eta \leftarrow l_i \in C$ for which disjoint mappings could not be found.
     End If
   End If
End While
Return $M_{Best}, |M_{Best}|$ and $\eta$.
```

Figure 5.6: Mapping algorithm.

```
INPUT: An at least 2-edge connected physical topology $G(V, E)$ and a logical topology $G_L(V_L, E_L)$.
OUTPUT: One link survivable mapping $M$ of $G_L$ in $G$.
Procedure hybrid algorithm 1:
While the logical topology is not reduced to a single node do
  $C \leftarrow$ a short cycle in $G_L$.
  Call the mapping procedure with cycle $C$.
  If $\eta$ returned by the mapping algorithm is empty then
    Contract $G_L$ by collapsing the edges and merging the nodes in $C$.
  Else
    For $(|M_{Best}| - 1)$ links in $\eta$ do
      $l_i \leftarrow$ a link in $\eta$.
      $s \leftarrow$ source of link $l_i$.
      $t \leftarrow$ destination of link $l_i$.
      Find two disjoint paths $P_1$ and $P_2$ between $s$ and $t$ in $G$.
      Add $P_1$ and $P_2$ to $M$ as the mappings for $l_i$.
    End For
    Copy mappings of link $(l_i \in C \wedge l_i \notin \eta)$ from $M_{Best}$ to $M$.
    Contract $G_L$ by collapsing the edges and merging the nodes in $C$.
  End if
End While
```

Figure 5.7: Hybrid algorithm 1 (HA-1).

5.3 Simulations and Results

To evaluate the proposed algorithms and SMART-H, simulation studies were conducted using VC++ 8.0. For simulation studies, random logical and physical topologies with varying number of nodes and degree were generated. The topologies were selected for processing, if the topologies were at least 2-edge connected.

Physical topologies were random topologies with 100, 200 and 300 nodes ($|V|$) with average degree 3. The logical topologies were also random topologies containing a random subset of physical nodes with degree 2.5 and 3.0. The physical and the logical topologies were generated using the procedure given in [32]. The number of logical nodes in the logical topology was set to $0.75 \times |V|$. The total number of such logical-physical topology pairs was 1000 for each case (40 physical and 25 logical

```
INPUT: An at least 2-edge connected physical topology $G(V, E)$ and a logical topology $G_L(V_L, E_L)$.
OUTPUT: One link survivable mapping $M$ of $G_L$ in $G$.
Procedure hybrid algorithm 2:
While the logical topology is not reduced to a single node do
  $C \leftarrow$ a short cycle in $G_L$.
  Call the mapping procedure with cycle $C$.
  If $\eta$ returned by the mapping algorithm is empty then
    Contract $G_L$ by collapsing the edges and merging the nodes in $C$.
  Else
    Pick an edge $l_i \in \eta$.
    $s \leftarrow$ source of link $l_i$.
    $t \leftarrow$ destination of link $l_i$.
    Find two disjoint paths $P_1$ and $P_2$ between $s$ and $t$ in $G$.
    Add $P_1$ and $P_2$ to $M$ as the mappings for $l_i$.
    Contract $l_i$ by collapsing the edge and merging the nodes $s$ and $t$.
  End if
End While
```

Figure 5.8: Hybrid algorithm 2 (HA-2).

topologies).

To make the disjoint mapping algorithm practical (Fig. 5.6), the number of iterations was limited to 100. To find a cycle, two nodes (s and t) were randomly picked and two link disjoint paths (M_1 and M_2) were found using a maximum flow algorithm [25] and provisioning the network with some spare capacity. The cycle was then constructed by $P_1 \cup P_2$. The number of subgraphs to be examined before unsuccessful termination was also 100 (MAX_ITER). SMART-H and the proposed algorithms do not keep track of subgraphs for which survivable mappings could not be found. Therefore, MAX_ITER was reduced to $MAX\_ITER - (MAX\_ITER - 10)$ when the subgraph size was equal to the contracted topology size for 10 consecutive iterations.

The statistics of interest were the number of logical topologies for which survivable mappings could be found in a physical topology, number of links added to a

```
INPUT: An at least 2-edge connected physical topology $G(V, E)$
and a logical topology $G_L(V_L, E_L)$.
OUTPUT: One link survivable mapping $M$ of $G_L$ in $G$.
Procedure hybrid algorithm 3:
While the logical topology is not reduced to a single node do
  While the termination condition is not met do
    $C \leftarrow$ a short cycle in $G_L$.
    Call the mapping procedure with cycle $C$.
    If $\eta$ returned by the mapping algorithm is empty then
      Contract $G_L$ by collapsing the edges and merging the nodes in $C$.
    End if
  End While
  For $(|M_{Best}| - 1)$ links in $\eta$
    $l_i \leftarrow$ a link in $\eta$.
    $s \leftarrow$ source of link $l_i$.
    $t \leftarrow$ destination of link $l_i$.
    Find two disjoint paths $P_1$ and $P_2$ between $s$ and $t$ in $G$.
    Add $P_1$ and $P_2$ to $M$ as the mappings for $l_i$.
  End For
  Copy mappings of link $(l_i \in C \wedge l_i \notin \eta)$ from $M_{Best}$ to $M$.
  Contract $G_L$ by collapsing the edges and merging the nodes in $C$.
End While
```

Figure 5.9: Hybrid algorithm 3 (HA-3).

logical topology in each algorithm and the execution time of the algorithms. Table 5.1, 5.2 and 5.3 summarize the results.

In Table 5.1, it can be seen that the proposed algorithms can map all the logical topologies in survivable manner by providing protection for some logical links. SMART-H can also map more topologies after the addition of a logical link (SMART-H+1) but some topologies still remain unsurvivable. It can also be observed that the number of mapped topologies, for a particular physical-logical degree, is relatively constant. Increasing the logical degree significantly increases this number, which is expected because the subgraphs becomes smaller in size and the probability of finding a survivable mapping increases.

Table 5.2 shows the average number of protected links in the proposed ap-

```
INPUT: An at least 2-edge connected physical topology $G(V,E)$
and a logical topology $G_L(V_L,E_L)$.
OUTPUT: One link survivable mapping $M$ of $G_L$ in $G$.
Procedure hybrid algorithm 4:
While the logical topology is not reduced to a single node do
  While the termination condition is not met do
    $C \leftarrow$ a short cycle in $G_L$.
    Call the mapping procedure with cycle $C$.
    If $\eta$ returned by the mapping algorithm is empty then
      Contract $G_L$ by collapsing the edges and merging the nodes in $C$.
    End if
  End While
  Pick an edge $l_i \in \eta$.
  $s \leftarrow$ source of link $l_i$.
  $t \leftarrow$ destination of link $l_i$.
  Find two disjoint paths $P_1$ and $P_2$ between $s$ and $t$ in $G$.
  Add $P_1$ and $P_2$ to $M$ as the mappings for $l_i$.
  Contract $l_i$ by collapsing the edge and merging the nodes $s$ and $t$.
End While
```

Figure 5.10: Hybrid algorithm 4 (HA-4).

proaches. In case of SMART-H+1, only one additional logical link was added that significantly increased the number of mapped topologies. HA-1 and HA-2 provide protection for a large number of logical links but execution time is comparatively very small (Table 5.3). HA-3 and HA-4 require protection for fewer links but the execution time is significantly higher since they must consider a larger number of subgraphs. It can also be noted that when the logical degree is increased, protection is required for fewer links and the execution time is also less. This work was presented in [38].

5.4 Chapter Summary and Conclusions

In this chapter, we noted that for some logical topologies a survivable embedding might not exist in a given physical topology. By adding new logical links or phys-

Number of Survivable Logical-Physical Topology Pairs						
	Physical degree = 3 Logical degree = 2.5			Physical degree = 3 Logical degree = 3		
	No. of Physical Nodes $\lvert V\rvert$					
METHOD	100	200	300	100	200	300
SMART-H	392	390	430	835	855	785
SMART-H + 1	666	695	725	925	965	950
HA-1	1000	1000	1000	1000	1000	1000
HA-2	1000	1000	1000	1000	1000	1000
HA-3	1000	1000	1000	1000	1000	1000
HA-4	1000	1000	1000	1000	1000	1000

Table 5.1: Number of Survivable Logical-Physical Topology Pairs.

Average Number of Protected Logical links						
	Physical degree = 3 Logical degree = 2.5			Physical degree = 3 Logical degree = 3		
	No. of Physical Nodes $\lvert V\rvert$					
METHOD	100	200	300	100	200	300
SMART-H	–	–	–	–	–	–
SMART-H + 1	1	1	1	1	1	1
HA-1	7	7.1	6.2	6.3	6.5	5.5
HA-2	6.3	6.1	4	3	3.2	2.1
HA-3	1.9	1.8	1.7	1.54	1.3	1.3
HA-4	2.3	2	1.9	1.54	1.4	1.3

Table 5.2: Average Number of Protected Logical links.

Average Execution Time (Seconds)						
	Physical degree = 3 Logical degree = 2.5			Physical degree = 3 Logical degree = 3		
	No. of Physical Nodes $\lvert V\rvert$					
METHOD	100	200	300	100	200	300
SMART-H	18	41	54	3.16	5.6	12.38
SMART-H + 1	25	53	71	4.20	6.5	14.30
HA-1	2.3	5.8	8	1.65	2.7	5.30
HA-2	1.4	4.1	6.8	1.58	3.5	5
HA-3	50	107	137	7.60	12.2	27.70
HA-4	41	95	117	6.90	11	25.6

Table 5.3: Average Execution Time (Seconds).

ical links, an unsurvivable logical-physical topology pair could be converted to a survivable one. Adding physical links by laying new optical fibers is difficult and expensive to do. However, logical links can be easily added by establishing additional IP connections. Therefore, we suggested using a combination of protection and restoration approaches to guarantee survivability for any given logical-physical topology pair, which is at least 2-edge connected. Protection was provided by setting up two lightpaths between certain pairs of IP routers such that the lightpaths did not share a physical link. We also provided several heuristics to minimize the number of lightpaths that must be added to make a logical-physical topology pair survivable.

In the next chapter, we will explore the role that duality between circuit based approach (SMART) and cutset based approach (Modiano and Narula-Tam) can play in realizing survivable IP-over-WDM networks. Using the duality results, we will develop several new algorithms and insightful results. We will also present a logical topology structure that can always be embedded in a physical topology in a survivable manner.

Chapter 6

Circuits/Cutsets Duality and a Unified Algorithmic Framework for Survivable Logical Topology Design in IP-over-WDM Optical Networks

Duality between circuits (cycles) and cuts in a graph is one of the well studied topics in graph theory. This concept has played a significant role in the development of methodologies for solving problems in various applications. Most of the early results in electrical circuit theory were founded on the duality relationship between circuits and cuts [31]. There is a wealth of literature on the role of duality in network optimization (that is, discrete optimization on graphs and networks) [25]. Most often, for a primal algorithm based on circuits there is a dual algorithm based on cuts for the same problem. The primal and dual algorithms possess certain characteristics that make one superior to the other depending on the application.

SMART algorithm for the survivable logical topology design problem is based on circuits [2]. The question then arises whether there exists a dual methodology based on cuts. In this chapter we establish such an algorithm and variants of this algorithm that are computationally very efficient. Our work also provides much insight into the structure of solutions for the survivable topology design problem and survivable networks.

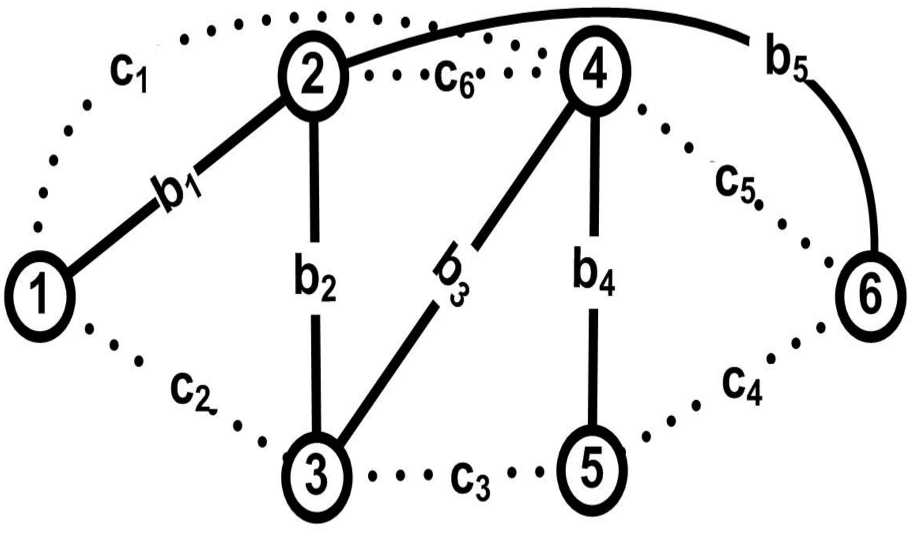
6.1 Circuits and Cutsets Duality

Duality between circuits and cuts in a graph has been extensively studied and plays a fundamental role in several applications [25] [31]. Deleting an edge and contracting an edge are also dual operations. In this section, we present several concepts and results relating to this duality. These results provide the basis for the algorithmic frameworks presented in the following sections.

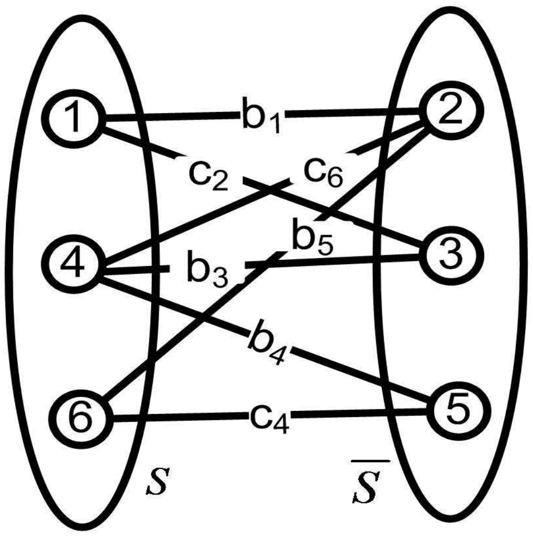
Consider a connected undirected graph $G(V, E)$ with vertex set V and edge set E. Without loss of generality, we assume that there are no parallel edges or self loops in G. Let G have $|V| = n$ vertices (or nodes) and $|E| = m$ edges (or links).

A connected acyclic subgraph of G containing all the n nodes is called a *spanning tree* T of G. The edges of a spanning tree T are called *branches* of T. The remaining edges of G are called *chords* with respect to T. We may also refer to chords as *non-tree edges*.

Consider a partition $(S, \bar{S})$ of vertex set V. Here $\bar{S}$ denotes the complement of S ($S \subseteq V$) in V, i.e. $\bar{S} = V - S$. Then the set of edges with one node in S and the other in $\bar{S}$ is called a *cut* of G. For example, consider the graph G in Fig. 6.1(a). Here the vertices are numbered $1, 2, ..., 6$. The bold edges in this figure denote the branches of a spanning tree T of G and the dotted edges are the chords of this tree. The partition $(S, \bar{S})$ with $S = \{1, 4, 6\}$ and $\bar{S} = \{2, 3, 5\}$ defines the cut shown in



(a) A graph with a spanning tree (bold lines).



(b) A cut.

Figure 6.1: Concept of a tree and a cut.

Fig. 6.1(b).

Adding a chord c to a spanning tree T produces exactly one circuit. This is called the *fundamental circuit* (in short, *f-circuit*) of T with respect to the chord c. We denote this circuit as $B(c)$. For example, if we add chord c_1 to the tree in Fig. 6.1(a) we get the fundamental circuit $B(c_1)$ consisting of the edges $\{c_1, b_1, b_2, b_3\}$. Similarly, if we add chord c_4 to the tree in Fig. 6.1(a) we get the fundamental circuit $B(c_4)$ that contains edges $\{c_4, b_3, b_4, b_5\}$.

Suppose we remove a branch b from a spanning tree T, then the tree T gets disconnected resulting in two trees (not spanning) T_1 and T_2. The sets of nodes in T_1 and T_2 define a partition of V. The corresponding cut is called the *fundamental cutset* (in short, *f-cutset*) of T with respect to branch b. For example, if we remove the branch b_3 from the tree T of Fig. 6.1(a) then we get two trees T_1 and T_2 given by branches $\{b_1, b_2, b_5\}$ and $\{b_4\}$, respectively. The corresponding fundamental cutset $Q(b_3)$ consists of the edges $\{b_3, c_3, c_4, c_5, c_6\}$. Note that the subgraphs induced by the vertex sets of T_1 and T_2 are both connected. Cuts with this property are also called *primary cuts* [17].

Given a spanning tree T with branches $\{b_1, b_2, ..., b_{n-1}\}$ and chords $\{c_1, c_2, ...,$ $c_{m-n+1}\}$, then the *fundamental circuit matrix* $B_f = [b_{ij}]_{(m-n+1)\times(m)}$ has one row for each chord and one column for each edge. The entry b_{ij} is defined as follows

$$\begin{aligned} b_{ij} &= 1, \text{ if } B(c_i) \text{ contains edge } j \\ &= 0, \text{ otherwise.} \end{aligned}$$

Arranging the rows of B_f such that the j^{th} row ($j \leq m-n+1$) corresponds to the fundamental circuit $B(c_j)$ and arranging the columns in the order $\{c_1, c_2, ..., c_{m-n+1},$ $b_1, b_2, ..., b_{n-1}\}$, we can write the B_f matrix as $B_f = [U|B_{ft}]$, where U is the unit matrix of size $(m-n+1)$. For example, the B_f matrix with respect to the spanning tree T of Fig. 6.1(a) is given in (1).

In a similar manner the *fundamental cutset matrix* with respect to the tree T can be defined as $Q_f = [q_{ij}]_{(n-1)\times(m)}$. Q_f has $(n-1)$ rows, one for each fundamental cutset and one column for each edge. The entry q_{ij} is defined as

$$\begin{aligned} q_{ij} &= 1, \text{ if } Q(b_i) \text{ contains edge } j \\ &= 0, \text{ otherwise.} \end{aligned}$$

$$\begin{array}{c|cccccc|ccccc|}
 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & b_1 & b_2 & b_3 & b_4 & b_5 \\
c_1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
c_2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
c_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
c_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
c_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
c_6 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0
\end{array} \text{-- (1)}$$

Arranging the rows of Q_f such that the j^{th} row corresponds to f-cutset $Q(b_j)$ and the columns correspond to edges in the order $\{b_1, b_2, ..., b_{n-1}, c_1, c_2, ..., c_{m-n+1}\}$, the Q_f matrix can be written as $Q_f = [U|Q_{fc}]$. For example, the Q_f matrix with respect to the tree T of Fig. 6.1(a) is given in (2).

$$\begin{array}{c|ccccc|cccccc|}
 & b_1 & b_2 & b_3 & b_4 & b_5 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\
b_1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
b_2 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
b_3 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
b_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
b_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0
\end{array} \text{--(2)}$$

A *subgraph* (for example, a circuit or a cut) can be represented by a binary vector with m entries, one entry for each edge, and with an entry equal to 1 if the corresponding edge is present in the subgraph. Thus, rows of B_f and Q_f are the binary vectors representing the fundamental circuit and fundamental cutsets. For convenience, in the following we will use the same symbol $B(c_j)(Q(b_j))$ to denote the set of edges in a fundamental circuit (cutset) as well as the corresponding binary vectors.

Proofs of the following dual results can be found in [31].

Theorem 6.1. *(a) If a cut contains the branches $\{b_1, b_2, ..., b_j\}$ then the corresponding cut vector can be represented as modulo 2 addition of the vectors $Q(b_1), Q(b_2), ...,$ $Q(b_j)$. That is, the cut vector is equal to $Q(b_1) \oplus Q(b_2) \oplus ... \oplus Q(b_j)$.*

(b) If a circuit contains the chords $\{c_1, c_2, ..., c_j\}$ then the corresponding circuit vector can be represented as modulo 2 addition of the vectors $B(c_1), B(c_2), ...,$ $B(c_j)$. That is, the circuit vector is equal to $B(c_1) \oplus B(c_2) \oplus ... \oplus B(c_j)$. □

Theorem 6.2. *(Orthogonality): A circuit and a cut have an even number of common edges.* □

Theorem 6.3. $B_{ft} = Q^t_{fc}$, *where* Q^t_{fc} *is the transpose of* Q_{fc}. □

The above properties can be verified using the B_f and Q_f matrices given in equation (1) and (2).

An ordered sequence $B(c_1), B(c_2), ..., B(c_k)$ is a *circuit cover sequence* or simply a *B-sequence* of length k if

$$\text{a)}[B(c_j) - c_j - \bigcup_{p=1}^{j-1} B(c_p)] \neq \emptyset, \quad 2 \leq j \leq k$$

$$\text{b)} \bigcup_{p=1}^{k} B(c_p) = E - \{\text{chords not in the } B\text{-sequence}\}$$

Note that for a given spanning tree and its f-circuits, there may be more than one B-sequence. For example for the fundamental circuits given in (1), following are the three B-sequences:

(1) $B(c_1), B(c_3), B(c_5)$

(2) $B(c_4), B(c_1)$

(3) $B(c_6), B(c_1), B(c_4)$

Note that the order in which the $B(c_j)$'s appear matters in the definition of B-sequences. Without the loss of generality assume that $B(c_1), B(c_2), ..., B(c_k)$ is a B-sequence of length k. Let us define $S(c_j)$ as follows:

a) $S(c_1) = B(c_1) - c_1$

b) $S(c_j) = B(c_j) - c_j - \bigcup_{p=1}^{j-1} B(c_p), \quad 2 \leq j \leq k$

Then the submatrix of the f-circuit comprised of the rows corresponding to $B(c_1), B(c_2), ..., B(c_k)$ will have the structure shown in (3).

An ordered sequence $Q(b_1), Q(b_2), ..., Q(b_k)$ is a *cutset cover sequence* or simply a *Q-sequence* of length k if

a)$[Q(b_j) - b_j - \bigcup_{p=1}^{j-1} Q(b_p)] \neq \emptyset, \quad 2 \leq j \leq k$

b) $\bigcup_{p=1}^{k} Q(b_p) = E - \{\text{branches not in the } Q\text{-sequence}\}$

Note that for a given spanning tree and its f-cutsets, there may be more than one Q-sequence. For example for the fundamental cutsets given in (2), following are the three Q-sequences.

(1) $Q(b_4), Q(b_5), Q(b_3)$

(2) $Q(b_4), Q(b_5), Q(b_1), Q(b_2)$

(3) $Q(b_1), Q(b_2), Q(b_4)$

Without the loss of generality assume that $Q(b_1), Q(b_2), ..., Q(b_k)$ is a Q-sequence of length k. Let us define $\hat{S}(b_j)$ as follows:

a) $\hat{S}(b_1) = Q(b_1) - b_1$

b) $\hat{S}(b_j) = Q(b_j) - b_j - \bigcup_{p=1}^{j-1} Q(b_p), \quad 2 \le j \le k$

Then the submatrix of the f-cutset comprised of the rows corresponding to $Q(b_1), Q(b_2), ..., Q(b_k)$ has a structure similar to (3) as shown in (4).

The following dual results are a consequence of the structures in (3) and (4).

$$\begin{array}{c|ccccc|cccc|cccc|cccc|cccc|cccc|}
 & \multicolumn{5}{c}{c_1 \;\ldots\ldots\; c_j \;\ldots\ldots\; c_k} & \multicolumn{4}{c}{S(c_1)} & \multicolumn{4}{c}{S(c_2)} & \multicolumn{4}{c}{\ldots\ldots\ldots} & \multicolumn{4}{c}{S(c_j)} & \multicolumn{4}{c}{S(c_k)} \\
c_1 & 1 & 0 & .. & 0 & .. & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
c_2 & 0 & 1 & .. & 0 & .. & 0 & \times & \times & \times & \times & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
c_j & 0 & 0 & .. & 1 & .. & 0 & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
c_k & 0 & 0 & 0 & 0 & .. & 1 & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & 1 & 1 & 1 & 1
\end{array} \quad \ldots(3)$$

In the above $\times$ means 0 or 1.

$$\begin{array}{c|ccccc|cccc|cccc|cccc|cccc|cccc|}
 & \multicolumn{5}{c}{b_1 \;\ldots\ldots\; b_j \;\ldots\ldots\; b_k} & \multicolumn{4}{c}{\hat{S}(b_1)} & \multicolumn{4}{c}{\hat{S}(b_2)} & \multicolumn{4}{c}{\ldots\ldots\ldots} & \multicolumn{4}{c}{\hat{S}(b_j)} & \multicolumn{4}{c}{\hat{S}(b_k)} \\
b_1 & 1 & 0 & .. & 0 & .. & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
b_2 & 0 & 1 & .. & 0 & .. & 0 & \times & \times & \times & \times & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
b_j & 0 & 0 & .. & 1 & .. & 0 & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
b_k & 0 & 0 & 0 & 0 & .. & 1 & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & \times & 1 & 1 & 1 & 1
\end{array} \quad \ldots(4)$$

In the above $\times$ means 0 or 1.

Theorem 6.4. *(a) Given a B-sequence $B(c_1), B(c_2), ..., B(c_k)$, let $B(c_{i_1})$, $B(c_{i_2}), ..., B(c_{i_l})$ be a subsequence of this sequence then $S(c_{i_l}) \subseteq B(c_{i_1}) \oplus B(c_{i_2})$ $\oplus .... \oplus B(c_{i_l})$.*

(b) Given a Q-sequence $Q(b_1), Q(b_2), ..., Q(b_k)$, let $Q(b_{i_1}), Q(b_{i_2}), ..., Q(b_{i_l})$ be a subsequence of this sequence then $\hat{S}(b_{i_l}) \subseteq Q(b_{i_1}) \oplus Q(b_{i_2}) \oplus .... \oplus Q(b_{i_l})$. □

Deletion of an edge and *contraction* of an edge are dual operations. Here by contraction of an edge we refer to the operation of identifying the end vertices of the edge (short-circuiting the end vertices) and removing self loops that result from this short-circuiting.

Given a B-sequence, the submatrix of the B_f matrix that results after removing the rows that do not correspond to the chords in the B-sequence is called a *B-sequence matrix.* For example the B-sequence matrix corresponding to the B-sequence $B(c_1), B(c_3), B(c_5)$ is shown in (5).

It can be shown that deletion of a row from B_f matrix corresponds to the deletion of the corresponding chord from the graph.

Given a Q-sequence, the submatrix of the Q_f matrix that results after removing the rows that do not correspond to the branches in the Q-sequence is called a *Q-sequence matrix.* For example the Q-sequence matrix corresponding to the Q-sequence $Q(b_4), Q(b_5), Q(b_2), Q(b_1)$ is shown in (6).

$$\begin{array}{c} \begin{array}{cccccccc} c_1 & c_3 & c_5 & b_1 & b_2 & b_3 & b_4 & b_5 \end{array} \\ \left[\begin{array}{ccc|ccccc} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{array}\right] \end{array} --(5)$$

It can be shown that deletion of a row from the Q_f matrix corresponds to contraction of the corresponding branch from the graph.

The following dual results are easy to verify.

Theorem 6.5. *a) The B-sequence matrix corresponding to a B-sequence is the*

$$\begin{array}{cccc|cccccc} b_4 & b_5 & b_2 & b_1 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ \end{array}$$
$$\left[\begin{array}{cccc|cccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right] \text{--(6)}$$

fundamental circuit matrix of the graph that results after deleting the chords that do not appear in the B-sequence.

b) The Q-sequence matrix corresponding to a Q-sequence is the fundamental cutset matrix of the graph that results after contracting the branches that do not appear in the Q-sequence. □

The *incidence set* of a vertex v in a graph is the set of vertices incident on that vertex. The incidence set of vertex v will be denoted by $INC(v)$. Each incident set is a cut of the graph. It is known that any set of $n-1$ incidence sets can be used to generate any cut in a graph. The *incident vector* $INC(v)$ of a vertex v is the binary vector of m entries with j^{th} entry being 1, if the j^{th} edge is in the corresponding incident set. The following result is a special case of Theorem 6.1(a). Note that we use $INC(v)$ to denote the incidence set and the corresponding binary vector. *Incidence matrix* of G is the matrix of incidence vectors of G.

Theorem 6.6. *The cut vector corresponding to the cut $(S, \bar{S})$ can be obtained as the modulo 2 addition of the incident vectors of the vertices in S as well as the modulo 2 addition of the incidence vectors of the vertices in $\bar{S}$.* □

A sequence of $INC(v_1), INC(v_2), ..., INC(v_k)$ is an *incident cover sequence* or simply an *INC-sequence* of length k if

a) $[INC(v_j) - \bigcup_{p=1}^{j-1} INC(v_p)] \neq \emptyset, \quad 2 \leq j \leq k$

b) $\bigcup_{p=1}^{k} INC(v_p) = E$

Given an INC-sequence, the submatrix of the incidence matrix consisting of the rows corresponding to the vertices in the INC-sequence has structure similar to the structure in (4).

The following result will be used in the proof of correctness of all algorithms developed in the following sections. This is also the basis of the algorithmic framework given in [1].

Theorem 6.7. [31]*: A graph is connected if and only if every cut of the graph contains at least one edge.* □

6.2 CIRCUIT-SMART: The Primal Algorithm for Survivable Logical Topology Design

Given a logical topology G_L and physical topology G of an optical network, the SMART algorithmic framework given in [2] provides a methodology for finding survivable mappings of the edges of G_L into lightpaths in G. To begin with, let us call G_L the current graph. In general SMART consists of the following steps.

1. Search for a survivable subgraph of G_L. If no such subgraph is found, then terminate SMART-H unsuccessfully.

2. If a survivable subgraph is found then contract the edges of the subgraph.

3. If the current graph is a single vertex, then terminate SMART-H successfully. Otherwise,return to step 1 and use the contracted graph as the current graph.

The mappings of the edges considered by SMART provide a survivable subgraph of G_L. All the edges that were not mapped by SMART can be mapped arbitrarily without affecting the survivability of G_L. It can be shown that the subgraph chosen in step 1 above must be 2-edge connected for the correctness of the algorithm. Since a circuit is the smallest 2-edge connected graph, usually in step 1 a circuit is selected for survivable mapping.

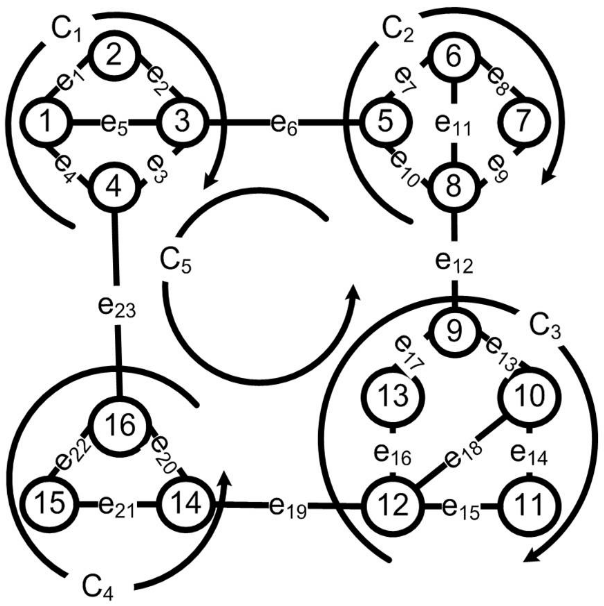


Figure 6.2: Illustration of SMART.

Now consider the graph G_L in Fig. 6.2. If the circuits are selected in the sequence C_1, C_2, C_3, C_4 and $\{e_6, e_{10}, e_{12}, e_{17}, e_{16}, e_{19}, e_{20}, e_{23}, e_3\}$ then these are fundamental circuits of the tree for which $\{e_1, e_8, e_{21}, e_{19}, e_{15}, e_6\}$ are chords. This observation is true for any choice of circuits selected and mapped by SMART. So our algorithm CIRCUIT-SMART starts with a set of fundamental circuits and a B-sequence constructed from these circuits. Since our interest is to guarantee survivability we add to G_L new edges in parallel to some of the edges in G_L whenever necessary. The new edge added in parallel to edge e of G_L will be denoted as e'. Both e and e' will be mapped as disjoint lightpaths in G. These edges will be called *protection edges.*

Theorem 6.8. *The graph G'_L of edges mapped by CIRCUIT-SMART forms a survivable logical graph.*

Proof:

We prove that G'_L is survivable by showing that after failure of any edge in the physical topology each cut in G'_L satisfies the condition in Theorem 6.7. Consider a cut in G'_L. If any edge in this cut is a protection edge e', then this edge and the corresponding edge e in G'_L are mapped by the algorithm into disjoint lightpaths. Hence, one of them will remain in the cut after a single physical edge failure, thereby satisfying the condition in Theorem 6.7. If there is no protection edge in the cut, then consider the branches in the cut. The cut must contain at least one branch of T because chords alone cannot form a cut. Note that each branch is in a unique $S(c_i)$. Let chord c_i be the chord with the smallest index in the B-sequence such that $S(c_i)$ contains one of the branches in the selected cut. If $S(c_i)$ contains two or more branches in the cut, then these branches are mapped by the algorithm into disjoint paths and so the cut will satisfy the condition of Theorem 6.7 after a single link failure. If $S(c_i)$ contains only one branch of the cut, say b, the cut must contain chord c_i because the cut and $B(c_i)$ must contain an even number of edges in common (Theorem 6.2). Since b and c_i are mapped by the algorithm into disjoint paths in the physical topology, one of these two edges will remain in the cut after a single edge failure, satisfying the condition of Theorem 6.7. Thus in all cases, the cut will satisfy the condition of Theorem 6.7 and so the graph G'_L is survivable. □

The essential difference between SMART and CIRCUIT-SMART is that instead of searching for a survivable circuit in each step (as in SMART), CIRCUIT-SMART uses a set of fundamental circuits. Since not all edges in a $S(c_i)$ set can be mapped in a disjoint manner, we add protection edges appropriately. The longer the B-sequence, the more are the number of edges in the survivable logical subgraph. On the other hand a smaller B-sequence may increase the sizes of $S(c_i)$-sets and hence

```
INPUT: A 2-edge connected physical topology $G$, logical topology $G_L$, a spanning tree $T$ of $G_L$, a set of fundamental circuits and a $B$-sequence $B(c_1), B(c_2), ...., B(c_k)$.
OUTPUT: A survivable logical graph $G'_L$ containing $G_L$.
Algorithm CIRCUIT-SMART:
1) For $i = 1, 2, ...., k$ do
     Map a maximum subset of edges in $S(c_k) \cup c_k$ into disjoint lightpaths in $G$.
     To all other edges in $S(c_k) \cup c_k$ add protection edges and map each edge and its protection edges into disjoint lightpaths in $G$.
   End For
2) Map all the chords not in the $B$-sequence into lightpaths in $G$ arbitrarily.
END
```

Figure 6.3: Algorithm CIRCUIT-SMART.

may result in more number of protection edges. These are considerations that must be taken into account while selecting the spanning tree. The algorithm CIRCUIT-SMART is given in Fig. 6.3 and an illustration of CIRCUIT-SMART is given in section 6.7.1.

```
INPUT: A 2-edge connected physical topology $G$, logical topology $G_L$ a spanning tree $T$ of $G_L$, a set of fundamental cutsets and a $Q$-sequence $Q(b_1), Q(b_2), ...., Q(b_k)$.
OUTPUT: A survivable logical graph $G''_L$ containing $G_L$.
Algorithm CUTSET-SMART:
1) For $i = 1, 2, ...., k$ do
     Map a maximum subset of edges in $\hat{S}(b_k) \cup b_k$ into disjoint lightpaths in $G$.
     To all other edges in $\hat{S}(b_k) \cup b_k$ add protection edges and map each edge and its protection edge into disjoint lightpaths in $G$.
   End For
2) To each unmatched branch $b$ add a protection edge $b'$ and map them into disjoint lightpaths in $G$.
END
```

Figure 6.4: Algorithm CUTSET-SMART.

6.3 CUTSET-SMART: The Dual Algorithm

We now present algorithm CUTSET-SMART that is the dual of algorithm CIRCUIT-SMART. In the following a branch is *unmatched* if it is not in the given Q-sequence.

Let G''_L be the graph of logical edges (including protection edges) mapped by CUTSET-SMART.

Theorem 6.9. *a) The graph G''_L and the mappings generated by CUTSET-SMART are survivable.*

b) The graph obtained from G''_L by contracting the branches not in the Q-sequence is survivable.

Proof:

a) Consider a cut in G''_L. If any edge in this cut is a protection edge then this edge and the corresponding edge in G_L are mapped by the algorithm into disjoint lightpaths. Hence one of them will remain in the cut after a single physical edge failure, thereby satisfying the condition in Theorem 6.7. If there is no protection edge in the cut, then consider the branch b_j in the cut that has the highest index in the Q-sequence. Then by Theorem 6.4(b), the set $\hat{S}(b_j)$ will be in the cut. Since the branch b_j and the edges in the set $\hat{S}(b_j)$ are mapped in disjoint manner by the algorithm, the cut will contain at least one edge after a physical edge failure, thereby satisfying the condition of Theorem 6.7. Thus G''_L is survivable.

b) The graph obtained from G''_L by contracting the branches not in the Q-sequence has no cut that contains the contracted tree branches and the corresponding protection edges. The result follows from the earlier part of the proof of (a). □

Note that Theorem 6.9(b) is the dual of the result that graph G'_L generated by CIRCUIT-SMART is survivable. The algorithm CUTSET-SMART is shown in Fig. 6.4.

A closer look at the above proof will show that in step 1 of CUTSET-SMART, it is sufficient to map in disjoint manner each branch b_i with some chord in the set $\hat{S}(b_i)$. This results in algorithm CUTSET_SMART_SIMPLIFIED shown in Fig. 6.5.

The above algorithm requires finding disjoint mappings for only certain pairs of vertices in the physical topology. Using a result in [39] we have the following Theorem.

```
INPUT: A 2-edge connected physical topology $G$, logical
topology $G_L$, a spanning tree $T$ of $G_L$, a set of fundamental
cutsets and a $Q$-sequence $Q(b_1), Q(b_2), ...., Q(b_k)$.
OUTPUT: A survivable logical graph $G''_L$ containing $G_L$.
Algorithm CUTSET_SMART_SIMPLIFIED:
1) For $i = 1, 2, ...., k$ do
     Map $b_i$ in disjoint manner with some chord in set $\hat{S}(b_i)$.
     If this is not possible for any chord in $\hat{S}(b_i)$ then
       Add a protection edge for one of the chord and map the chord and
       its protection edge in disjoint manner.
     End If
   End For
2) To each unmatched branch $b$ add a protection edge $b'$ and map them
   as disjoint lightpaths in $G$.
3) Map all the unmapped logical edges arbitrarily.
END
```

Figure 6.5: Algorithm CUTSET_SMART_SIMPLIFIED.

Theorem 6.10. *Given any Q-sequence of length k, algorithm CUTSET_SMART_SIMPLIFIED finds a survivable mapping of a logical topology with at most $n - k - 1$ protection edges, if the physical topology is 3-edge connected.*
□

The main reason for the above result is that any pair of edges $\{(s_1, t_1), (s_2, t_2)\}$ in G_L can be mapped into disjoint lightpaths, if the physical topology is 3-edge connected.

The proof of the Theorem 6.9 shows that every cut obtained from G''_L by contracting the branches not in the Q-sequence will contain at least $Min\{|\hat{S}(b_i)|+1, i =$

$1, 2, ..., k\}$ edges after a single edge failure in the physical topology. This property of CUTSET-SMART will be of great help in protecting the logical topology when multiple edge failures occur in the physical topology. This property is not true in the case of algorithm CUTSET_SMART_SIMPLIFIED though it is computationally superior to CUTSET-SMART and is likely to require less number of protection edges. An interesting consequence of algorithm CUTSET_SMART_SIMPLIFIED is the following.

Theorem 6.11. *The structure shown in Fig. 6.6 is survivable, if the physical topology is 3-edge connected.* □

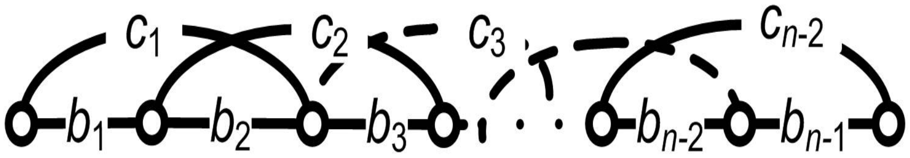


Figure 6.6: A survivable network structure.

An interesting application of this result is as follows. Note that protection edges are used in the algorithms of sections 6.2, 6.3 and 6.4 to guarantee survivability. Consider a set of edges that form a path and require protection edges, then it can be shown that we can add new logical edges $c_1, c_2, ..., ..., ...$ as in Fig. 6.6, instead of protection edges (that is, new parallel logical edges) and guarantee survivability.

Note that we have not been able to prove a property similar to the one in Theorem 6.10 in the case of CIRCUIT-SMART. Nor has it been possible for us to find a simplified version of CIRCUIT-SMART akin to CUTSET_SMART_SIMPLIFIED (shown in Fig. 6.5). An illustration of CUTSET_SMART_SIMPLIFIED is provided in section 6.7.2.

```
In this method we replace step (2) of CUTSET-SMART and
CUTSET_SMART_SIMPLIFIED by the following:
"Apply CIRCUIT-SMART on the graph obtained from $G_L$ by contract-
ing the matched branches (branches in the $Q$-sequence)."
```

Figure 6.7: A 2-Phase method.

6.4 Efficient Heuristics to Minimize Number of Protection Edges

In cutset based algorithms, to guarantee survivability we add protection edges in parallel to branches (unmatched branches) that are not in the Q-sequence. See step (2) in these algorithms. To minimize the number of such protection edges we suggest two heuristics shown in Fig. 6.7 and Fig. 6.8.

```
1) For $i = 1, 2, ...., k$ do
     Map $b_i$ in disjoint manner with some chord in set $\hat{S}(b_i)$
   End For
2) Construct a bipartite graph $(X, Y)$ where each vertex in $X$ represents
an unmatched branch and each vertex in $Y$ represents an unmatched
chord. The bipartite graph has an edge $(x, y)$ if the fundamental cutest
with respect to branch $x$ contains the chord $y$.
3) Find a maximum matching in this bipartite graph using any
standard maximum matching algorithm [25].
4) For each edge in the maximum matching, find disjoint mappings
for the corresponding pair of branch and chord.
5) For all unmatched branches (those not matched in step 1 or 2) add
protection edges as in step (2) of CUTSET-SMART.
END
```

Figure 6.8: Maximum matching based heuristic.

We have not been able to show that these heuristics guarantee survivability in certain scenarios. However, we expect them to guarantee survivability except in this case of certain combinations of logical edge failures.

6.5 INCIDENCE-SMART

Incident sets are special cases of cuts. So an algorithm similar to CUTSET-SMART can be designed. Instead of explicitly starting with a spanning start tree with some new logical edges, we present an algorithm which reflects the unified framework in terms of INC-sets defined in section 6.1. Note that for any INC-sequence there is at least one vertex that is not in the sequence. Let us call one such vertex as *datum*.

In the following algorithm (Fig. 6.8) the given G_L will be the initial current graph.

Theorem 6.12. *Algorithm INCIDENCE-SMART provides a survivable mapping of the edges of a graph G''_L that contains the given logical graph G_L.*

Proof:

Consider any cut $(S, \bar{S})$ in G_L. Let S be the partition of the cut that does not contain the datum vertex. Consider the vertex v in S that has the highest index in the INC-sequence. Then in the current graph at the step when v is considered by the algorithm it will not be adjacent to any vertex in S. So, according to the algorithm v will be connected to at least two vertices in $\bar{S}$, and the corresponding edges connecting S and $\bar{S}$ are mapped into disjoint lightpaths, guaranteeing that at least one of these edges will remain in the cut after a single edge failure in G and satisfying the condition of Theorem 6.7. Since this is true for all cuts, the mappings generated by the algorithm are survivable. □

An example of INCIDENCE-SMART is given in section 6.7.3.

```
INPUT: A 2-edge connected physical topology $G$, a logical topology $G_L$, $INC$-sequence $INC(v_1), INC(v_2), ...., INC(v_k)$.
OUTPUT: A survivable logical graph $G''_L$ containing $G_L$.
Algorithm INCIDENCE-SMART:
For $i = 1, 2, ...., k$ do
1) If vertex $v_i$ has degree greater than or equal to 2 in the current
   graph then
     Map all the edges incident on $v_i$ into disjoint lightpaths in $G$.
2) If the degree of $v_i$ in the current graph is one then
     Add a new logical edge connecting $v_i$ to the datum vertex.
     Map this new edge and the only edge incident on $v_i$ into disjoint
     lightpaths.
3) If the degree of $v_i$ in the current graph is zero then
     Add two new parallel logical edges connecting $v_i$ to the datum vertex.
     Map these two edges into disjoint lightpaths in $G$.
End For
END
```

Figure 6.9: Algorithm INCIDENCE-SMART.

6.6 Illustration of CIRCUIT-SMART, CUTSET_SMART_SIMPLIFIED and INCIDENCE-SMART

In this section we provide examples that illustrate algorithms CIRCUIT-SMART, CUTSET_SMART_SIMPLIFIED and INCIDENCE-SMART. The topology shown in Fig. 6.10 is used as the logical topology (G_L) for the CIRCUIT-SMART and CUTSET_SMART_SIMPLIFIED. A spanning tree T of G_L consists of edges (branches) b_1, b_2, b_3 and b_4 and the chords for T are given by edges c_1, c_2, c_3, c_4, c_5 and c_6. Furthermore, for CIRCUIT-SMART, it is assumed that an arbitrary at least 2-edge connected physical topology is given and for CUTSET_SMART_SIMPLIFIED a 3-edge connected topology is given.

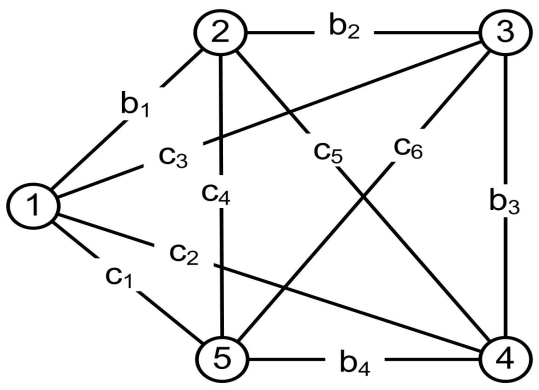


Figure 6.10: Logical topology for the illustration of CIRCUIT-SMART and CUT-SET_SMART_SIMPLIFIED.

$$\begin{array}{c|cccccc|cccc} & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & b_1 & b_2 & b_3 & b_4 \\ \hline c_1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ c_2 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ c_3 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ c_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ c_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ c_6 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array} \quad \text{--(7)}$$

6.6.1 Illustration of CIRCUIT-SMART

The fundamental circuits with respect to the spanning tree T are as follows.

1) $B(c_1) = \{c_1, b_1, b_2, b_3, b_4\}$

2) $B(c_2) = \{c_2, b_1, b_2, b_3\}$

3) $B(c_3) = \{c_3, b_1, b_2\}$

4) $B(c_4) = \{c_4, b_2, b_3, b_4\}$

5) $B(c_5) = \{c_5, b_2, b_3\}$

6) $B(c_6) = \{c_6, b_3, b_4\}$

The fundamental circuit matrix (B_f) corresponding to T is shown in (7). Now, assume that the given circuit cover sequence (B-sequence) is $B(c_5), B(c_3), B(c_6)$ of length $k = 3$, then

$$
\begin{aligned}
S(c_5) &= B(c_5) - c_5 \\
&= \{c_5, b_2, b_3\} - \{c_5\} \\
&= \{b_2, b_3\}
\end{aligned}
$$

Now for

$j = 2$:

$$
\begin{aligned}
S(c_3) &= B(c_3) - c_3 - \bigcup_{p=1}^{1} B(c_p) \\
&= \{c_3, b_1, b_2\} - \{c_3\} - \{c_5, b_2, b_3\} \\
&= \{b_1\}
\end{aligned}
$$

$j = 3$:

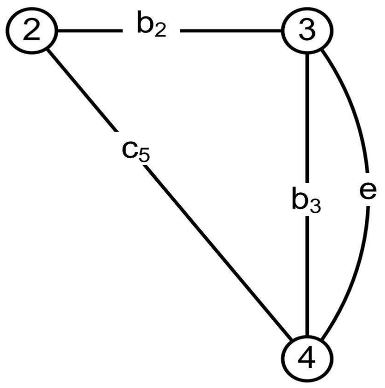
$$
\begin{aligned}
S(c_6) &= B(c_6) - c_6 - \bigcup_{p=1}^{2} B(c_p) \\
&= \{c_6, b_3, b_4\} - \{c_6\} - [\{c_5, b_2, b_3\} \cup \{c_3, b_1, b_2\}] \\
&= \{b_4\}
\end{aligned}
$$

Now $S(c)$ corresponding to the B-sequence $B(c_5), B(c_3), B(c_6)$ is given in (8).

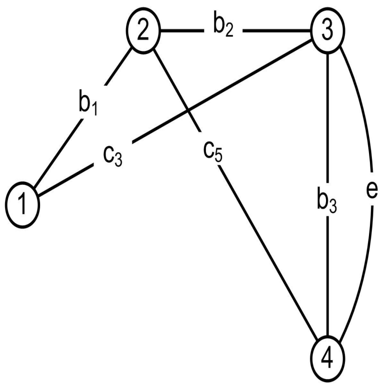
$$
\begin{array}{ccc|cc|c|c}
c_5 & c_3 & c_6 & \multicolumn{2}{c|}{S(c_5)} & S(c_3) & S(c_6) \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1
\end{array} \quad --(8)
$$

After obtaining the B_f matrix and $S(c)$ corresponding to the B-sequence $B(c_5)$, $B(c_3), B(c_6)$ CIRCUIT-SMART algorithm given in Fig. 6.3 is applied. In the first iteration, $S(c_5) \cup c_5 = \{c_5, b_2, b_3\}$. Therefore, the maximum subset of edges in $S(c_5) \cup c_5$ are mapped into disjoint lightpaths. Assume that all the edge in $S(c_5) \cup c_5$ could not be mapped into disjoint lightpaths and a protection edge e is provided for the logical edge b_3 (Fig. 6.11(a)). In the next iteration, $S(c_3) \cup c_3 = \{c_3, b_1\}$ and assume that all the edges can be mapped into disjoint lightpaths (Fig. 6.11(b)). In the last iteration, $S(c_6) \cup c_6 = \{c_6, b_4\}$ and again assume that all the edges can mapped into disjoint lightpaths (Fig. 6.11(c)). Chords c_1, c_2 and c_4 are not in the B-sequence and are mapped arbitrarily (not shown in Fig. 6.11).

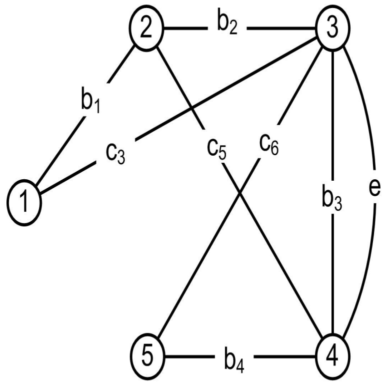
Fig. 6.11(c) shows the mapped chords and their corresponding branches. Now if a physical failure disconnects some the logical connections, the logical topology would still stay connected. Fig. 6.11(d) shows the case where a physical link failure disconnects logical edges b_2, b_3, c_3 and c_6 but it can be seen that the logical topology is still connected.



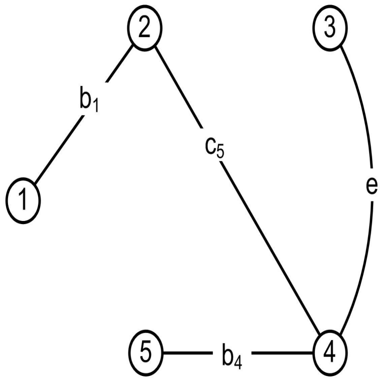
(a) $S(c) = \{c_5, b_2, b_3\}$



(b) $S(c) = \{c_3, b_1\}$



(c) $S(c) = \{c_6, b_4\}$



(d) Logical topology after removal of b_2, b_3, c_3 and c_6

Figure 6.11: Illustration of CIRCUIT-SMART.

6.6.2 Illustration of CUTSET_SMART_SIMPLIFIED

Consider the logical topology shown in Fig. 6.10 again. The fundamental cutsets with respect to the spanning tree T are as follows

1) $Q(b_1) = \{b_1, c_1, c_2, c_3\}$

2) $Q(b_2) = \{b_2, c_1, c_2, c_3, c_4, c_5\}$

3) $Q(b_3) = \{b_3, c_1, c_2, c_4, c_5, c_6\}$

4) $Q(b_4) = \{b_4, c_1, c_4, c_6\}$

The fundamental cutset matrix (Q_f) for spanning T is shown in (9). Now assume that we are given a cutset cover sequence (Q-sequence) $Q(b_1), Q(b_4), Q(b_2)$ of length $k = 3$, then

$$\begin{aligned}\hat{S}(b_1) &= Q(b_1) - b_1 \\ &= \{b_1, c_1, c_2, c_3\} - \{b_1\} \\ &= \{c_1, c_2, c_3\}\end{aligned}$$

Now for

$j = 2$:

$$\begin{aligned}\hat{S}(b_4) &= Q(b_4) - b_4 - \bigcup_{p=1}^{1} Q(b_p) \\ &= \{b_4, c_1, c_4, c_6\} - \{b_4\} - \{b_1, c_1, c_2, c_3\} \\ &= \{c_4, c_6\}\end{aligned}$$

$j = 3$:

$$\begin{aligned}\hat{S}(b_2) &= Q(b_2) - b_2 - \bigcup_{p=1}^{2} Q(b_p) \\ &= \{b_2, c_1, c_2, c_3, c_4, c_5\} - \{b_2\} - [\{b_1, c_1, c_2, c_3\} \cup \{b_4, c_1, c_4, c_6\}] \\ &= \{c_5\}\end{aligned}$$

The $\hat{S}(b)$ corresponding to the Q-sequence $Q(b_1), Q(b_4), Q(b_2)$ is given in (10)

$$\begin{array}{c|cccc|cccccc|} & b_1 & b_2 & b_3 & b_4 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \\ b_1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ b_2 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ b_3 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ b_4 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \text{---(9)}$$

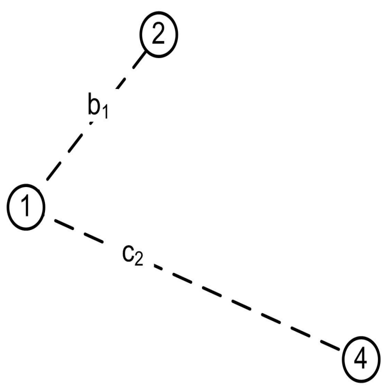
$$\begin{array}{ccc|ccc|cc|c} b_1 & b_4 & b_2 & & \hat{S}(b_1) & & \hat{S}(b_4) & & \hat{S}(b_2) \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \end{array} \text{ --(10)}$$

Now the CUTSET_SMART_SIMPLIFIED algorithm given in Fig. 6.5 is applied by matching branch b_1 with chord c_2 (Fig. 6.12(a)), branch b_4 with chord c_6(Fig. 6.12(b)) and branch b_2 is matched to chord c_5 (Fig. 6.12(c)). After matching a branch with a chord, the branch and the corresponding chord are mapped in disjoint manner. Since branch b_3 could not be matched to any chord, a protection edge e is added for b_3, and then e and b_3 are mapped in disjoint fashion (Fig. 6.12(d)). Fig. 6.12(d) shows all the mapped branches and their corresponding chords. Now if a physical failure disconnects some the logical connections, the logical topology would still stay connected. Fig. 6.12(e) shows the case where a physical link failure disconnects all the branches but it can be seen that the logical topology is still connected.

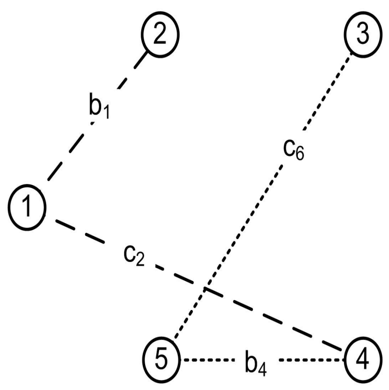
6.6.3 Illustration of INCIDENCE-SMART

As an example of the INCIDENCE-SMART, consider the logical topology shown in Fig. 6.13(a). Assume that the vertex v_4 is chosen as the datum vertex. The incidence sets for Fig. 6.13(a) are shown in (11) and the incidence sequence $INC(v_1), INC(v_2)$, $INC(v_3), INC(v_5)$ is shown in (12). Now the INCIDENCE-SMART is applied to the topology shown in Fig. 6.13(a).

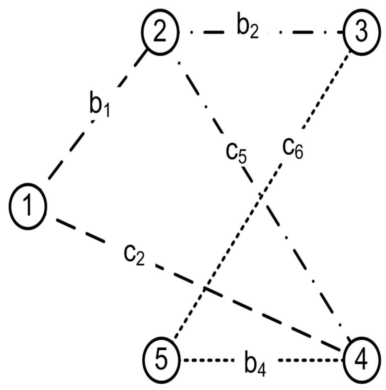
First vertices with degree ≥ 2 are mapped. Edges $\{e_1, e_5,\}$ incident on v_1 are mapped in disjoint manner (Fig. 6.13(b)), then $\{e_2, e_8\}$ (Fig. 6.13(c)) and $\{e_3, e_{10}\}$ (Fig. 6.13(d)) incident on v_2 and v_3, respectively, are mapped in disjoint fashion.



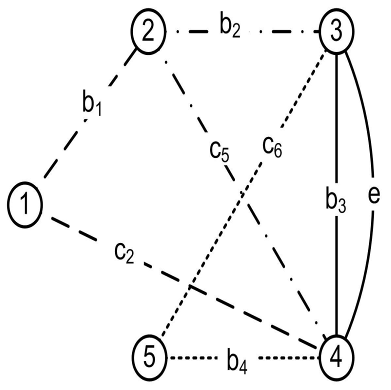
(a) Logical topology after b_1 and c_2 are mapped



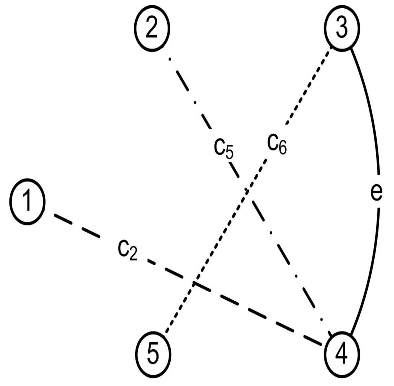
(b) Logical topology after b_4 and c_6 are mapped.



(c) Logical topology after b_2 and c_5 are mapped.



(d) Logical topology after b_3, and e are mapped.



(e) Logical topology after the removal of b_1, b_2, b_3, and b_4.

Figure 6.12: Illustration of CUTSET_SMART_SIMPLIFIED.

After removing all the edges incident on v_1, v_2 and v_3, vertex v_5 has degree 1. Therefore, a new edge e is added between v_5 and the datum node (v_4) (Fig. 6.13(d)). The new edge e and the edge incident on v_5 (e_4) are then mapped in disjoint fashion, providing a survivable mapping for the logical topology given in Fig. 6.13(e).

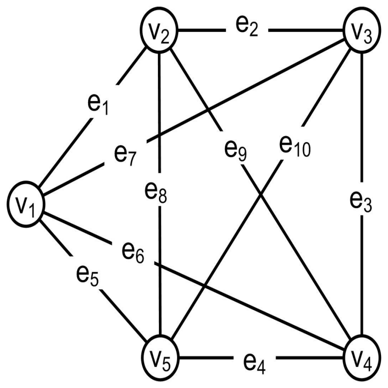
Fig. 6.13(e) shows all the pairs of edges that were mapped in disjoint fashion and Fig. 6.13(f) shows the logical topology after a physical link failure, which disconnects at least one edge from all pairs of edges that were mapped in disjoint manner.

$$\begin{array}{c|cccccccccc|} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} \\ v_1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ v_2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ v_3 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ v_4 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ v_5 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \quad \text{...(11)}$$

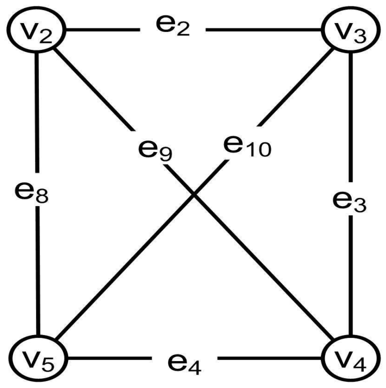
$$\begin{array}{c|cccc|ccc|cc|c|} & e_1 & e_5 & e_6 & e_7 & e_2 & e_8 & e_9 & e_3 & e_{10} & e_4 \\ v_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_2 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ v_5 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{array} \quad \text{...(12)}$$

6.7 Simulation Study and Results

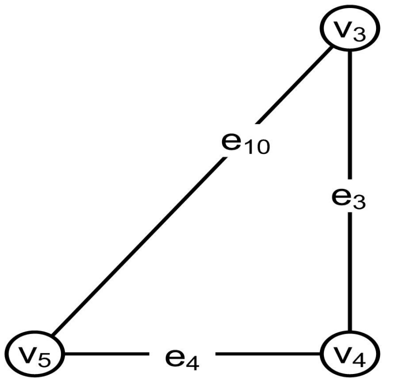
To compare the performance of the proposed algorithms, simulation studies were conducted using VC++ 8.0. For simulation studies, random logical topologies with varying number of nodes and degree were generated using the procedure given in [32]. The physical topologies were regular topologies with degree 4, constructed using a procedure originally given by Harary [30] and described in [31]. The number of nodes in the physical topologies was set to 100, and 200 nodes ($|V|$). The logical topologies were generated randomly with average degrees 2.5, 3.0, 3.5 and 4.0. The



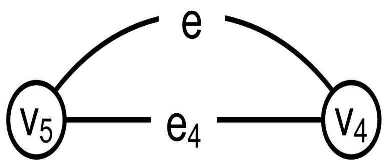
(a) A logical topology for the illustration of INCIDENCE-SMART.



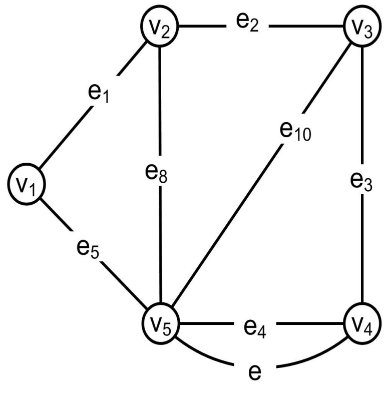
(b) Logical topology after edges incident on v_1 are mapped.



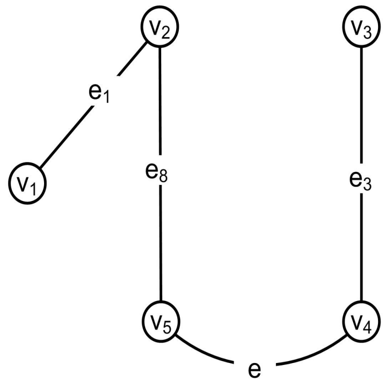
(c) Logical topology after edges incident on v_2 are mapped.



(d) Logical topology after edges incident on v_3 are mapped and a protection edge e is added for v_4.



(e) All the mapped logical edges.



(f) Logical topology after the removal of logical edges e_5, e_2, e_{10} and e_4.

Figure 6.13: Illustration of INCIDENCE-SMART.

nodes in the logical topologies are a subset of the physical nodes and number of logical nodes in the logical topology was set to $0.75 \times |V|$. For each case, 40 physical and 25 logical topologies were generated, providing a total of 1000 logical-physical topology pairs for comparison. The topologies were subjected to further processing, only if the topologies were at least 2-edge connected.

To find the maximum number of logical links that can be mapped in a mutually disjoint manner, a procedure described in [26] was used. To find mutually disjoint mappings of a pair of logical links, the algorithm given in [39] was used. Fundamental circuits and cutsets were found using procedures given in [31] and were part of the preprocessing phase. The survivability of a logical topology was tested by picking a physical link, removing all the logical links which used this physical link in their mapping, and checking if the resulting logical topology is connected. This test was repeated for every physical link.

The statistics of interest were survivability success rate (that is, the number of logical topologies for which survivable mappings could be found), protection capacity (measured as average number of protection links added to a logical topology to make it survivable) and the execution time of the algorithms.

We now make some general observations on the performance of these algorithms based on the trends that we noticed during the simulations.

Tables 6.2- 6.7 summarize the results. Tables 6.2 and 6.5 provide the number of successful survivable logical-physical topology pairs. It can be seen that we were able to find survivable mappings for 100% of the topologies (see Table 6.1 for legend) as expected in the case of $M1, M2, M3$ and $M7$. Also, we would like to point out that in the case of $M4$ and $M6$ we were able to achieve 100% survivability in all the tests but we have not been able to prove that this result holds in general.

Tables 6.3 and 6.6 show a general trend that the number of protection edges required in the case of circuit based methods is considerably less than the number required by cutset based methods. The number of protection edges required by $M4$, which is obtained by integrating $M1$ and $M3$, is less than the number for $M1$. Note that in the case of cutset based methods, for a given value of n the number of edges in a spanning tree does not change. Since $M3$ maps only $n-1$ pairs of edges, the computation time for this method does not change very significantly with a change in average degree.

$M3$ is the best in terms of execution time. $M7$ is the next best. $M7$ also does very well in terms of number of protection edges required.

Summarizing, if protection capacity is an issue of concern, then the two phase algorithm $M4$ is a good choice but it does not guarantee survivability. If computational time is an issue of primary interest, then $M3$ and $M7$ are good choices. Finally, if guarantee of survivability is also an issue of interest then $M7$ is the best choice.

The above work was presented in [40].

6.8 Chapter Summary and Conclusions

Duality plays a significant role in optimization theory, particularly in discrete optimization on graphs and networks. Circuits and cuts in graphs are dual concepts. Similarly, deletion and contraction of an edge are dual graph operations. Most often, for a primal algorithm based on circuits there is a dual algorithm based on cuts for the same problem. The primal and dual algorithms possess certain characteristics that make one superior to the other depending on the application. The SMART algorithm for the survivable logical topology design problem is based on circuits. This is a novel and very significant contribution to the problem. The question then arises whether there exists a dual methodology based on cuts. In this chapter we have investigated this question and developed certain new dual algorithms and insightful results.

First, we reviewed several results that highlight the dual relationship between circuits and cuts and the dual graph operations of deletion and contraction of an edge. We also introduced certain new concepts and results that form the foundation of the methodologies developed in the rest of the chapter. We then presented the primal algorithm CIRCUIT-SMART (similar to SMART) and algorithm CUTSET-SMART that is dual of CIRCUIT-SMART and unified proofs of correctness of these algorithms. Our investigation has provided much insight into the structural properties of solutions to this problem and the structure of survivable logical graphs (Theorems 6.10 and 6.11). Specifically, we presented a highly simplified version of CUTSET-SMART that always provides a survivable mapping as long as the physical topology is 3-edge connected. We presented a logical topology structure that can always be embedded in a survivable manner (Theorem 6.11). We also presented algorithm INCIDENCE-SMART that uses incidence sets. Two efficient heuristics, one based on maximum matching and a 2-phase method that combines both the primal and dual algorithms, were also presented. To guarantee survivability, our algorithms add new logical edges called protection edges, whenever needed. Simulations comparing the different algorithms in terms of computational time, protection capacity and survivability success rate are also provided.

We wish to add that though all our algorithms are formulated in terms of fundamental circuits or fundamental cutsets, they can be presented in a general form as in the original SMART algorithm. Such a general form would require a search of a spanning tree whose fundamental circuits or cutsets can be mapped in a survivable manner. In our simulations, we have not compared our algorithms with the original SMART algorithm for two reasons. First, SMART will require significantly higher execution times because it searches for survivable circuits and so the comparison will not be fair. Secondly, SMART does not add protection edges to guarantee survivability.

As we stated in section 6.8 , if protection capacity is an issue of concern, then the two phase algorithm, CUTSET_SMART_SIMPLIFIED combined with CIRCUIT-SMART, is a good choice but it does not guarantee survivability. If computational time is an issue of primary interest, then CUTSET_SMART_SIMPLIFIED and INCIDENCE-SMART are good choices. Finally, if guarantee of survivability is also an issue of interest then INCIDENCE-SMART is the best choice.

The performance of our algorithms depends on the choice of the spanning tree and the resulting B- and Q- sequences. Research is in progress on approaches for selecting appropriate spanning trees. We believe that our work studying the problem from different perspectives has advanced the state of the art and thrown much insight into the problem.

CIRCUIT-SMART	M1
CUTSET-SMART	M2
CUTSET_SMART_SIMPLIFIED	M3
2 Phase Method (with CUTSET_SMART_SIMPLIFIED)	M4
Maximum-Matching + Parallel protection edges	M5
Maximum-Matching + CIRCUIT-SMART	M6
INCIDENCE-SMART	M7
Logical Degree	LD
Physical Degree	PD
Method Number	M

Table 6.1: Legend

Survivable Logical-Physical Topology Pairs ($\|V\| = 100,\ PD = 4$)				
LD/M	2.5	3.0	3.5	4.0
M1	1000	1000	1000	1000
M2	1000	1000	1000	1000
M3	1000	1000	1000	1000
M4	1000	1000	1000	1000
M5	460	544	654	748
M6	1000	1000	1000	1000
M7	1000	1000	1000	1000

Table 6.2: Survivable Logical-Physical Topology Pairs ($|V| = 100,\ PD = 4$)

Average Number of Protected Links ($\|V\| = 100, PD = 4$)				
LD/M	2.5	3.0	3.5	4.0
M1	335.48	18.35	15.53	11.89
M2	55.94	41.48	32.05	25.51
M3	56.18	41.92	32.37	25.54
M4	18.79	6.01	3.85	2.31
M5	55.22	36.72	18.42	3.85
M6	38.97	3.15	1.23	1.32
M7	56.30	40.81	30.94	22.99

Table 6.3: Average Number of Protected Links ($|V| = 100, PD = 4$)

Time per Logical-Physical Topology-Pair (sec) ($\|V\| = 100, PD = 4$)				
LD/M	2.5	3.0	3.5	4.0
M1	10.013	5.736	5.0071	4.14986
M2	1.5213	1.557	1.5531	1.54455
M3	1.5147	1.549	1.5315	1.53
M4	5.6325	2.404	1.9429	1.64718
M5	1.3824	1.207	1.0690	0.92201
M6	4.1158	0.863	0.7701	0.87369
M7	0.4831	0.400	0.3575	0.3425

Table 6.4: Time per Logical-Physical Topology-Pair (sec) ($|V| = 100, PD = 4$)

Survivable Logical-Physical Topology Pairs ($\|V\| = 200, PD = 4$)				
LD/M	2.5	3.0	3.5	4.0
M1	1000	1000	1000	1000
M2	1000	1000	1000	1000
M3	1000	1000	1000	1000
M4	1000	1000	1000	1000
M5	480	608	701	750
M6	1000	1000	1000	1000
M7	1000	1000	1000	1000

Table 6.5: Survivable Logical-Physical Topology Pairs ($|V| = 200, PD = 4$)

Average Number of Protected Links ($\|V\| = 200, PD = 4$)				
LD/M	2.5	3.0	3.5	4.0
M1	74.39	45.77	35.36	30.95
M2	113.61	86.16	65.79	53.63
M3	113.91	86.37	65.97	53.83
M4	40.53	15.12	9.31	6.66
M5	111.7	75.02	40.28	8.71
M6	77.34	8.91	4.94	3.48
M7	112.80	84.55	61.99	47.99

Table 6.6: Average Number of Protected Links ($|V| = 200, PD = 4$)

Time per Logical-Physical Topology-Pair (sec) ($\|V\| = 200, PD = 4$)				
LD/M	2.5	3.0	3.5	4.0
M1	72.1626	46.2953	38.2611	33.2624
M2	11.5136	11.351	11.3872	11.1478
M3	11.366	11.029	11.1649	11.0306
M4	39.9924	18.7964	14.8348	13.3879
M5	10.4621	8.96543	7.81091	6.59595
M6	27.5417	6.74687	4.96552	6.13079
M7	3.05346	2.59381	2.23091	2.12124

Table 6.7: Time per Logical-Physical Topology-Pair (sec) ($|V| = 200, PD = 4$)

Chapter 7

Summary and Future Work

7.1 Summary

In this dissertation, we have focused on developing efficient algorithms to design survivable IP-over-WDM networks. IP-over-WDM networks are realized by embedding an IP network in a WDM network. The embedding is done by setting up all-optical connections, called *lightpaths*, between the IP routers in the WDM network. An embedding is considered *survivable*, if the IP network remains connected in the presence of a single or multiple failures in the WDM network.

Survivable IP-over-WDM networks are widely touted as the next generation architecture for high speed backbone networks. The problem of finding such embedding can be formulated as an Integer Linear Program (ILP) and solved optimally using commercial or freely available software. However, given the fact that the number of constraints in the ILP grows exponentially with the size of the network, we focus mainly on developing efficient heuristics that find embeddings that leave the IP network connected in the event of a single WDM link failure.

In Chapter 2, we provided a brief overview of the different types of WDM networks and the corresponding survivability mechanisms. Chapter 2 also defined IP-over-WDM networks, survivable IP-over-WDM networks and discussed some of the issues involved in realizing such networks. Chapter 3 formally introduced the problem of designing survivable IP-over-WDM networks and presented a review of some of the common approaches proposed in the literature. Chapter 3 also provided a detailed description of the SMART framework, which formed the basis for some of our work.

In Chapter 4, we provided a detailed analysis of SMART framework and its heuristic version (SMART-H). The analysis pointed out certain shortcomings of SMART and SMART-H, which allowed us to propose several enhancements that improved their performance. Also in chapter 4, we presented a new embedding algorithm based on randomized rounding and fractional multicommodity flow approximation. The embedding algorithm combined with a new subgraph finding algorithm was able to find survivable mappings for a larger number of IP-WDM network pairs.

In Chapter 5, we noted that for some IP networks a survivable embedding may not exist in a given WDM network. By adding new IP links or WDM links, an unsurvivable IP-WDM network pair can be converted to a survivable one. Adding WDM links by laying new optical fibers is difficult and more expensive to do. However, IP links can be easily added by establishing additional IP connections. In Chapter 5, we suggested using a combination of protection and restoration approaches to guarantee survivability for any given IP-WDM network pair, which is at least 2-edge connected. Protection was provided by setting up two lightpaths between certain pairs of IP routers such that these lightpaths did not share a WDM link. To determine the pairs of IP routers between which additional lightpaths must be established, we provided several heuristics that used SMART-H.

In Chapter 6, we explored the role that duality between circuits and cuts can play in realizing IP-over-WDM networks. Most often, for a primal algorithm based on circuits there is a dual algorithm based on cuts for the same problem. SMART algorithm is based on circuits, which begs the question whether there exists a dual methodology based on cuts. In Chapter 6, we investigated this question and developed certain new dual algorithms and insightful results. We also presented a logical topology structure that can always be in a survivable manner.

7.2 Future Work

There are many interesting directions to follow for future work.

1. The NP-complete nature of the problem leaves the door open for the development of more sophisticated algorithms that can find survivable embeddings for IP networks in WDM networks.

2. Another direction is to consider capacity utilization, when finding survivable mappings for the logical topologies. By considering the traffic load carried on each fiber, while finding the mappings, it is possible to reduce the number of lightpaths that must be rerouted in case of a failure. If a physical link that is heavily loaded fails, a large number of lightpaths must be rerouted. This may require significantly more time to recover from the failure. However, if the traffic load is distributed evenly, then the time to recover from the failure would be almost identical.

3. One key assumption, in the literature related to survivable IP-over-WDM networks, is that the logical topology is at least 2-edge connected. As future work, we would like to extend our work to topologies that are not initially 2-edge connected. Since logical and physical topologies that are at least 2-edge connected can survive single link failure, we can make given logical topologies 2-edge connected by adding additional links. By carefully adding links, it

is expected that a large number of logical topologies could be mapped in survivable manner.

4. One more direction to consider is to map logical topologies in such a way that they are able to survive multiple physical link failures. Noting that the problem of protecting networks against a single link failure is NP-complete, protecting against multiple simultaneous failures is also NP-complete. Therefore, designing heuristics that can protect networks against multiple failures will be an interesting and a challenging task.

Bibliography

[1] E. Modiano and A. Narula-Tam, ”Survivable Routing of Logical Topologies in WDM Networks,” in *Proc. IEEE INFOCOM 2001*, pp. 348-357, 2001.

[2] M. Kurant and P. Thiran, “On Survivable Routing of Mesh Topologies in IP-over-WDM Networks,” in *Proc. IEEE INFOCOM 2005*, pp. 1106-1116, 2005.

[3] T. Stern and K. Bala, *Multiwavelength Optical Networks: A Layered Approach.* Addison-Wesley Longman Publishing Co., Boston, MA, 1999.

[4] C. Glingener, “Next Generation Optical Networks and Related Components,” *Microsystem Technologies*, Springer Berlin/Heidelberg, Issue 5, Vol. 9, pp. 286-290, 2003.

[5] I. Chlamtac, A. Ganz and G. Karmi, “Lightpath Communications: An Approach to High Bandwidth Optical WDM,” *IEEE Transactions on Communications*, Issue 7, Vol. 40, pp. 1171-1182, Jul. 1992.

[6] A. Gnauck, G. Charlet, P. Tran, P. Winzer, C. Doerr, J. Centanni, E. Burrows, T. Kawanishi, T. Sakamoto and K. Higuma, “25.6-Tb/s WDM Transmission of Polarization-Multiplexed RZ-DQPSK Signals,” *Journal of Lightwave Technology*, Issue 1, Vol. 26, pp. 79-84, Jan. 2008.

[7] N. Ghani, S. Dixit and T. Wang, “On IP-over-WDM Integration,” *IEEE Communications Magazine*, pp. 72-83, Mar. 2000.

[8] W. Grover and D. Stamatelakis, “Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration,” in *Proc. IEEE ICC 1998*, pp. 537-543, 1998.

[9] M. Ryu and H. Park, “Survivable Network Design Using Restricted P-cycle,” in *Proc. International Conference on Information Networking*, Vol. 2662, pp. 918-927, 2003.

[10] C. Hou, “Design of a Fast Restoration Mechanism for Virtual Path-Based ATM Networks,” in *IEEE INFOCOM 1997*, pp. 361-369, 1997.

[11] K. Murakami and H. Kim, “Comparative Study on Restoration Schemes of Survivable ATM Networks,” in *IEEE INFOCOM 1997*, pp. 345-352, 1997.

[12] A. Fumagalli and L. Valcarenghi, “IP Restoration vs.WDM protection: Is There an Optimal Choice¿‘ *IEEE Network*, Issue 6, Vol. 14, pp. 34-41, 2000.

[13] L. Sahasrabuddhe, S. Ramamurthy and B. Mukherjee, “Fault Management in IP-Over-WDM Networks: WDM Protection Versus IP Restoration,” *IEEE Journal on Selected areas in Communication*, Issue 1, Vol. 20, pp. 21-33, Jan. 2002.

[14] J. Wei, “Advances in the Management and Control of Optical Internet,” *IEEE Journal on Selected Areas Communications*, Issue 4, Vol. 20, pp. 768-785, May 2002.

[15] E. Modiano and A. Narula-Tam, “Survivable Lightpath Routing: A New Approach to the Design of WDM-based Networks,” *IEEE Journal of Selected Areas in Communication*, Issue 4, Vol. 20, pp. 800-809, May 2002.

[16] O. Crochat, J. Boudec and O. Gerstel, “Protection Interoperability for WDM Optical Networks,” *IEEE/ACM Transactions on Networking*, Issue 3, Vol. 8, pp. 384-395, Jun. 2000.

[17] A. Todimala and B. Ramamurthy, “A Scalable Approach for Survivable Virtual Topology Routing in Optical WDM Networks,” *IEEE Journal on Selected Areas in Communications*, Issue 6, Vol. 25, pp. 63-69, Aug. 2007.

[18] F. Ducatelle and L. Gambardella, “FastSurv: A New Efficient Local Search Algorithm for Survivable Routing in WDM Networks,” in *IEEE Globecom 2004*, pp. 1925-1929, 2004.

[19] F. Ducatelle and L. Gambardella, “A Scalable Algorithm for Survivable Routing in IP-Over-WDM Networks,” in *Proc. BroadNets 2004*, pp.54-63, 2004.

[20] C. Liu and L. Ruan, “Logical Topology Augmentation for Survivable Mapping in IP-over-WDM Networks,” in *IEEE Globecom 2005*, pp.1885-1889, 2005.

[21] E. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numerische Mathematik*, pp. 269-271, 1959.

[22] M. Kurant and P. Thiran, “Survivable Mapping Algorithm by Ring Trimming (SMART) for Large IP-over-WDM Networks,” in *Proc BroadNets 2004*, pp. 44-53, 2004.

[23] M. Kurant and P. Thiran, “Survivable Routing of Mesh Topologies in IP-over-WDM Networks by Recursive Graph Contraction,” *IEEE Journal on Selected Areas in Communications*, Issue 5, Vol. 25, pp. 922-933, Jun. 2007.

[24] M. Javed, K. Thulasiraman, M. Gaines and G. Xue, “Survivability Aware Routing of Logical Topologies: On Thiran-Kurant Approach, Evaluation and Enhancements,” in *IEEE Globecom 2006*, pp. 1-6, 2006.

[25] R. Ahuja, T. Magnanti and J. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Upper Saddle River: Prentice Hall, 1993.

[26] J. Kleinberg, "Approximation Algorithms for Disjoint Paths Problems," MIT. Cambridge, MA, PhD Thesis, 1996.

[27] M. Blesa and C. Blum, "Ant Colony Optimization for the Maximum Edge-Disjoint Paths Problem," *Applications of Evolutionary Computing,* Springer Berlin/Heidelberg, Vol. 3005/2004, pp. 160-169, 2004.

[28] P. Mateti and N. Deo, "On Algorithms for Enumerating All Circuits of a Graph," *SIAM Journal on Computing*, Issue 1, Vol. 5, pp. 90-99, Mar. 1976.

[29] K. Mehlhorn and S. Naher, *LEDA: A Platform for Combinatorial and Geometric Computing.* Cambridge University Press, Cambridge, MA, 1999.

[30] F. Harary, "The Maximum Connectivity of a Graph," in *Proc. of the National Academy of Sciences of the United States of America*, Issue 7, Vol. 48, pp. 1142-1146, Jul. 1962.

[31] K. Thulasiraman and M. N. S. Swamy, *GRAPHS: Theory and Algorithms.* Wiley-Inter-science, 1992.

[32] B. Waxman, "Routing of Multipoint Connections," *IEEE Journal on Selected Areas in Communications*, Issue 9, Vol. 6, pp. 1617-1622, Dec. 1988.

[33] P. Raghavan and C. Thompson, "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs," *Combinatorica 7*, Issue 4, Vol. 7, pp. 365-374, Dec. 1987.

[34] N. Garg, and J. Konemann, "Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems," in *IEEE Symposium on Foundations of Computer Science*, pp. 300-309, 1998.

[35] L. Fleischer, "Approximation Fractional Multicommodity Flow Independent of the Number of Commodities," *SIAM Journal of Discrete Mathematics*, Vol. 4, pp. 505-520, 2000.

[36] M. Javed, K. Thulasiraman and G. Xue, "Lightpaths Routing for Single Link Failure Survivability in IP-over-WDM Networks," *Journal of Communications and Networks, Special Issue on Routing, Path Computation and Traffic Engineering in Future Internet*, Issue 4, Vol. 9, pp. 394-401, Dec. 2007.

[37] J. Suurballe and R. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks*, Issue 2, Vol. 14, pp. 325-336, 1984.

[38] M. Javed, K. Thulasiraman and G. Xue, "Logical Topology Design for IP-over-WDM Networks: A Hybrid Approach for Minimum Protection Capacity," in *Proc. IEEE ICCCN 2008*, pp. 1-7, 2008.

[39] Y. Perl and Y. Shiloach, “Finding Two Disjoint Paths Between Two Pairs of Vertices in a Graph,” *Journal of the ACM*, Issue 1, Vol. 25, pp. 1-9, 1978.

[40] K. Thulasiraman, M. Javed and G. Xue, “Circuits/Cutsets Duality and a Unified Algorithmic Framework for Survivable Logical Topology Design in IP-over-WDM Optical Networks,” in *Proc. IEEE INFOCOM 2009*, 2009.

Appendix A

Regular Graphs

Regular graphs were generated using a procedure introduced by Harary in [30]. The procedure constructs a k-connected graph $H_{k,n}$, which has exactly $\lceil \frac{kn}{2} \rceil$ edges, here n is the number of vertices in the graph and k is the required connectivity. [30] considers three cases

(i) k is even

(ii) k is odd and n is even

(iii) k is odd and n is odd.

For our simulations we generated graphs with even k (case (i)), using the procedure given in [31].

Let $k = 2r$. Then $H_{2r,n}$ has vertices $v_0, v_1, v_2, ..., v_{n-1}$ and two vertices v_i and v_j are adjacent if $i - r \leq j \leq i + r$, where addition is modulo n. The code for generating code using $VC++8.0$ and $LEDA$ [29] is given below.

A regular graph with $n = 10$ and $k = 4$ using the above procedure is shown in Fig. A.2. To introduce randomness in regular graphs, each node was assigned a random unique node number (node_number). For physical topologies following code was used.

```
/* This method generates regular graph with a specified number of nodes
and connectivity.
@param G The generated regular graph.
@param numNodes The required number if nodes in the regular graph G.
@param connectivity The desired connectivity of the generated regular
graph */
void generate_regular_topology(GRAPH<int, int> & G, int numNodes,
int connectivity) {
  int r;
  list <node> allNodes;
  list_item item1;
  list_item item2;
  if ((connectivity % 2) != 0) {
    std::cout << "ERROR: connectivity of the regular topology to be
generated must be even." <<std::endl;
    exit(1);
  }
// Create new nodes
  for (int i = 0; i < numNodes; i++)
    G.new_node();
// Get a list of the newly created nodes
  allNodes = G.all_nodes();
// Now, assign edges in such a way that the graph will be regular and
of the desired connectivity
  r = connectivity / 2;
  for (int j = 0; j < numNodes; j++){
    for (int k = 0; k < numNodes; k++) {
      if (((abs(j - k) <= r) || (abs(j - k + numNodes) <= r)) &&
      (j < k)) {
        item1 = allNodes.get_item(j);
        item2 = allNodes.get_item(k);
        G.new_edge(allNodes.contents(item1),
        allNodes.contents(item2));
      }
    }
  }
  G.make_bidirected ();
}
```

Figure A.1: Procedure to generate Regular Graphs.

```
1. int node_array[ number_of_nodes ];
2. for( int i = 0; i < number_of_nodes; i++ ) {
       node_array[i] = i;
   }
3. for( int i = 0; i < number_of_nodes /2; i++ ) {
       permute(node_array);
   }
4. for( int i = 0; i < number_of_nodes; i++ ) {
       node_number[i]= node_array[i];
   }
```

Since the logical nodes were a subset of logical nodes, the above statements 1, 2, and 3 were adapted for logical_node_number and number 4 above was changed to:

```
for( int i = 0; i < number_log_of_nodes; i++ ) {
    log_node_number[i]= node_array[i];
}
```

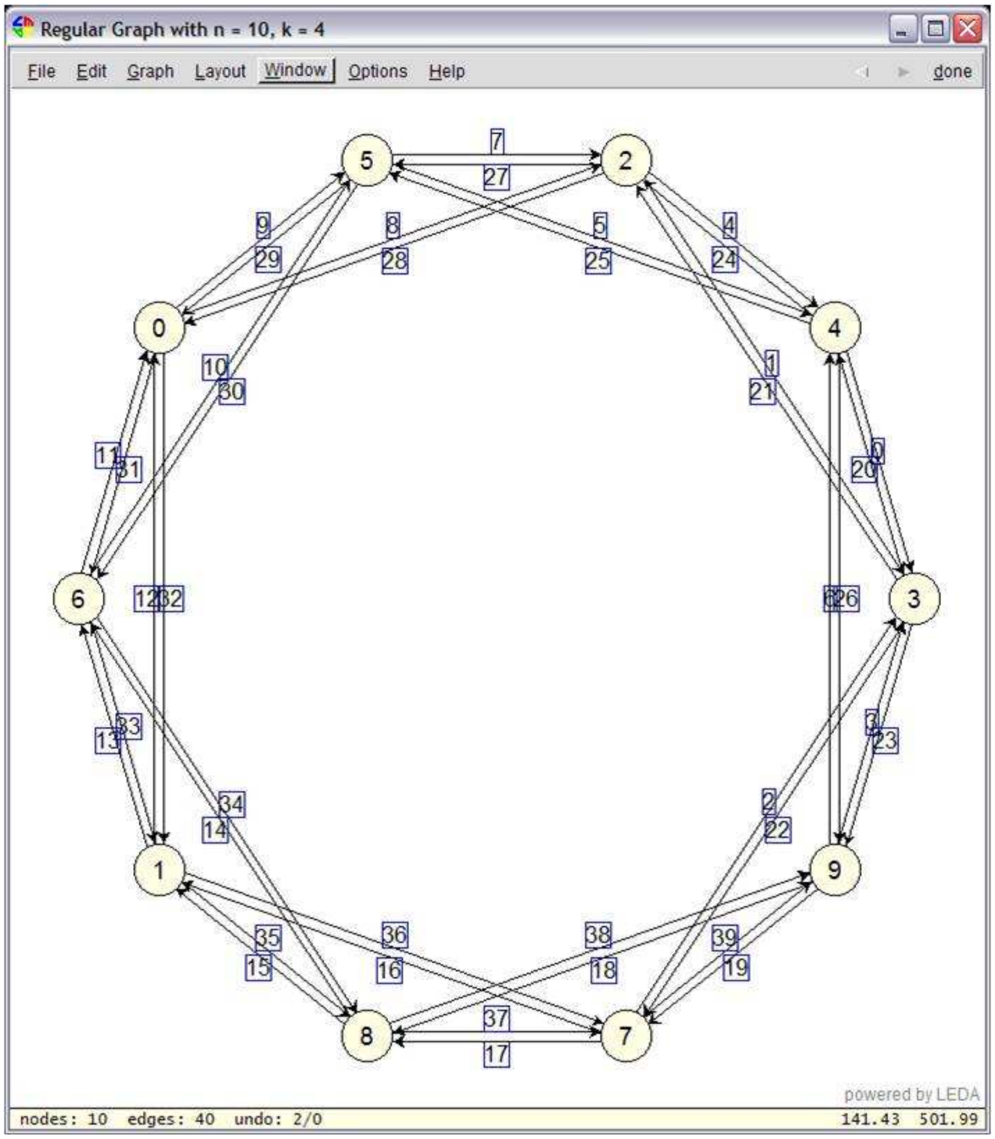


Figure A.2: A regular graph with number of nodes $(n) = 10$ and connectivity $(k) = 4$ generated using procedure given in [31].

Appendix B

Waxman's Random Graphs

Waxman's random graphs exhibit some of the characteristics of actual networks. [32] and were used when random graphs were required in the simulation. [32] proposes two type of the random graph. We adopted the graph generation procedure described below.

1. Given a node set, for every node pair a distance is chosen in the range (0, $L > 0$) from a uniform random distribution.

2. An edge was introduced between a node pair (u, v) with a probability given as

   $prob\{u, v\} = \beta exp \left\{ -\frac{d(u,v)}{L\alpha} \right\}$

   Here α and β are parameters in the range $(0, 1)$.

Since all the presented algorithms require 2-edge connected graphs, first all the nodes were connected to form a ring and then additional edges were added using the above procedure.

To further introduce randomness, each node was assigned a random unique node number. To assign node numbers to physical and logical nodes, the procedure describe in Appendix A was used.

Appendix C

List of Abbreviations

ACO	Ant Colony Optimization
ADM	Add-drop Multiplexor
AON	All Optical Network
APS	Automatic Protection Switching
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
BGA	Bounded Greedy Algorithm
CWDM	Coarse Wavelength Division Multiplexing
DCS	Digital Cross-connect Switch
DWDM	Dense Wavelength Division Multiplexing
EMI	Electro-magnetic Interference
Gbps	Gigabits per second
HA-1	Hybrid Algorithm 1
HA-2	Hybrid Algorithm 2
HA-3	Hybrid Algorithm 3
HA-4	Hybrid Algorithm 4
ILP	Integer Linear Program
IP	Internet Protocol
ISP	Internet Service Provider
LBM	Load Based Mapping
LEDA	Library of Efficient Data Types and Algorithms
MCF	Multicommodity Flow
MM	Modify and Map

MMCF	Maximum Multicommodity Flow
MPLS	Mutiprotocol Label Switching
ms	Milliseconds
MSGA	Multi-start Greedy Algorithm
OA	Optical Amplifier
OADM	Optical Add-drop Multiplexor
O-E-O	Optical-Electrical-Optical
OSPF	Open Shortest Path First
OXC	Optical Cross-connect
P-cycle	Protection Cycle
PID	Protection Interoperable Design
PIW	Protection Interoperability for WDM Networks
PRC	Pure Ring Cover
RFI	Radio Frequency Interference
RIP	Routing Information Protocol
SDH	Synchronous Digital Hierarchy
SHR	Self Healing Rings
SM	Simple Mapping
SMART	Survivable Mapping Algorithm by Ring Trimming
SMART-H	SMART Heuristic
SONET	Synchronous Optical Networks
Tbps	Terabits per Seconds
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
VP	Virtual Path
WC	Wavelength Converter
WDM	Wavelength Division Multiplexing
WWW	World Wide Web