2010

On Traffic Grooming and Survivability in WDM Optical Networks

Long Long
Iowa State University

On traffic grooming and survivability in WDM optical networks

by

Long Long

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Program of Study Committee:
Ahmed E. Kamal, Major Professor
Arun Somani
Manimaran Govindarasu
Lu Ruan
Lizhi Wang

Iowa State University

Ames, Iowa

2010

DEDICATION

I would like to dedicate this thesis to my grandparents and my parents who have given me endless love since the first day of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to everyone who helped me through my PhD journey. This thesis could not have been be completed without their support.

First, I want to thank my major advisor, Dr. Kamal, for his constant guidance, patience and support throughout my research and the writing of this thesis. His dedication and professionalism inspired me to keep working hard, and he has set a perfect example for me to follow in my career. I feel truly fortunate to have had an opportunity to be advised by him. I would also like to thank my committee members for their efforts and contributions to this work: Dr. Somani and Dr. Govindarasu have given great comments and advice to me while I was completing this thesis. Dr. Ruan instilled in me the rigorous attitude of research through our collaborative research work, and my discussions with Dr. Wang inspired my research ideas by presenting me with new perspectives.

I also have a number of great colleagues who have provided me great ideas and suggestions throughout my research. It has been a great experience to collaborate with Taiming Feng and Shizheng Li, from whom I have learned a passion for research and life. I also want to thank my labmates, Mizard Mohandespour, Hisham Almasaeid and Mohammad Salah, for their suggestions and effort that inspired me to produce better work. I thank my roommate, Fei Wang, and my friends, Kang Gui, Daolin Chen and others for making my life in Ames so enjoyable and memorable.

I would like to express my deepest gratitude to my grandparents whose unselfish love and constant support have been truly invaluable throughout my life. I also want to thank my parents deeply for their continuous love, support and encouragement, which always give me strength and warmth no matter how far away I was from them. Lastly but most importantly,

I would like to thank my wife, Ciarra, for her love, patience, understanding and support. Without her, my life would have been just incomplete.

This research was supported in part by grant CNS-0626741 from the National Science Foundation.

CHAPTER 1. BACKGROUND

1.1 Introduction

In this chapter, we will first introduce the concept of traffic grooming and traffic grooming problems in WDM optical networks. Second, we will particularly introduce the multipoint traffic grooming problem, which includes three communication modes: multicast, many-to-one and many-to-many. Finally, we will introduce network survivability and summarize general network resilience techniques.

1.1.1 The Traffic Grooming Problem

Wavelength-division multiplexing (WDM) technology allows an aggregate traffic on the order of Tbps to be carried on a single fiber, with each wavelength carrying traffic in the tens of Gbps order. In WDM networks, data is allocated and transported in all-optical channels where each end-to-end optical channel is called a "lightpath". A lightpath can traverse one or multiple physical fiber links and each link has to use the same wavelength if the network nodes are not equipped with wavelength converters. Otherwise, a lightpath does not need to comply with the wavelength continuity constraint such that any available wavelength can be used by a lightpath on any physical link traversed.

Although each wavelength can carry data up to 100 Gbps based on the current technology [1], the traffic demands of network applications are at much smaller granularities than wavelength bit rates. In order to utilize network capacity more efficiently, a number of flows from multiple network connections with sub-wavelength granularities may be packed onto the same lightpath. This process of allocating low bit rate tributary streams to a lightpath with high bandwidth is referred to as *traffic grooming* [2]. To be capable of grooming the traffic, a node

should be equipped with devices to multiplex or demultiplex low rate traffic streams at the sub-wavelength layer, which is the electronic layer. Such device is called Electronic Add/Drop Multiplexer (ADM), which is required at a node if it either has data to transmit to or receive from another counterpart.

There are two types of traffic grooming problems, static and dynamic. In the static problems, traffic demands are given in advance and the objective is usually to minimize the overall network cost of provisioning all the traffic demands. As the dominant cost of optical networks, the number of ADMs is considered as the objective to be minimized in most of the literature. In the dynamic scenarios, the traffic requests are generated and arrive at the networks in real time and the traffic only stays for a certain period of time. Due to the lack of knowledge of future traffic demands and the limited network resources, some incoming traffic requests may not be provisioned due to the lack of available resources. Therefore, the objective for dynamic traffic grooming is usually to maximize the throughput or minimize the blocking probability of connections.

1.1.2 Multipoint Traffic Grooming

The unicast service is the dominant service type in the Internet and both the static and dynamic unicast traffic grooming problems have been widely studied in the literature. Recently, multipoint communication has emerged as an important service mode for a number of applications such that real-time multimedia streaming, distributed computing and so on. Multipoint communication includes many-to-one, one-to-many (multicast) and many-to-many, in which many-to-many traffic mode is the generalization of all-to-all traffic scenario. All-to-all communication requires all nodes in the network to exchange their traffic, while in the many-to-many case, there are multiple communication groups and each node in a group has to receive the data from all the other users in the same group. Since a single multipoint application is very unlikely to consume the entire bandwidth provided by a wavelength, multipoint traffic grooming becomes essential.

As the most common multipoint service, a multicast session will form a multicast tree in

the network rooted at one source and destined to multiple destinations. Multicast will be implemented in the next generation internet as an intrinsic functionality at the IP layer, in which each router is able to replicate an incoming packet into multiple copies and deliver them through multiple outgoing links. In order to transmit data through optical channels, however, data at each router on the multicast tree has to go through O-E-O conversion, defined as Optical to Electronic to Optical conversion, since incoming optical signals have to be converted into electronic signals in order to be duplicated at the electronic domain, e.g., at the IP layer. The duplicated data then needs to be converted back to the optical domain and continue the transmission. In this case, the data stream may experience a large delay due to the O-E-O conversion at each intermediate node.

However, a more efficient way to perform multicasting in optical networks is at the optical layer by using optical splitters, which are passive devices that are capable of splitting one optical signal into multiple copies with each copy transmitted at a fraction of the power. If a multicast session is provisioned as a tree in the optical domain, it is called a "light-tree". Implementing a light-tree results in two advantages: low latency and cost reduction. Without O-E-O conversion at each intermediate node, traffic can be transmitted on the optical layer transparently with less latency. Moreover, duplicating traffic at the optical layer avoids extra traffic generation and termination such that the number of required ADMs can be reduced. Compared to the high cost of an Electronic ADMs, the cost of deploying optical splitters is almost negligible.

On the contrary, many-to-one service is the opposite of multicast services where the traffic streams from multiple sources merge and are aggregated in the network to reach the same destination. Traffic has to be terminated at merging point in order to be aggregated and allocated into the same lightpaths, which can only be performed in the electronic domain. Thus, a many-to-many service can be considered as the combination of many-to-one and multicast services and can be implemented in both the electronic and optical domains. Considering the complexity of either many-to-one or multicast traffic grooming problems, many-to-many traffic grooming problems are at least as hard as the two problems.

1.1.3 Network Survivability

Network survivability, defined as networks' ability to continue functioning correctly in the presence of failures of any network components, is an important requirement for WDM optical networks due to their ultra-high capacity [3]. A single failure can disrupt millions of applications and results in tremendous revenue loss to both end users and network operators. The common requirement of the downtime of a leased connection in the industry is less than 5 minutes per year [1]. Although many network components can cause the failure of a connection, such as fibers, switches, transceivers and so on, the most common network failure is the link failure and most likely it is the fiber cut because of construction and dig up. Therefore, link failures will be the only failure event considered in our study.

In general, there are two ways to provide recovery from failures, namely, protection and restoration. In the protection paradigm, each connection is provisioned and allocated certain amounts of spare resources for protection, which can be used to reroute the traffic upon a failure on the connection. The protection resources are allocated to each connection prior to any network failure. On the contrary, the restoration paradigm does not assign any spare resource for the protection in advance. Upon a failure, network has to search spare resources to reroute each disrupted connection around the failure. Clearly, restoration paradigm cannot guarantee the protection for any service and may take much longer to recover traffic upon a failure, due to the real-time resource searching and backup provisioning. However, the protection paradigm can guarantee the protection of any provisioned connection and has much better control of the traffic recovery time. Therefore, protection strategies are more favored in the network survivability, and our study here will focus on protection-based schemes.

Any protection-based scheme consists of two types of spare resource allocation, dedicated and shared. Dedicated protection schemes assign a unique unit of spare resource specifically for each connection whereas shared protection schemes allow multiple connections to share the same unit of resource for protection in order to increase resource efficiency. However, there is a trade-off between the resource efficiency and traffic recovery speed. In general, dedicated protection schemes provide faster traffic resilience than shared protection schemes, because no

reconfiguration is needed in the middle of a backup route upon a network failure. If a unit of spare resource is shared among a number of connections, some network nodes may need to reconfigure their switches in order to reroute the traffic correctly, which may result in a longer recovery time. Dedicated protection schemes are prevalently adopted in ring-based networks. As mesh-based networks emerge, shared protection schemes become more popular due to the significant capacity savings. However, by carefully designing the protection scheme, the traffic recovery time can also be controlled within a low range, which is comparable to dedicated protection schemes.

Although network survivability can be implemented on multiple layers, such as the optical layer (WDM, SONET and etc.), the link layer (ATM, MPLS and etc.) and the application layer, the optical layer provides the most effective protection mechanism against link failures in optical networks, particularly in all-optical switching networks, since all the disrupted traffic can be recovered in a matter of tens of milli-seconds and the information required for traffic recovery and the design on the management plane can be much simpler. Thus, we will focus on the protection schemes in the optical layer.

1.2 Literature Review

In this section, we will study the recent work of traffic grooming and survivability in the literature. We will first review the work addressing multipoint traffic grooming problems in both WDM ring and mesh networks. And then we will study the related work of network survivability in optical mesh networks. A variety of protection schemes handling link failures will be reviewed for both unicast and multicast service modes.

1.2.1 Multipoint Traffic Grooming

1.2.1.1 WDM Ring Networks

Due to the prevalence of unicast services in networks, both static and dynamic unicast traffic grooming problem have been widely studied in the literature. Among a number of network architectures, ring topologies drew significant attention in the research community

due to the availability of legacy SONET equipment, which are usually configured in ring topologies, especially in Metro areas [5]-[9]. A survey of traffic grooming problems in WDM ring networks was conducted in [5], which also proved the problem NP-complete. References [7], [8] and [9] solved the problem of unicast traffic grooming in ring networks by defining the number of transceivers or ADMs as the cost objective to be minimized. Reference [8] proposed a near-optimal heuristic algorithm to solve the static traffic grooming problem and reference [9] formulated the same problem as an ILP and then proposed Simulated-annealing-based heuristic for solving the problem. Reference [7] studied several OADM ring architectures and addressed both static and dynamic traffic patterns. Furthermore, the authors in [6] consider another cost criterion, the number of routing wavelengths, as the major cost factor and obtain the optimal solutions by using an ILP formulation. They also derive several bounds on the total number of wavelengths used for provisioning.

As multipoint services are being introduced in the Internet to serve a growing number of applications, many recent work starts to address multipoint traffic grooming in ring networks. The work presented in [11]-[14] address the all-to-all traffic pattern in which every node has to receive traffic from the rest of the nodes in the network. In [11], the authors proposed algorithms to obtain the optimal solution for some special traffic scenarios and derived a lower bound on the number of ADMs required in the case of uniform all-to-all communication. They also proposed an optimal algorithm to pack the low bit rate traffic streams into the minimum number of wavelengths at a hub node if traffic bifurcation is allowed. Similar work was done in [12] and [13], in which several lower and upper bounds are obtained for general uniform all-to-all traffic scenarios and the optimal solutions are obtained if certain conditions are satisfied with the grooming ratio and the number of nodes in the network by using graph theory techniques. Arbitrary traffic scenarios are studied in [14]. In [15], authors proposed a new multicast virtual path scheme and several multicast assignment schemes on SONET rings to implement video conferencing, which is a specific application of all-to-all communication. The objective minimized in this work is bandwidth utilization.

Both [17] and [18] cope with multicast traffic grooming in WDM ring networks and consider

the cost of electronic devices as the objective. However, another cost function similar to the number of ADMs, defined as the number of e-DaC grooming ports, is presented in [17] where an e-DaC grooming port refers to Electronic Drop-and-Continue grooming ports, in which part of the traffic is dropped off at a node, and the rest of the traffic continues. Adopting the similar idea of multicasting in the optical domain, the authors in [18] proposed a node architecture, referred to as the Tap-and-Continue node, in which a node is capable of replicating optical signals in the optical domain using optical splitters, and one copy is dropped locally, while the remaining signal continues on the fiber without the additional cost of ADMs. Although multicast, many-to-one and all-to-all traffic patterns have been studied in both SONET and WDM ring networks, to the best of our knowledge, many-to-many communication has not been addressed in the literature, which will be one of the problems addressed in this dissertation.

1.2.1.2 WDM Mesh Networks

Traffic grooming in mesh-based WDM networks has attracted an increased amount of effort in the research community due to the transition from ring to mesh networks. A lot of work has addressed unicast traffic grooming problems [2],[19]-[21]. The authors in [2] defined the basic traffic grooming problem and proved it NP-complete in mesh networks and then proposed a novel network model based on an auxiliary graph in [19], by which a unicast traffic grooming can be achieved with various objectives. A number of heuristic algorithms are also proposed to solve both static and dynamic problems. However, the study presented in [20] focus on minimizing the number of optical transponders (ADM, LTE and etc) with static traffic requests. The problem is first formulated as an Integer Linear Program and then solved by a heuristic method by decomposing the problem into two subproblems, in which the first subproblem (traffic grooming and routing) is solved first, and the solution is used as an input to the second subproblem (wavelength assignment). The study presented in [21] addressed the dynamic traffic grooming case, but the traffic demands can incorporate both unicast and multicast modes.

Recently, multipoint traffic grooming in mesh networks became very important and this is

why a number of studies addressing this problem have recently appeared in the literature [22]-[24]. Reference [22] addresses network design and session provisioning under both static and dynamic multicast traffic cases and summarizes a variety of algorithms to solve the grooming problem. Furthermore, the authors in [23] developed a unified framework for the optimal provisioning of multicast traffic grooming with static traffic. An ILP formulation is designed to minimize the total number of electronic equipments and wavelengths used. In order to solve large size problems, a number of heuristic approaches are proposed. Many-to-one traffic grooming problem was studied in [24] and the problem was first formulated as an Mixed ILP and then solved by an Dynamic Programming heuristic approach. A more recent work presented in [25] studied many-to-many traffic grooming in mesh networks, in which a couple of Mixed ILP formulations were proposed based on two provisioning approaches. By studying the optimal solutions obtained by solving MILPs, the authors also proposed a heuristic that can achieve near-optimal solutions but significantly reduces the time complexity comparing original MILPs.

1.2.2 Survivability in WDM Mesh Networks

We discuss the various protection schemes in all-optical WDM mesh networks. A comprehensive study of survivability in optical networks is provided by reference [4]. A protection scheme is usually evaluated by four criteria: recovery speed, restorability, capacity efficiency and algorithm scalability [26]. They are explained as follows:

- Recovery speed: how fast the disrupted traffic can be recovered upon a failure.
- Restorability: what type of failures that a protection scheme can cope with, such as single-link, multiple-link or node failure.
- Capacity efficiency: how much capacity to provision and protect the traffic demands.
- Algorithm scalability: how efficiently the algorithm can run as the problem size increases.

In the context of WDM mesh networks, we will focus on link failure scenarios, in which single-link failure will be the main concern in our study.

1.2.2.1 Link/Segment-based Protection Schemes

Link-based protection schemes are the most straightforward protection approaches. Once a span is down, the two end nodes of the failed span will reconfigure their switches and reroute all the traffic traversing that span through a backup path that connects the two end nodes. The corresponding failure recovery architecture was proposed in [26]. Since the end nodes can immediately detect the failure and reconfigure their switches, this type of protection schemes usually provides the fastest recovery speed.

Extended from link-based schemes, segment-based protection schemes protect segments, where a segment is defined as a sequence of successive links. A path is divided into a number of segments and each segment is protected by a backup path. Any link failure occurring in a segment results in the failure of the segment such that the traffic traversing this segment will be rerouted through the backup path. The recovery process is similar to that of link failure protection and relatively takes longer recovery time but less capacity than link-based protection schemes.

Both link and segment-based protection schemes against single-link failure have been studied extensively in the literature. There are two types of link-based approach: dedicated and shared [27]-[30]. Shared Link Protection (SLP) schemes have a significant advantage over Dedicated Link Protection (DLP) schemes in terms of capacity cost. However, DLP has better traffic recovery time than SLP because the capacity sharing in SLP can result in longer switching time on backup paths. Due to the flexibility of segments, a number of shared segment-based protection schemes were proposed in the literature recently. First, a heuristic algorithm, named Short Leap Shared Protection (SLSP), was proposed in [31], in which a path is divided into several equal-length and overlapped segments. And then the problem with segment-based protection was formulated as an ILP in [32]. However, the high complexity of the ILP makes it incapable of solving large-sized problems and therefore a near-optimal dynamic programming heuristic algorithm was proposed. A new shared segment protection method was proposed in [33] along with the GMPLS-based recovery framework. The simulation results with dynamic traffic showed that the proposed approach achieved high reliability of transport services and

has advantages over both link and path-based protection schemes.

1.2.2.2 Path-based Protection Schemes

Path-based protection schemes drew great attention in the research community due to the high capacity efficiency. A path here is defined as a lightpath instead of a path in the application layer such that protection can be implemented in the optical layer. The path-based protection schemes can be failure independent and failure dependent. In failure independent path protection (FIPP) schemes, each working path is protected by a unique link-disjoint backup path between the same pair of nodes. If a link fails on the working path, the traffic switches to the backup regardless of the location of the failure. However, a working path may be protected by multiple distinct backup paths in failure dependent path protection (FDPP) schemes. Based on the location of the failure, the end nodes of the path choose the corresponding backup path to reroute the traffic. FDPP schemes have greater flexibility but require more complex implementation.

We first review FIPP schemes. The dedicated path protection (1+1) was studied in both [3] and [34], in which each path is protected by a backup path with a dedicated wavelength. However, it is not efficient in terms of network capacity. Furthermore, the problems using both shared and dedicated path protection schemes are formulated as ILPs in [30] and the capacity efficiency and the traffic recovery time of different schemes were compared. It was shown that Shared Path Protection (SPP) schemes provide significant savings in capacity utilization over Dedicated Path Protection (DPP) and link-based protection schemes. Among all the protection schemes, an approach, called Shared Backup Path Protection (SBPP), achieves the optimal capacity cost [3, 29]. However, in terms of traffic recovery time, the results in [30] showed that path-based schemes take longer time than link-based schemes, and that shared protection schemes perform worse than dedicated protection schemes, which demonstrated the tradeoff between the capacity efficiency and traffic recovery speed. Moreover, two types of FDPP schemes are proposed in [35] and [36], respectively, in which Strict FDPP scheme, proposed in [35], does not switch to the backup path until a failure occurs on the working

path. On the contrary, traffic may be switched from a working path to a backup path upon a failure, even if this failure does not occur on any link used by the working path. This method is referred to as Flexible FDPP and is proposed in [36].

1.2.2.3 *p*-Cycle-based Protection Schemes

A promising protection technique designed particularly for mesh networks is the preconfigured protection cycles (*p*-cycles). A *p*-cycle can achieve the speed of ring-based schemes with capacity efficiency of mesh [39, 40]. *p*-Cycles are established by assigning the spare capacity into pre-configured cycles. As a link protection scheme, *p*-cycles only need to reconfigure the two end nodes of the failed link and hence achieve a very short traffic recovery time since the protection path along the cycle is fixed and pre-configured. Besides the on-cycle links, a *p*-cycle can also protect straddling links. A unitary *p*-cycle can protect two units of working capacity on a straddling link by providing two diverse paths on the cycle, which accounts for the enhanced capacity efficiency of *p*-cycles.

Since the concept of the p-cycle was first introduced in [39], a large number of papers in the literature have addressed the *p*-cycle design problem with unicast traffic against a single-link failure. The authors in [39]-[41] solved the problem in two steps by first routing the connections using routing algorithms and then selecting the best candidates from the enumeration of all the cycles to protect the established connections. However, the optimality of the solution was relaxed by dividing it into two subproblems. The approaches proposed in [42] and [43] solve the problem optimally by minimizing the total cost of primary and protection capacity jointly. Besides link protection, p-cycles are also extended to protect segments and paths in [44] and [45], in which [45] proposed a Failure Independent Path-Protecting (FIPP) *p*-cycle approach that achieves the best capacity efficiency among all *p*-cycle-based protection schemes. The work conducted in [46] actually showed that *p*-cycles achieve much better capacity utilization than dedicated protection schemes and slightly less than shared link and path protection schemes.

1.2.2.4 Multicast Protection Schemes

Due to the high capacity of a fiber, a single fiber cut may lead to more serious consequence in the scenario of multicast service compared to unicast service, since the data delivery to multiple nodes can be disrupted. Therefore, efforts have been exerted to deal with protection of a multicast session against single link failure.

A straightforward method proposed in [77] is to find two link disjoint light-trees such that both of them start from the source and end at the destination nodes. It is clear that this method is not capacity efficient and it is not always possible to find two link-disjoint trees in a network. In [79], the authors introduced a number of protection schemes: link-based, segment-based and path-based. In link-based and segment-based approaches, a multicast session is routed first to construct a multicast tree, and then each link or segment on the tree is protected by a path starting at the tail node and finishing at the head node of the link or segment it protects. Alternatively, a path-based protection scheme also proposed in [79], named optimal path-pair-based shared disjoint paths (OPP_SDP) algorithm, achieves the best result in terms of network resource efficiency by self-sharing primary and spare capacity [81]. The idea is to find two shortest link disjoint paths for each source and destination pair.

Recently, a couple of new technologies were applied to the survivability problem of multicast services, namely, p-cycle [83] and network coding [84]. They do have some nice features such as fast recovery speed of p-cycle or high bandwidth utilization of network coding. However, the design of a p-cycle to protect multicast session can be fairly complicated compared with the link protection. For instance, the ILP formulations proposed in [83] soon become intractable as the problem size increases. Network coding introduces extra computational cost as well as O-E-O conversion since network coding can only be performed in the electronic domain in current optical networks, which may introduce an additional expense.

1.3 Outline of Thesis

The rest of the thesis will be organized as follows:

- Chapter 2 addresses the many-to-many traffic grooming problem in unidirectional ring

networks [10]. We propose to employ network coding instead of traditional traffic grooming scheme to provision given many-to-many traffic requests. Two types of unidirectional ring networks, single-hub and unhubbed, are considered, respectively.

- Chapter 3 addresses unicast protection problem against double-link failures by using p-cycles [37]. This chapter is a collaborative work, in which both static and dynamic traffic scenarios are addressed. My major contributions are the theoretical analysis of protection conditions and static traffic protection. The objective of static traffic protection problem is to optimize the total provisioning cost and is formulated as an Integer Linear Program (ILP). In another part, the work addresses dynamic traffic protection where two heuristic algorithms are proposed to address this cases.

- Chapter 4 presents a new protection scheme, extended from traditional p-cycle, called p-cycle with Parasitic Protection links (PPL). A p-cycle with PPL is named as p^2-cycle [38]. We will study the cost and failure recovery performance of p^2-cycles and apply p^2-cycle to both static and dynamic traffic scenarios.

- Chapter 5 addresses the multicast survivability problem against any single-link failure with minimum cost [75]. We will propose a new protection scheme, namely, Segment-based Protection Tree (SPT), to protect a multicast session and then extend it to address dynamic traffic by proposing two heuristic algorithms. The overall performance of cost and failure recovery will be studied.

- Chapter 6 will conclude the thesis as well as the future work.

CHAPTER 2. MANY-TO-MANY TRAFFIC GROOMING IN WDM RING NETWORK USING NETWORK CODING

A paper published in *IEEE/OSA Journal of Lightwave Technology* [1]

Long Long and Ahmed E. Kamal

Abstract

In this paper we address the problem of traffic grooming in WDM rings with all-to-all and its generalization to many-to-many service by using network coding. We consider minimizing the number of Line Terminating Equipment (LTE) on two types of unidirectional rings, namely, single-hub and un-hubbed rings, as our objective. In single-hub rings, we investigate the minimum cost provisioning of uniform all-to-all traffic in two cases: where network coding is used to linearly combine data, and where it is not used and data is transmitted without coding. We generalize the service mode to many-to-many and evaluate the cost of provisioning. In un-hubbed ring, we propose a multi-hub approach to obtain the minimum cost provisioning in the case of all-to-all and many-to-many traffic. In each type of ring topology, two network scenarios are considered: first, the distinct communication groups in the ring are node-disjoint and second, the different groups may have common member nodes. From our numerical results, we find that under many-to-many traffic pattern for both scenarios, network coding can reduce the network cost by 10-20% in single-hub rings and 1-5% in un-hubbed rings in both network scenarios.

[1] This is a modified version of the work published in J. of Lightwave Tech., 2009, Vol. 27, Issue 19, pp. 4209-4220 [10].

2.1 Introduction

Wavelength-division multiplexing (WDM) technology allows an aggregate traffic on the order of Tbps to be carried on a single fiber, with each wavelength carrying traffic in the tens of Gbps order. However, the traffic demands of network applications are at much smaller granularity than wavelength bit rates. In order to utilize wavelength capacity more efficiently, a number of flows from multiple network connections with sub-wavelength granularity may be packed onto the same wavelength. This process of allocating low bit rate tributary streams to wavelengths with high bandwidth is referred to as *traffic grooming*. There are two types of traffic grooming problems, static and dynamic. The objective of the static problem is usually to minimize the overall network cost, given the traffic demands, whereas in the dynamic problem, maximizing the throughput or minimizing the blocking probability of connections.

The static traffic grooming problem of unicast traffic has been widely studied in the literature [2], [6, 20]. But recently, multipoint traffic has become more important in a number of application environments, and this is why a number of studies addressing multipoint traffic grooming have recently appeared in the literature [22, 23, 24]. Among a number of network architectures, ring topologies drew significant attention in the research community due to the availability of legacy SONET equipment [7, 8, 9]. In ring networks, both all-to-all [11, 12, 13, 15], multicast [17, 18] and arbitrary[14] traffic scenarios have been studied. Most of the literature addressing this problem focuses on evaluating and reducing the dominant cost in the optical network, namely, Electronic Add-Drop Multiplexers (ADM), which is required at a node if it either has data to transmit to or receive from another counterpart. The number of ADMs required at a node is only a function of the number of lightpaths established and terminated at the node. Another cost function that is similar to the number of ADM is the number of e-DaC grooming ports presented in [17], which refers to Electronic Drop-and-Continue grooming ports, in which part of the traffic is dropped off at a node, and the rest of the traffic continues.

In this paper, we address the static traffic grooming problem of a class of multipoint traffic in unidirectional ring networks with the number of electronic Line Terminating Equipments

(LTE) ports, be it ADM ports, as the network cost. The number of wavelengths and the cost of other optical equipment, such as the optical splitter (which is negligible compared to the electronic LTEs) are not factors to be considered here. We consider uniform all-to-all traffic grooming, in which all users in the network exchange data. We also consider a generalized case where there are multiple communication groups and each node in a group has to receive the data from all the other users in the same group. A user may belong to multiple groups. We consider two types of unidirectional rings, namely, single-hub and un-hubbed rings. In a single-hub ring [16], all the traffic has to be sent to the hub and then forwarded to the destinations by the hub. In an un-hubbed ring, there is no such hub.

All-to-all traffic can be implemented using two approaches, unicast and multicast. In unicast mode, traffic duplication can only be implemented in the electronic domain, whereas in multicast mode, traffic duplication can be done in the optical domain by using optical splitters. If a node needs to send a traffic stream on two outgoing links, the node requires two LTE ports in unicast mode but only one in multicast mode. All-to-all communication will benefit significantly in terms of the network cost by dividing it into multiple multicast sessions. However, it requires multicast capable nodes to be deployed in the network. The corresponding node architecture is referred to as the Tap-and-Continue node, and is introduced in [18], in which a node is capable of replicating optical signals in the optical domain using optical splitters, and one copy is dropped locally, while the remaining signal continues on the fiber without the additional cost of LTEs. Since each node on the unidirectional ring has only one incoming link and one outgoing link, and routing of all lightpaths is fixed along the direction of the ring, there is only one possible way to multicast - drop and forward. We use such node architecture to implement any multicast needed in the network.

Network coding [85] is a promising new technique that enables network nodes to perform algebraic operations on the multiple received packets besides simply forwarding them. It has been applied to a variety of network applications in order to improve the performances, such as in multi-hop wireless networks [92], network tomography [89], network protection [66, 90] and content distribution in peer-to-peer networks [91]. However, it is rarely applied to optical

networks for the cost saving objective. We will study two unidirectional ring networks, single-hub and un-hubbed, and investigate whether applying network coding will reduce the network cost to provision given traffic requests.

The paper is structured as follow. In Section 2.2, we introduce network coding and its benefit in saving LTE ports in optical networks. We will explore the network costs with or without applying network coding in single-hub and un-hubbed rings in Section 2.3 and 2.4, respectively. Numerical results of multiple many-to-many communication will be shown in Section 2.5. Finally in Section 2.6, we conclude the paper.

2.2 Network Coding in Optical Networks

Network coding is a novel technique which was originally proposed for improving network capacity, particularly in multicast scenarios[85, 86]. Besides the traditional routing functions, network nodes are designed to linearly combine packets arriving at input edges and transmit those combinations on output ports. By carefully choosing coding coefficients for each coding node in a network, where a coding node refers to a network node that has the capability of forming linear combinations of packets, a network can achieve the maximum multicast throughput for a given multicast request, which is equal to the min-cut max-flow of the network. In other words, given a traffic demand, employing a network coding scheme may provision the traffic by utilizing fewer network resources than the traditional routing scheme. Work has been done in recent years to explore the efficient coding schemes. For example, a classic polynomial time algorithm of codes construction for multicast traffic is proposed in [87] and a random coding scheme is proposed in [88].

In an optical network, traffic flow is carried and conveyed in a lightpath as an optical signal. To establish a lightpath, one LTE port is needed at the source to originate the signal and one is required at the destination to terminate it, and then the total number of LTE ports[2] required at each node is the total number of lightpaths terminated and originated at this node. Given a multicast service request, if network coding scheme can be employed to reduce the

[2]In rest of this paper, we use LTE to represent LTE port for short.

total bandwidth used, the total number of lightpaths required may be reduced, which will consequently result in a saving in the total network cost.

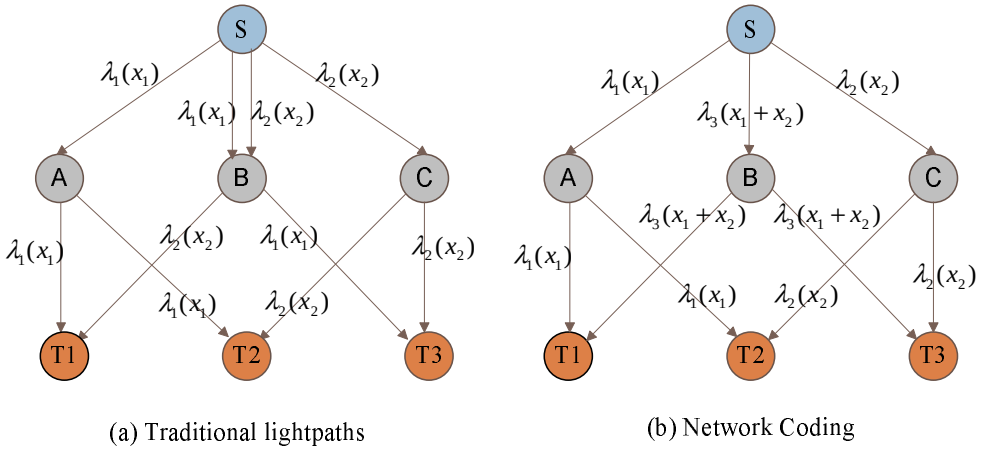


Figure 2.1 An example of LTE cost reduction through network coding

An example is shown in Figure 2.1 to illustrate the benefits of using network coding to save network cost in terms of LTE in a multicast traffic scenario. In the given network, one multicast session needs to be implemented, in which one source S generates two traffic flows λ_1 and λ_2 and both of them should be received by all three destination nodes T_1, T_2 and T_3. We also assume that each traffic flow takes an entire wavelength to accommodate. Thus, Figure 2.1.(a) shows the provisioning of the multicast session by using the traditional routing scheme. In order to achieve the multicast throughput, which is 2 in this case, four lightpaths have to be set up at source S and one of its outgoing links need to carry two traffic flows. In this case, links (S, A) and (S, C) carry one optical signal each and link (S, B) must carry two optical signals. We assume that each intermediate node is deployed with the capability of both multicasting and forwarding functionality in optical domain and thus no LTE is required at node A, B and C. Therefore, source node S requires four LTEs to establish the lightpaths and each destination node needs two LTEs to receive the traffic. Network coding is introduced in Figure 2.1.(b) to provision the same multicast service. Instead of generating four optical signals at source S in (a), only three lightpaths need to be set up as shown, in which two of them carry the original traffic flow λ_1 and λ_2 and the third one carries the encoded data

$\lambda_1+\lambda_2$ generated by performing bitwise operation "*Exclusive OR*" on the two original signals. In this case, node $T1$ will receive the traffic λ_1 and $\lambda_1+\lambda_2$ while node $T3$ receives λ_2 and $\lambda_1+\lambda_2$. Clearly, both nodes perform "XOR" operation on the received flow in order to recover the original traffic λ_2 and λ_1, respectively, and hence multicast service is fulfilled. Therefore, employing network coding in the example in Figure 2.1.(b) reduces the cost by one LTE at the source node S when compared to the cost required in 2.1.(a).

Although a recent study has introduces ways to conduct the network coding operation in optical domain[93, 94], our scheme is based on the current network infrastructure where network coding can only be implemented in an electronic domain, which requires the traffic that participates in the network coding operation to go through O-E-O conversion at the coding nodes. Such conversion may not be necessary if traditional routing is used. Thus, employing network coding may require more LTEs to terminate original traffic and regenerate encoded signals at coding nodes, hence making the problem a trade-off between reduction in the number of coding nodes and the reduction in LTEs in order to use fewer lightpaths in total. Moreover, applying network coding to optical networks also introduces new issues such as where and how should network coding be performed? The answer to this question is straightforward in a single-hub ring network. Since all traffic is collected by the hub, the hub is the perfect node to combine packets. However, it is not as clear in an un-hubbed ring. In addition, achieving network coding requires determining the coding scheme and the finite field size $GF(q)$ from which we choose coding coefficients. Of course, network coding does not come for free, and all the coding nodes should be equipped with the capability of coding, which introduces extra computation cost in the application layer. Compared to the cost of LTEs in physical layer, however, such resource consumption is almost negligible. Therefore, we only consider the number of LTEs as the network cost in this paper.

2.3 Cost Analysis in Single-hub Unidirectional Ring

In this part, we address the problem of grooming all-to-all traffic in a single-hub unidirectional ring and then extend the problem to address many-to-many traffic service. In many-to-

many communication, multiple groups are required to fulfill the all-to-all service where each group consists of two or more nodes. We consider two different network scenarios where groups are node-disjoint and when a node may belong to different groups. In each scenario, the cost of provisioning traffic is derived using two approaches: traditional routing and by applying network coding. For many-to-many communication, we prove that the traffic grooming problem is NP-complete and also propose an Integer Linear Programming (ILP) formulation to solve the problem optimally in the case where the groups are non-disjoint.

2.3.1 Uniform all-to-all traffic

Under all-to-all service, each node should receive data from all the other nodes on the ring. The problem can be stated as follows: Given a group of n nodes and grooming factor[3] g, each node i, generates and transmits traffic at constant rate, r, and must receive the traffic sent by other nodes such that the network resources, the LTEs in particular, are minimized.

We assume each data unit has to be transmitted to the hub before being relayed to the destination(s). As mentioned earlier, each node is equipped with optical splitters such that traffic can be duplicated in the optical domain. Moreover, we allow traffic bifurcation, which refers to splitting the traffic from a session over multiple lightpaths, since this may result in the minimum number of lightpaths and achieve the minimum overall cost in the case when g is not a multiple of r.

The all-to-all communication process involves two steps. The first step is to deliver traffic upstream from nodes to the hub. In the second step, the hub grooms the traffic into the minimum number of wavelengths and multicasts the groomed traffic downstream to every node on the ring. In the upstream direction, each node generates r units of low-rate traffic stream, which requires $\lceil r/g \rceil$ wavelengths to accommodate the data as well as $\lceil r/g \rceil$ LTEs to send it, and the hub needs $\lceil r/g \rceil$ LTEs to receive the traffic from one node.Either grooming the traffic from different nodes before sending it to the hub, or sending data to the hub directly by each node, will not change the total number of LTEs during the upstream process. This

[3]Grooming factor refers to the maximum number of low-rate traffic demands that can be multiplexed into one wavelength channel.

only effects the number of wavelengths used. However, the number of wavelengths is not a factor of the network cost according to our assumptions. Thus, there are $n\lceil r/g \rceil$ unidirectional lightpaths established from each node to the hub in the upstream process and the total cost includes the LTEs used by the nodes to transmit traffic and the hub to receive traffic, which is $n\lceil r/g \rceil + n\lceil r/g \rceil = 2n\lceil r/g \rceil$.

Let us consider downstream now. The total amount of traffic units collected at the hub is nr. Since traffic bifurcation is allowed, the minimum number of wavelengths can be used to pack all the traffic, denoted by $\lceil nr/g \rceil$, which is also equal to the number of LTE required by the hub to transmit and by each node to receive. Each node employs a tap-and-continue function which splits the optical signal and receives a small portion of power that is just enough to be detected and leave the rest of power to continue propagating on the ring. Such a power splitting function, performed by optical splitters, enables a broadcast service to be fulfilled by using only $n+1$ LTEs, one required at the hub and one required at each receiving node. Thus, broadcasting all of the traffic, which requires $\lceil nr/g \rceil$ wavelengths, will use a minimum cost of $(n+1)\lceil nr/g \rceil$ LTEs for the downstream traffic delivery.

To sum up, the resources consumed in both the upstream and downstream direction result in an overall minimum cost of $2n\lceil r/g \rceil + (n+1)\lceil nr/g \rceil$ LTEs to achieve all-to-all communication.

2.3.2 Application of Network Coding

When using network coding, it is obvious that the hub is a perfect place to perform network coding, since all data has to be delivered to the hub first and then converted into electronic signals for grooming, and hence no additional LTEs will be needed for O-E-O conversion to perform network coding. Therefore, the encoding operation is performed at the hub and the decoding is done at each node. We can also consider this problem in the upstream and downstream contexts. Since upstream is unicast and no network coding is needed, the number of LTEs required in the upstream process remains the same. In order to save sub-wavelength channels in the downstream data delivery, we use the following coding scheme. Since each node needs to receive data from different $n-1$ nodes, then without implementing network

coding but using splitters, each node has to receive all the data units from the hub, which is denoted by nr, in order to achieve minimum network cost. However, if a node receives linear combinations of the traffic instead of the original data, only $n-1$ linearly independent combinations are needed. By counting its own data in, each node has n linearly independent combinations, from which original data of all other nodes can be decoded, given the coding coefficients are known. This is the basic idea for using network coding to save lightpaths and hence LTEs.

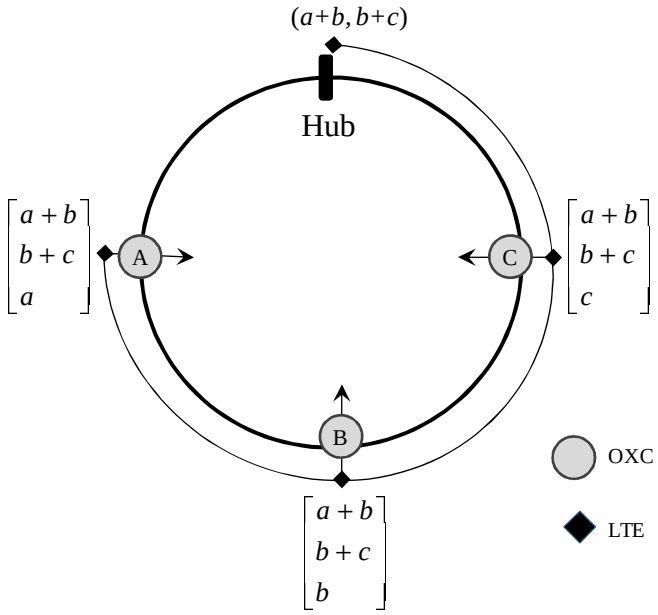


Figure 2.2 An application of network coding in a single-hub ring

The example shown in Figure 2.2 illustrates how to use network coding in the downstream process. We suppose that the hub has received the traffic from nodes A, B and C whose data units are denoted by a, b, c, respectively. The grooming factor is 2 and transmission rate at each node is 1. Hence, each wavelength is able to accommodate the traffic transmitted by two nodes. Instead of sending all the traffic a, b and c to each node on the ring, the hub encodes the data and generates code words $a+b$ and $b+c$ using modulo 2 addition and broadcast

them. Hence, node A will have combinations $a, a+b$ and $b+c$, where

$$\begin{bmatrix} a \\ a+b \\ b+c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Therefore, the coefficient matrix shown above has a full rank such that a, b and c can be decoded from the combinations. Following the same method, node B and C are also able to obtain all the original data a, b, c. Apparently, this coding scheme can be applied to the n-node case where the hub needs to generate $n-1$ linearly independent combinations which are also independent from all raw data units and broadcast them to each node.

Hence, in the case where the number of nodes is n and the traffic rate associated with each node is r, the cost of upstream transmission does not change from the case without network coding, which is denoted by $2n\lceil r/g \rceil$. In the downstream direction, however, the total traffic that the hub has to deliver is reduced to $(n-1)r$ which requires $\lceil (n-1)r/g \rceil$ wavelengths. The total cost of LTE ports in the downstream is $(n+1)\lceil (n-1)r/g \rceil$.

2.3.3 Multiple Many-to-Many Groups

In practice, the most common applications which use the all-to-all service mode are multimedia conferences and cooperative processing. Usually, there is more than one multimedia conference group simultaneously in the network. Thus, it is essential to consider multiple groups. In this scenario, the network nodes are divided into multiple groups, and within each group, nodes engage in all-to-all communication, while the traffic rates can be different in different groups. Therefore, the overall cost savings by the application of network coding in a single-hub ring with all-to-all traffic demand is $(n+1)(\lceil nr/g \rceil - \lceil (n-1)r/g \rceil) \geq 0$. The savings can be either in LTEs or network bandwidth, depending on the specific network scenarios as indicated above.

Disjoint Groups The definition of disjoint groups is straightforward - it refers to the case in which different groups do not share any common node. The problem of optimizing the

network cost in a disjoint group is stated as follows: minimize the number of LTEs used in a single-hub unidirectional ring, given m disjoint groups where each group i ($1 \leq i \leq m$) has n_i ($n_i \geq 2$) nodes and each node has to transmit $r_i \geq 1$ units of traffic to all the other nodes within the same group.

Let us consider the case without network coding first. The process is similar with the all-to-all case, which includes upstream and downstream process. The analysis still begins with the upstream process. Every group i is independent from each other such that the total number of LTEs used is the sum of LTEs consumed by each group individually. Following the analysis in Section 3.2.1, for each group i, the upstream consumes $2n_i\lceil r_i/g \rceil$ LTEs, resulting in $\sum_{i=1}^{m} 2n_i\lceil r_i/g \rceil$ in total.

In the downstream direction, the hub first grooms the traffic from the same group together. For group i, the number of wavelengths used to carry the traffic is $\lceil r_i n_i/g \rceil$, and $\lfloor r_i n_i/g \rfloor$ of them are filled up and sent back to the nodes belonging to the same group directly. Every lightpath used here is fully loaded and carries data for only one group. Thus, we only need to pay attention to the remaining portion of aggregate traffic for each group that cannot fill up a single wavelength if it exists, since it may need to be groomed with the traffic from other group(s) before delivery in order to save the lightpaths and hence LTEs. Each group has at most one piece of such traffic. This piece of traffic of group i is equal to $n_i r_i - g\lfloor r_i n_i/g \rfloor$. We denote this portion of the traffic by p_i, where $0 \leq p_i < g$. Thus, the problem can be stated formally as follow:

Problem GMP: Given m pieces of traffic, each piece i with p_i units and has to be received by the corresponding n_i nodes, groom and multicast the traffic at the hub such that the total LTE used is minimized.

NP-completeness In the optimal solution, any p_i must not be bifurcated and packed into more than one wavelength. We can prove it by contradiction. Assume that one piece p_i is split and assembled into two different wavelengths, which costs each node of group i two LTE ports to receive. This results in $2n_i$ LTE ports in total. If the hub uses a separate wavelength to send p_i, it only takes n_i LTEs at the nodes and 1 more LTE at the hub. Since

$2n_i - (n_i + 1) = n_i - 1 \geq 0$, it means that any optimal solution can be transformed to be the solution without traffic bifurcation for the problem. Therefore, this problem turns into a special case of general traffic grooming problem in a ring network, which has been proven to be NP-complete by reduction from *Bin Packing problem* in polynomial time in [11].

Solutions without Network Coding We now analyze the minimum network cost of many-to-many groups communication without employing network coding. The network cost of upstream transmission for n groups has been derived in 2), which is $\sum_{i=1}^{m} 2n_i\lceil r_i/g \rceil$. The cost of downstream transmission consists of two parts. The first part is the number of LTE ports used for broadcasting the lightpaths that are fully loaded for each individual group; the second part is the number of LTEs used for transmitting all the remaining traffic p_i. Since problem *GMP* is equivalent to the *Bin Packing problem*, which has been solved by many approaches in the literature, we consider two methods to obtain the minimum wavelengths in *GMP*. The first method is a heuristic algorithm based on First Fit Decreasing (FFD) [95]; In the second one, it is formulated by using Integer Linear Programming [96].

In the downstream transmission, for each group i, $\lfloor n_i r_i/g \rfloor$ wavelengths are fully utilized to pack the data, which takes $(n_i + 1)\lfloor n_i r_i/g \rfloor$ LTEs. To sum up, the total number of LTEs used to transmit this portion of the traffic is $\sum_{i=1}^{m}(n_i + 1)\lfloor n_i r_i/g \rfloor$. We use W to denote the number of wavelengths used at the hub to accommodate the p_i units of traffic of all the groups, which can be solved by either the heuristic or ILP described above. Hence, we need W LTEs at the hub to transmit and 1 LTE for each node of group i to receive p_i if it exists, which is determined by a binary number, $\lceil n_i r_i/g \rceil - \lfloor n_i r_i/g \rfloor$. Therefore, the total cost of transmitting all the p_i data units is equal to $W + \sum_{i=1}^{m}(\lceil n_i r_i/g \rceil - \lfloor n_i r_i/g \rfloor)n_i$.

Thus, the total network cost in both upstream and downstream for m groups without network coding is:

$$\sum_{i=1}^{m}\left(2n_i\left\lceil\frac{r_i}{g}\right\rceil + (n_i+1)\left\lfloor\frac{n_i r_i}{g}\right\rfloor + n_i\left(\left\lceil\frac{n_i r_i}{g}\right\rceil - \left\lfloor\frac{n_i r_i}{g}\right\rfloor\right)\right) + W. \tag{2.1}$$

Solutions with Network Coding To apply network coding to single-hub ring networks, the hub acts as an encoder and generates $n_i - 1$ code words for each group following the same

coding scheme described in Fig.2.2 with coefficients from $GF(2)$. The combinations of original data has the same rate as the original traffic, which is r_i. Thus, network coding can reduce the total traffic broadcast rate for each group i by r_i. The upstream transmission consumes the same amount of resources as in the case without network coding, but the total traffic rate turns out to be $(n_i-1)r_i$ units for each group i in the downstream process. Following the same computation rules used above, the network cost in this case also includes two parts. The first part is denoted by $\sum_{i=1}^{m}(n_i+1)\lfloor(n_i-1)r_i/g\rfloor$. In the second part, let W' denote the minimum number of wavelengths used at the hub to pack all the remaining traffic p_i' from each group after network coding.

Combining the cost spent in both upstream and downstream process gives us the total number of LTEs with the application of network coding in a many-to-many traffic scenario, which is expressed as:

$$\sum_{i=1}^{m}\left(2n_i\left\lceil\frac{r_i}{g}\right\rceil+(n_i+1)\left\lfloor\frac{(n_i-1)r_i}{g}\right\rfloor+n_i\left(\left\lceil\frac{(n_i-1)r_i}{g}\right\rceil-\left\lfloor\frac{(n_i-1)r_i}{g}\right\rfloor\right)\right)+W'. \quad (2.2)$$

2.3.4 Non-disjoint Groups

Intuitively, non-disjoint groups denote that in a network, communication groups are not node disjoint and hence some nodes may belong to multiple groups. It is clear that in single-hub ring network, non-disjoint group scenario is the generalization of disjoint group case. Therefore, it follows that the traffic grooming problem with non-disjoint groups is also *NP-complete* by reduction to the special disjoint group case.

Since the network cost is exactly the same for both cases with and without network coding in the upstream direction, we will just focus on the cost analysis of the downstream process. Assume after the hub grooms the data sent from the same group, each group i has a remaining traffic rem_i that may not be able to fill up a wavelength. Thus, the hub will groom rem_i from all the groups together. The difference between this case and the node-disjoint group case is that by grooming the rem_i of groups that share some common nodes, we save some LTEs at the common receiving nodes of these groups. Therefore, the hub may have preference to groom the rem_i of the groups that have more common nodes together.

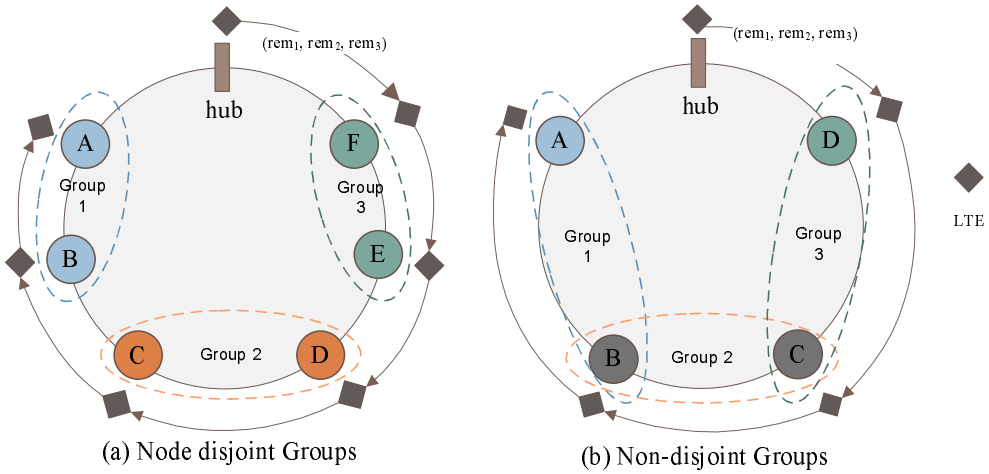


Figure 2.3 An cost comparison of disjoint and non-disjoint groups

Consider the case in which there are three groups on the ring and each group has two nodes as shown in Figure 2.3. In example (a) on the left side, three groups are node disjoint in which node A and B belong to group 1, C and D belong to group 2 and E, F belongs to group 3, respectively. We assume that each group has a remaining traffic rem_i and the hub grooms them together and broadcasts them back to all the nodes on one lightpath. Thus, a total of seven LTEs are required, one at the hub and one at each receiving node, to fulfill the multicast communication from the hub to all other nodes. However, if the groups share some common nodes as depicted in the case in Figure 2.3.(b) where groups 1 and 2 share node B and groups 2 and 3 have a common node C, although each group still has two member nodes, there are only a total of four distinct receivers for the multicast. If rem_1, rem_2 and rem_3 happen to be packed into one wavelength, only five LTEs are needed, one for transmitting at the hub and four for receiving at each node. Therefore, utilizing such feature of node sharing among distinct groups, the total number of LTEs required will be reduced if the hub node can groom the right rem_i together. The greater the number of nodes being shared by the groups whose rem_i are groomed together, the greater the saving in LTEs.

In order to achieve the optimal solution of the total number of LTE required in downstream direction, we formulate the problem by using integer linear programming. Given the network topology, traffic rate and number of nodes of each group, as well as the number of nodes shared

by any pair of groups, we could achieve the optimal solution for both cases with and without using network coding. The only difference between the two cases is the remaining traffic rem_i of each group i.

Without employing network coding in the network, for each group i, the total number of sub-wavelength circuits needed to broadcast at the hub is denoted by:

$$tr_i = \sum_{k=1}^{N} m_k^i r_i, \quad \forall i \leq M$$

where N is the total number of nodes, M denotes the total number of groups and binary m_k^i is equal to 1 if node k is a member of group i.

Whereas in the case where network coding is applied, the total number of circuits for each group i that needs to be broadcast at the hub is equal to:

$$tr_i = \left(\sum_{k=1}^{N} m_k^i - 1 \right) r_i, \quad \forall i \leq M$$

Accordingly, for both cases, the remaining portion of traffic flow for group i is denoted by

$$rem_i = tr_i - g \left\lfloor \frac{tr_i}{g} \right\rfloor, \ (0 \leq rem_i \leq g).$$

Therefore, the total number of LTEs required at the hub and nodes for remaining traffic communication can be obtained by solving the following ILP:

Table 2.1 Variables used in the ILP formulation for downstream process

Symbol	Meaning
W	the maximum number of the wavelengths used in the formulation
p_i^w	the remaining circuit from group i that is allocated into wavelength w at the hub
h^w	a binary variable that equals 1 if a LTE is required at the hub to transmit the remaining traffic carried on wavelength w
$s_{i,j}^w$	a binary variable that equals 1 if wavelength w carries circuits for both groups i and j where $i \neq j$; 0, otherwise
m_k^i	node k is a member of group i, a constant given in the problem

- **Objective:**

$$\text{Minimize} \quad \sum_{w=1}^{W} h^w + \sum_{k=1}^{N} \sum_{i=1}^{M} \left(m_k^i - \bigvee_{j=1}^{i} \left(\sum_{w=1}^{W} s_{i,j}^w m_k^i m_k^j \right) \right)$$

- **Subject to:**

$$h^w \geq \frac{1}{M}\sum_{i=1}^{M} p_i^w, \quad \forall w; \tag{2.3}$$

$$\sum_{i=1}^{M} p_i^w * rem_i \leq g, \quad \forall w; \tag{2.4}$$

$$\sum_{w=1}^{W} p_i^w = 1, \quad \forall i; \tag{2.5}$$

$$p_i^w + p_j^w - 1 \leq s_{i,j}^w \leq \frac{1}{2}(p_i^w + p_j^w), \quad \forall i < j \leq M,\ w \leq W. \tag{2.6}$$

The objective function consists of three terms. The first term denotes the total number of LTEs required at the hub. Since any $rem_i < g$, any node of group i needs at most one LTE to receive the flow for this group. Hence, the second term sums up the number of LTEs required for all the nodes of group i if $rem_i > 0$. In the parenthesis, the first term sums up the total number of nodes for each group i without considering node sharing. It means that we may count some LTEs redundantly if a node k belongs to multiple groups. Therefore, in the second term in the parenthesis, we subtract the portion of the cost counted redundantly. If multiple groups, say i and j, share one node k and their remaining traffic are packed into one wavelength, node k needs just one LTE to receive the traffic from both groups instead of two LTEs. Thus, one LTE should be subtracted. The notation $\vee_j^i \left(\sum_{w=1}^{W} s_{i,j}^w m_k^i m_k^j\right)$ denotes the disjunction operation over the terms from $j = 1$ to $j = i$.

Constraint (2.3) ensures that if wavelength w is used to accommodate traffic of any group, the corresponding lightpath should be set up at the hub such that one LTE is required. The wavelength capacity constraint is satisfied by equation (2.4) such that any wavelength can be overloaded. Constraint (2.5) makes sure that each remaining traffic rem_i should be allocated into a wavelength. Constraint (2.6) ensures that if the circuits of group i and j are groomed into the same wavelength w, then $s_{i,j}^w$ should be 1.

Therefore, adding up the LTEs cost obtained from ILP to the network cost of provisioning the rest of the traffic will yield the total number of LTEs required in this downstream process.

This is represented by

$$\sum_{i=1}^{M}(n_i+1)\left\lfloor\frac{tr_j}{g}\right\rfloor + LTE_{ILP}. \tag{2.7}$$

where LTE_{ILP} denotes the number of LTEs obtained by solving the ILP for the remaining traffic communication.

2.4 Cost Analysis in un-hubbed Unidirectional Rings

Following the same sequence of the previous section, we will first investigate the traffic grooming problem in an un-hubbed ring with all-to-all traffic and then generalize it to many-to-many group communication. All the assumptions made in the single-hub ring case remain except that a hub is not used.

2.4.1 Uniform all-to-all Traffic

The problem can be defined as follows: given a grooming factor g and a group of n nodes, each of which has r units of traffic, find the minimum number of LTEs required to fulfill all-to-all communication.

First, the lower bound and upper bound of the special case of unitary traffic, i.e. $r = 1$, can be derived if optical splitters are allowed. Each node needs at least $\lceil n - 1/g \rceil$ LTEs to receive and one LTE to send traffic. Thus, the *lower bound* of network cost is $(1+\lceil n-1/g\rceil)n$. However, if no traffic is groomed, the maximum number of LTEs requested is equal to n^2, which can be considered as the *upper bound* on the cost of this special case.

For the general case of arbitrary r, we propose a *multi-hub* approach to serve all-to-all demands while minimizing the total number of LTEs. The approach can be done in two steps:

1. Divide the nodes into a number of sub-groups such that the aggregated traffic of each sub-group is just enough to fill up a wavelength;

2. Choose one node of each sub-group as a hub to groom the traffic from other group members and broadcast the groomed traffic to all the nodes on the ring.

Unlike the situation of a single-hub ring where all nodes send traffic to the same hub, in an un-hubbed ring, once enough traffic is groomed to fill up a wavelength at a node, then this node will set up a lightpath and broadcast the data to the other $n-1$ nodes on the ring. This node is called a "hub" and we may have multiple hubs on the ring. The number of such hubs is equal to the number of wavelengths to accommodate all the traffic. The minimum number of wavelengths that can be achieved equals $\lceil nr/g \rceil$. In addition, we only consider the case where $r < g$, since if $r \geq g$, the excess traffic $(r - r \bmod g)$ of each node can fill up separate $\lfloor r/g \rfloor$ wavelengths without traffic grooming. In this case, we only need to consider the remaining traffic, denoted by $r - g\lfloor r/g \rfloor$, which is less than g. Thus, there is no need to consider the case where $r > g$.

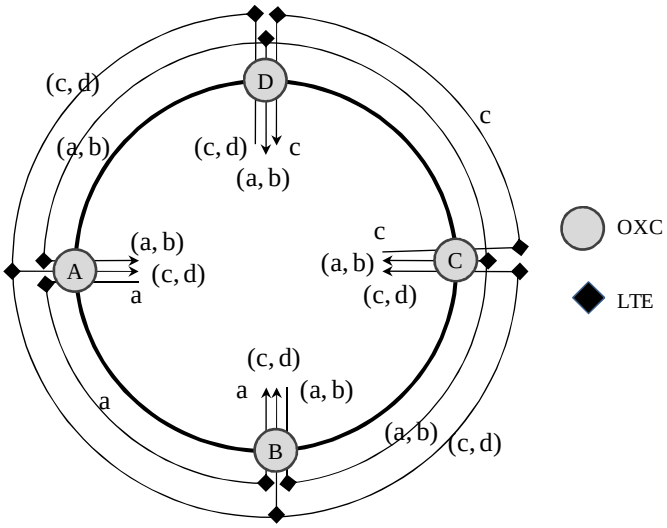


Figure 2.4 All-to-all transmission in un-hubbed unidirectional ring using the multi-hub approach

Figure 2.4 illustrates the all-to-all transmission mechanism proposed in multi-hub approach. In such an un-hubbed ring network, nodes A, B, C and D need to exchange information represented by a, b, c and d, respectively. In the example, the traffic rate of each stream is 2 and the grooming factor is $g = 4$. Each wavelength can accommodate traffic from two users. Thus, node A sends a to node B and then B packs both a and b into one wavelength and broadcasts the data to every node on the ring except itself. Such a process is called a *broadcast cycle*. Since

an optical splitter is used by each node to implement a drop-and-continue(DAC) functionality, each broadcast cycle consumes 6 LTEs, 2 for collecting data and 4 for broadcasting. Nodes C and D execute the similar process with node D being the hub. Thus, two broadcast cycles are needed to accomplish the communication such that the total number of LTEs required is $2 \times 6 = 12$.

We analyze the network cost in two different cases: when g is a multiple of r or not. In fact, the case when g is a multiple of r is a special case of the other case in terms of the solution. Thus we will only discuss the case when g is not a multiple of r in detail.

In the case when g is not a multiple of r, a wavelength cannot be filled up without traffic bifurcation. Minimizing the total number of wavelengths used for broadcasting with traffic bifurcation will result in the minimum number of broadcast cycles. However, each traffic split introduces two additional LTEs, because when a piece of traffic is allocated into two wavelengths instead of one, this requires two lightpaths to carry and results in two additional LTEs (one transmitter and one receiver). Alternatively, if we do not split any traffic, we cannot guarantee that the number of broadcast cycles is at a minimum. Each broadcast uses n LTEs, which means that saving one wavelength will save n LTEs. Therefore, there is a trade-off between the number of traffic splits and the total number of broadcast cycles. In the multi-hub approach, the minimum network cost is obtained by taking the minimum value of the solutions obtained from the two scenarios where we split and we do not split the traffic.

First, we consider the case without traffic bifurcation. Each wavelength can accommodate traffic from at most $\lfloor g/r \rfloor$ nodes, denoted by k. A small amount of bandwidth equal to, $g-kr$, is wasted on each wavelength. Then, the total number of broadcast cycles is $\lceil n/k \rceil$. Among them, each of $\lfloor n/k \rfloor$ broadcast cycles fully utilizes a wavelength with a small wastage of bandwidth, and each of such cycles requires $2(k-1)+n$ LTEs for transmitting and receiving, which results in a total $\lfloor n/k \rfloor (2(k-1)+n)$ LTEs for the first $\lfloor n/k \rfloor$ broadcast cycles. However, the number of nodes remaining in the last cycle is $(\lceil n/k \rceil - \lfloor n/k \rfloor)(n - k\lfloor n/k \rfloor)$ where $(\lceil n/k \rceil - \lfloor n/k \rfloor)$ is a binary number to indicate if there are some nodes left in the last cycle whose number is less than k and their aggregated traffic cannot fill up one wavelength. Hence, the number of LTEs

needed in the last broadcast cycle is $(\lceil n/k \rceil - \lfloor n/k \rfloor)(2(n - k\lfloor n/k \rfloor - 1) + n)$.

Therefore, the number of LTEs in this scenario is given by combining the cost in all broadcast cycles, namely:

$$\left\lceil \frac{n}{k} \right\rceil \left(3n - 2k \left\lfloor \frac{n}{k} \right\rfloor - 2\right) + \left\lfloor \frac{n}{k} \right\rfloor \left(2k \left\lfloor \frac{n}{k} \right\rfloor - 2n + 2k\right), \text{ where } k = \lfloor g/r \rfloor. \tag{2.8}$$

This general solution can also be applied to the case when g is a multiple of r.

In the second scenario where traffic bifurcation is applied, the minimum number of wavelengths to accommodate all the traffic can be achieved, which is given by $\lceil nr/g \rceil$, which also equals the minimum number of broadcast cycle. Let $\lceil nr/g \rceil = w_{min}$. Hence, the number of LTEs employed for a broadcast can by obtained by nw_{min}. However, we know that traffic splitting was used to achieve this due to the assumption that g is not a multiple of r. Since each traffic split uses two additional LTEs, the problem of minimizing the total number of LTEs actually turns out to be a problem of minimizing the number of traffic splits in order to groom the traffic on the minimum number of wavelengths, w_{min}. This problem has been solved by an iterative algorithm proposed in [11]. In each iteration, three steps are processed:

1. Fill each of w_{min} wavelengths with $\lfloor g/r \rfloor r$ loads. Therefore, the unused capacity left on each wavelength becomes $g' = g - \lfloor g/r \rfloor r$ and the number of nodes whose traffic has not been assigned is equal to $n' = n - \lfloor g/r \rfloor w_{min}$;

2. We have $n' < w_{min}$ and $r > g'$ from the first step. And then allocate g' units of traffic of each unassigned node to n' wavelengths, respectively, to fill them up.

3. As a result, only $w_{min} - n'$ wavelengths still have g' units of capacity available. Update $w_{min} := w_{min} - n'$, $r := r - g'$ and $n = n'$, and then repeat the three steps until all the traffic is assigned.

We use the algorithm here to obtain the minimum number of traffic splits in this situation, denoted by sp_{min}, given the minimum number of wavelengths used. The number of broadcast cycles determines the number of hubs in the ring, which is also equal to w_{min}. If no traffic split happens, collecting traffic at those hub nodes from other nodes before broadcast requires

$2(n - w_{min})$ LTEs. However, each traffic split increases the number of LTEs by 2. Thus, the total number of LTEs in this collection process is $2(n - w_{min} + sp_{min})$. In addition to the LTEs used for broadcasting, denoted by nw_{min}, the total number of LTEs used in this scenario with traffic bifurcation is $2(n - w_{min} + sp_{min}) + nw_{min}$.

Therefore, taking the minimum of the two solutions obtained in the two scenarios above will give us the overall minimum network cost. Thus, the number of LTEs of all-to-all traffic without network coding is:

$$\min\left\{\left\lceil\frac{n}{k}\right\rceil\left(3n - 2k\left\lfloor\frac{n}{k}\right\rfloor - 2\right) + \left\lfloor\frac{n}{k}\right\rfloor\left(2k\left\lfloor\frac{n}{k}\right\rfloor - 2n + 2k\right),\ 2(n - w_{min} + sp_{min}) + nw_{min}\right\}, \tag{2.9}$$

where $k = \lfloor g/r \rfloor$, $w_{min} = \lceil nr/g \rceil$, and sp_{min} is the minimum number of traffic splits obtained by the iterative algorithm proposed in [11], given w_{min}.

2.4.2 Application of Network Coding

In order to save network cost by performing network coding, a node should be chosen to collect all the original data. Thus, we propose a *one-hub* scheme in which only one node acts as a hub. The traffic is gathered and encoded at this node following the same coding scheme proposed in Section 2.3.2, where the network context is a single-hub ring. The hub can be selected from any node in the ring.

Hence, in the upstream direction, every node sending traffic to the hub consumes $2(n-1)$ LTEs. A total of $n-1$ linearly independent code words with traffic rate r are generated and packed into $\lceil r(n-1)/g \rceil$ wavelengths. In the downstream direction, the minimum number of broadcast cycles can be achieved, which also equals to $\lceil r(n-1)/g \rceil$. Each broadcast costs n LTEs such that the transmission in the downstream direction takes total $n\lceil r(n-1)/g \rceil$ LTEs.

Therefore, the total network cost with network coding using one-hub scheme is: $2(n-1) + n\lceil r(n-1)/g \rceil$.

Though *one-hub* scheme can save LTEs in downstream direction, it uses a few more LTEs in the upstream process compared to the *multi-hub* approach. Thus, the total number of LTEs consumed in both upstream and downstream directions may not always saved by employ-

ing one-hub scheme, depending on the specific network scenario. However, given the traffic demands, we can always use the multi-hub approach to solve the problem without applying network coding. Therefore, by comparing the solutions yielded by the one-hub and multi-hub approaches, we choose the solution of the minimum value.

Thus, assuming that LTE_{multi} denote the solution obtained from the multi-hub approach, the total network cost in un-hubbed rings with n nodes and all-to-all traffic demands r while applying network coding is:

$$\min\{2(n-1)+n\lceil r(n-1)/g\rceil,\quad \mathrm{LTE}_{\mathrm{multi}}\}. \tag{2.10}$$

2.4.3 Multiple Many-to-Many Groups

We now extend the all-to-all communication to multiple many-to-many disjoint groups on an un-hubbed unidirectional ring. We first consider disjoint groups. Since no node is shared by more than one group, there is no common hub being able to groom the traffic from different groups together. Thus, each group can be provisioned independently, and the total network cost is the sum of the cost of all groups.

Suppose there are m groups in an un-hubbed ring and each group i has n_i nodes with each node in the group sourcing r_i traffic units, for $1 \le i \le m$. The minimum network cost in terms of LTEs can be represented based on whether network coding is employed or not:

In the case where no network coding is employed, the total network cost in terms of the number of LTEs is:

$$\sum_{i=1}^{m}\mathrm{LTE}^i_{\mathrm{multi}}=\sum_{i=1}^{m}\min\left\{\left\lceil\frac{n_i}{k_i}\right\rceil\left(3n_i-2k_i\left\lfloor\frac{n_i}{k_i}\right\rfloor-2\right)+\left\lfloor\frac{n_i}{k_i}\right\rfloor\left(2k_i\left\lfloor\frac{n_i}{k_i}\right\rfloor-2n_i+2k_i\right),\right.$$
$$\left.2(n_i-w^i_{min}+sp^i_{min})+n_iw^i_{min}\right\} \tag{2.11}$$

where $k_i=\lfloor g/r_i\rfloor$, $w^i_{min}=\lceil n_ir_i/g\rceil$, and sp^i_{min} is the minimum number of traffic splits of group i given w^i_{min}.

In the case of using network coding, the total network cost in terms of the number of LTEs is:

$$\sum_{i=1}^{m}\min\left\{2(n_i-1)+n_i\lceil r_i(n_i-1)/g\rceil,\quad \mathrm{LTE}^i_{\mathrm{multi}}\right\}. \tag{2.12}$$

2.4.4 Non-disjoint Groups

We also consider group non-disjointness in an un-hubbed ring network where multiple groups may overlap. Two network scenarios, either employing network coding or not, will be discussed, respectively. We will first consider the case of applying network coding.

With Network Coding In order to perform network coding on the traffic for a group, one member node should be chosen to play the role of the hub as in hubbed-rings. The encoding operation remains the same as in the disjoint groups case and the total number of circuit needed to broadcast for each group i is: $tr_i = (\sum_k m_k^i - 1)r_i$, where m_k^i is equal to 1 if node k is a member of group i. The traffic requires $\lceil tr_i/g \rceil$ wavelengths to accommodate group i. Among them, $\lfloor tr_i/g \rfloor$ are filled up and the corresponding number of LTEs used to broadcast this portion of traffic for group i is $n_i \lfloor tr_i/g \rfloor$.

Let $p_i = tr_i - \lfloor tr_i/g \rfloor$ denote the remaining piece of traffic of group i, and each hub needs to broadcast p_i to the nodes within the same group. Since some nodes are shared between distinct groups, if one node can act as the hub for mutilple groups and the p_i of those groups happen to be packed into the same wavelength, LTE cost is reduced by sharing this hub. Notice that if two groups cannot share a hub node, though if their p_i are small enough to be allocated into one wavelength, we should not do it. The reason is as follow: given two groups, i and j, a node of group i needs one LTE to send the p_i to the hub node of group j in order to groom the traffic together, which results in another LTE at the receiving node. Thus, such grooming action results in two more LTEs for an extra delivery. In order to reduce the cost, we do not perform the traffic grooming in such case where two groups have no common node even if their remaining traffic can be packed into the same wavelength.

Since the cost of establishing the fully loaded lightpaths is fixed, we only need to concentrate on the cost of provisioning p_i of each group i. In order to minimize the number of LTEs used to broadcast all p_i, we propose a heuristic algorithm, *Hub Sharing Minimization (HSM)*, to allocate all the p_i based on the situation where the nodes are assigned to different groups, from which we can calculate how many LTEs can be saved by sharing the hub nodes and receiving

nodes, and consequently the total number of LTEs required. Prior to the description of *HSM*, we introduce some symbols used in the algorithm and the preliminary processes.

1. First, we create a bipartite graph $G = (V, E)$, where $V = X \bigcup Y$ and $X \bigcap Y = \emptyset$. Partition X represents the groups and partition Y represents the nodes. An undirected link $e \in E$ is established between $i \in X$ and $j \in Y$ if node j is a member of group i in the ring;

2. Each vertex $i \in X$ is assigned a weight value equal to p_i. The degree of vertex i is denoted by d_i and its neighbors, defined as the vertices in Y that connect to it, are denoted by N_i;

3. All the vertices in X will be gradually divided into multiple sets. Each set, denoted by $S_k, (1 \leq k \leq ||X||)$, is composed of a number of vertices that must have a common neighbor in Y, say j, and their corresponding traffic p_i are assigned in one wavelength together. We use $SN(S_k)$ to denote the total number of times that the vertices in S_k share common neighbors. Let us take an example as shown in Fig.2.5. If the first set of X is $S_1 = \{1, 2, 3\}$, and then $SN(S_1) = 2+1 = 3$, since vertex $1, 2$ and 3 share a common neighbor B in Y and vertex 2 and 3 also share node C in Y. We say that vertex B is shared twice while vertex C is shared once. Thus, the total number of sharing is equal to 3.

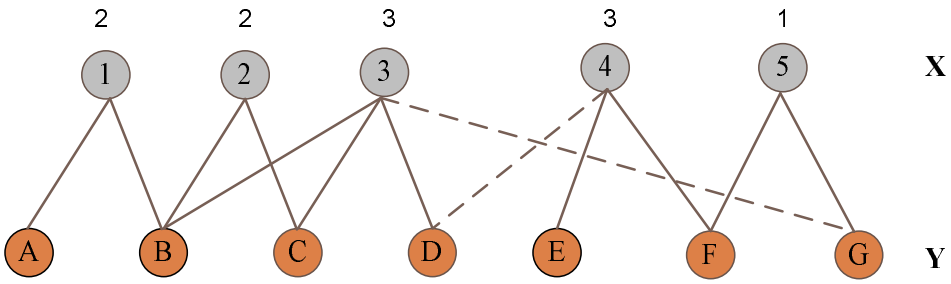


Figure 2.5 An example of the procedure of *HSM* in which $g = 8$

We assume that the ring network has been transformed to a bipartite graph $G = (V, E)$ described in 1) and the procedure of algorithm *HSM* is shown in Algorithm 1:

Algorithm 1: Heuristic algorithm HSM

Input: $G = (X \bigcup Y, E)$
Output: $G' = (X \bigcup Y, E')$

```
k = 1, ℂ = ∅;
while ℂ ≠ X do
    select j ∈ Y with max(d_j) where Σ_{i∈N_j} p_i ≤ g;
    if Num(j) > 1 then
        select the vertex j ∈ Y with max(SN(N_j));
        if Num(j) > 1 then
            select the vertex j ∈ Y with max(Σ_{i∈N_j} p_i);
        end
    end
    Let S_k = N_j and ℂ = ℂ ⋃ S_k;
    add S_k, ⋃_{i∈S_k} N_i and e ∈ E between them into G';
    remove S_k, ⋃_{i∈S_k} N_i and e ∈ E connected to them from G;
    k++;
end
```

Basically, heuristic *HSM* is a greedy based algorithm and the idea behind it is to divide all the groups into several sets, and each set represents a combination of group(s) that groom their traffic p_i into the same wavelength and share the same hub node. For each group set S_i, the algorithm tries to choose the groups that share the most number of nodes such that the greatest number LTEs can be saved at both hubs and receivers. which is equal to $SN(S_i)$.

The example shown in Figure 2.5 illustrates how *HSM* works and how LTEs can be saved. Figure 2.5 shows the bipartite graph G mapping from a ring topology, in which we assume there are five groups represented by the vertices $1, 2, 3, 4$ and 5 in partition X and each p_i is indicated by the value above the vertex. The vertices in partition Y represent the network nodes. Both solid and dash edges show which node belongs to which group in the ring. In the first iteration, vertex $B \in Y$ is picked with the largest degree, since this node is shared by the most number of groups. Since $p_1 + p_2 + p_3 = 7 < g$, groups $1, 2$ and 3, defined as set 1, groom their traffic together in one wavelength at the common hub node B and the total number of LTEs saved by selecting this set is equal to $SN(S_1)$, which is 4. Then the set 1 is separated from G by removing two dash lines, $(3, G)$ and $(4, D)$. In the second step, vertex F is picked and the vertices 4 and 5 are grouped as set 2. Since node F is the only common node for them,

only one LTE is saved at this node. Thus, the bipartite graph G is divided into two group sets and the total number of saved LTEs is equal to four.

We now derive the time complexity of this algorithm. We assume that the number of groups is denoted by $|X|$ whereas the number of nodes is denoted by $|Y|$ and each group has a constant number of members and each node is shared by a constant number of groups. Hence, each N_i and d_i is constant number. Thus, the complexity of the algorithm is dominated by the computation in line 3, since line 4-13 take a constant computation time, denoted by c. The worst scenario is when no group shares any common node and the computation cost of line 3 is $|Y|$ and the total number of iterations in the *while* loop is $|X|$. Therefore, the complexity of algorithm *HSM* is $O(|X|(|Y|+c)) = O(|X||Y|)$.

If we denote total number of saved LTEs obtained from *HSM* by h_s, we can then derive the total number of LTEs in both upstream and downstream processes:

In the upstream process, $\sum_i 2(n_i - 1)$ LTEs are required to collect the data at the hub of each group.

In the downstream process, without considering group sharing, each group i uses $n_i\lceil tr_i/g\rceil$ to broadcast the encoded data, which costs $\sum_i n_i\lceil tr_i/g\rceil$ LTEs in total. However, sharing member nodes among groups will save h_s LTEs given by the heuristic *HSM*.

Therefore, the total number LTEs used in this scenario is given by:

$$\sum_i 2(n_i - 1) + n_i \left\lceil \frac{tr_i}{g} \right\rceil - h_s, \text{ where } tr_i = \left(\sum_k m_k^i - 1 \right) r_i. \tag{2.13}$$

Without Network Coding In the scenario where no coding is employed, we can use the same approach employed to derive the cost for node disjoint case without application of network coding, described in Equation (2.11), to obtain the cost for this scenario. The only difference between two cases is that we can further reduce the cost of LTEs in this scenario by sharing the common hubs among distinct groups if they have common nodes by using the algorithm *HSM* to allocate all the remaining traffic p_i for all the groups, where p_i is derived as follows:

For any group i,

- In the case where no traffic bifurcation is applied, $p_i = r_i(n_i - k_i\lfloor n_i/k_i \rfloor)$, where $k_i = \lfloor g/r_i \rfloor$.
- In the case where traffic bifurcation is applied, $p_i = tr_i - \lfloor tr_i/g \rfloor$, where $tr_i = \sum_k m_k^i r_i$.

Thus, we assume the number of LTEs saved by sharing the common hubs among different groups is denoted by h_s and h'_s, solved by *HSM*, for the cases without and with traffic bifurcation, respectively. By subtracting h_s and h'_s from the costs in node disjoint group case for both scenarios and taking the minimum value, we obtain the number of LTEs required for the many-to-many communication in this non-disjoint group case:

$$\min \left\{ \sum_{i=1}^{m} \left(\left\lceil \frac{n_i}{k_i} \right\rceil \left(3n_i - 2k_i \left\lfloor \frac{n_i}{k_i} \right\rfloor - 2 \right) + \left\lfloor \frac{n_i}{k_i} \right\rfloor \left(2k_i \left\lfloor \frac{n_i}{k_i} \right\rfloor - 2n_i + 2k_i \right) \right) - h_s, \right.$$
$$\left. \sum_{i=1}^{m} \left(2(n_i - w_{min}^i + sp_{min}^i) + n_i w_{min}^i \right) - h'_s \right\}, \quad (2.14)$$

where $k_i = \lfloor g/r_i \rfloor$, $w_{min}^i = \lceil n_i r_i / g \rceil$, sp_{min}^i is the minimum number of traffic splits of group i given w_{min}^i.

2.5 Numerical Results

We first summarize the all the traffic grooming problems in various network scenarios, their corresponding complexity and the proposed solutions in Table 2.2. Based on the solutions, we compare the results of two different cases - with or without network coding - in various network scenarios and under different traffic conditions. Since all-to-all communication is a special case of multiple many-to-many group cases where the number of groups is equal to 1, in this section we consider more general cases in which $m \geq 1$.

Each network scenario is generated according to the following assumptions:

1. The number of many-to-many groups in the ring network is fixed at $m = 10$;
2. The number of nodes per group is lower bounded by 2 and upper bounded by a given positive integer value n_{max} and is uniformly distributed in $[2, n_{max}]$;

3. The traffic rate of each node is lower bounded by 1 and upper bounded by a given positive integer value r_{max} and is uniformly distributed in $[1, r_{max}]$;

Table 2.2 The summary of the problems addressed in the paper

		All-to-All (Uniform)	**Many-to-Many** (Disjoint)	**Many-to-Many** (Non-Disjoint)
Single-hub	complexity	Polynomial	NP-hard	NP-hard
	solutions	Optimal	Optimal (ILP) Heuristic (FFD)	Optimal (ILP)
Un-hubbed	complexity	Conjectured to be NP-hard[11][13]	Conjectured to be NP-hard	NP-hard
	solutions	Bounds Heuristic	Bounds Heuristic	Heuristic

2.5.1 Single-hub Rings

Since network cost in the upstream direction in single-hub networks is always the same whether or not network coding is used, we only compare the cost factors in the downstream direction.

Table 2.3 The comparison of downstream network cost of disjoint groups in the single-hub ring with g=4, 8 and 16

		(Traffic,nodes)	(2,20)	(2.5,20)	(2,30)	(2.5,30)	(2,40)	(2.5,40)	(2,50)	(2,60)
g=4	FFD	Without NC	36.8	42.6	71.2	84.8	122.4	146.0	184.2	258.0
		With NC	25.5	26.7	57.5	64.5	98.3	115.4	158.3	223.6
	ILP	Without NC	36.8	42.6	71.2	84.8	122.4	146.0	184.2	258.0
		With NC	25.5	26.7	57.5	64.5	98.3	115.4	158.3	223.6
g=8	FFD	Without NC	25.5	26.7	46.1	51.4	73.2	84.0	101.0	144.3
		With NC	23.0	23.6	40.3	43.2	61.5	70.0	91.8	126.9
	ILP	Without NC	25.5	26.7	46.1	51.4	73.2	84.0	101.0	144.3
		With NC	23.0	23.6	40.3	43.2	61.4	70.0	91.8	126.8
g=16	FFD	Without NC	23.0	23.5	34.3	35.4	49.2	54.8	66.7	88.3
		With NC	21.9	22.0	33.0	33.7	44.1	47.8	55.9	81.2
	ILP	Without NC	23.0	23.5	34.3	35.4	49.2	54.7	66.6	88.3
		With NC	21.9	22.0	33.0	33.7	44.0	47.8	55.8	81.2

Table 2.4 The comparison of downstream network cost of non-disjoint groups in the single-hub ring with g=8 and 16

	(Traffic,nodes)	(2,20)	(3,20)	(2,30)	(3,30)	(2,40)	(3,40)	(2,50)	(3,50)	(2,60)
g=8	Without NC	20.2	27.9	39.5	52.6	64.9	102.4	91.0	147.5	128.3
	With NC	14.8	23.4	34.4	45.4	56.2	85.2	82.3	115.2	134.2
g=16	Without NC	15.2	18.1	30.3	34.1	39.8	57.7	56.1	77.3	75.3
	With NC	11.0	12.2	25.5	28.7	33.3	48.9	50.7	68.0	69.3

Table 2.3 shows the heuristic and exact solutions of the number of LTE ports in different network scenarios. Three different grooming factors, g, are used. Each network scenario is represented by a pair of numbers in the parenthesis. The first number denotes the average traffic rate transmitted at each node, derived from $(1+r_{max})/2$ and the second number denotes the average total number of nodes in the ring network, derived from $(2 + n_{max})m/2$ where $m = 10$ based on the properties of uniform distribution. For instance, given a scenario where $r_{max} = 3$ and $n_{max} = 4$, this maps to the case where the average traffic rate and average total nodes are represented by $(2, 30)$ in the table. In each scenario, upper bounds r_{max} and n_{max} are fixed, but the actual n_i and r_i for each group $i \leq m$ are randomly chosen between the lower bounds and upper bounds for each single experiment. Each network cost value is obtained by taking the average result of 500 independent experiments associated with a unique network scenario.

Under the same network scenarios and grooming factors, the table shows that the exact network cost is almost the same as its heuristic counterpart in most cases. There are only a few cases that the heuristic solutions are a little bit greater than the optimum due to the small difference between the exact and heuristic solutions of the problem *GMP*.

We can observe that the network cost increases in proportion of the traffic rates and the number of nodes. Network coding can save network cost in all cases where the grooming factor $g = 4, 8$ and 16. The relative savings of the network cost, denoted by the ratio of cost savings to the network cost without applying network coding, are almost the same under different network traffic conditions with the same g. However, the incremental saving of the network cost decreases as g increases. The overall relative cost saving under all the network conditions

considered in the examples is between $10\% - 20\%$, which translates to a large CAPEX[4] saving considering the cost of LTEs.

Almost the same cost saving ratio is achieved in the non-disjoint group case. The corresponding network cost factors are shown in the Table 2.4. The case where groups may have common nodes results in further reduction in the overall cost compared to the case where each group has disjoint network nodes. However, the results demonstrates that the advantages of employing network coding are not affected.

2.5.2 Un-hubbed Rings

The network costs of un-hubbed unidirectional rings under various network scenarios are shown in Table 2.5 with $g = 4$, 8 and 16. By inspecting the table, network cost saving increases as the total amount of traffic transmitted increases with the same g. However, the saving achieved by employing network coding decreases with the increase of the grooming factor, since the greater the grooming factor, the fewer lightpaths are set up for the given communication requests. Hence, the fewer lightpaths are saved by applying network coding, which results in less saving of LTEs.

Table 2.5 The comparison of total network cost of disjoint groups in the un-hubbed ring with g=4, 8 and 16

	(Traffic,nodes)	(2,20)	(2,30)	(2.5,30)	(2,40)	(2.5,40)	(2,50)	(2.5,50)	(2,60)	(2.5,60)
g=4	Without NC	40	81	84.3	145.2	154.1	212.5	233.7	300.3	337.1
	With NC	40	80.1	83.5	138.8	149.6	207.2	229.7	290.6	329
	Lower Bound	40	75.9	80.3	125.7	139.8	187.8	214.7	262.9	308
	Upper Bound	40	95	95	182.2	182.2	290.1	290.1	425	425
g=8	Without NC	40	72.4	73.8	117	121.3	159.3	170.7	218.3	236.7
	With NC	40	71.2	72.2	113	117.3	156.5	167.1	209.5	228.8
	Lower Bound	40	63.7	65.7	95.6	101.7	131.6	144.6	175.9	198.9
	Upper Bound	40	95	95	182.2	182.2	290.1	290.1	425	425
g=16	Without NC	40	69.1	69.1	103.7	106.6	136.8	142.1	176.9	186.1
	With NC	40	69.1	69.1	100.8	102.7	134.9	139.7	172.5	180.9
	Lower Bound	40	59.4	59.4	80.6	83.4	106.6	113.4	136.5	146.9
	Upper Bound	40	95	95	182.2	182.2	290.1	290.1	425	425

[4]CAPEX refers to Capital Expenditure.

Table 2.6 The comparison of total network cost of non-disjoint groups in un-hubbed ring with g=8 and 16

	(Traffic,nodes)	(2,20)	(3,20)	(2,30)	(3,30)	(2,40)	(3,40)	(2,50)	(3,50)	(2,60)	(3,60)
g=8	Without NC	35.6	39.0	71.4	77.8	113.8	130.0	158.0	184.9	217.7	255.7
	With NC	35.6	39.0	69.8	75.2	109.3	125.8	152.6	181.2	211.1	249.2
g=16	Without NC	35.4	36.4	65.1	68.4	100.5	105.4	132.6	143.3	171.2	190.1
	With NC	35.4	36.4	64.3	66.5	97.5	101.4	130.1	140.1	167.3	185.0

Notice that when the number of nodes in each group is exactly 2 and the total number of nodes on the ring is 20, the network cost is a constant regardless of the grooming factor and whether or not network coding is employed, because each node needs two LTEs, one as transmitter and one as receiver, to implement all-to-all communication without necessity of network coding and traffic grooming. Except for this case, the overall cost savings under other different network scenarios considered is between 1-5%, which is less significant than the saving obtained by using network coding in single-hub rings.

In addition, we also obtain the network cost for the case of non-disjoint groups in the ring. As shown in Table 2.6, the network cost for each network scenario is slightly less than its counterpart in the previous case where groups do not share any node, which illustrates the fact that sharing hub nodes among different groups does reduce the overall network cost. However, the benefit that network coding could achieve remains almost the same as that in the disjoint group case.

The reason for the difference between single-hub rings and un-hubbed rings is that in an un-hubbed ring, all the traffic does not need to be transmitted to the same hub when using the *multi-hub* approach. Once a wavelength is filled up at a node, the data is broadcast. However, network coding requires a common hub on the ring to collect the data from all the nodes within the same group in the *one-hub* scheme. Even if a wavelength is fully loaded, it has to experience an extra delivery to the common hub. Such extra delivery consumes more LTEs in un-hubbed rings. Only if the number of nodes and traffic rate of a group satisfies certain conditions, can network coding save costs for this group. This means that not every group with an all-to-all traffic demand will benefit from network coding in un-hubbed rings. Therefore, the total saving - the sum of the saving from each group - will not be as high as

that in single-hub rings.

2.6 Conclusions

In this paper we provided the first study of the traffic grooming problem of two types of unidirectional rings, single-hub and un-hubbed, with uniform all-to-all, and its extension, many-to-many, traffic scenarios, and with or without network coding. We considered the number of LTEs as the dominant factor of network cost. Traffic bifurcation and optical splitters are allowed in our analysis of network costs in all network scenarios.

Applying network coding into WDM ring networks introduces several issues: the first issue is the selection of coding nodes and coding coefficients and this has been addressed in our proposed schemes. Second, the encoding process may cause delay due to the synchronization between original traffic signals to be combined at the hubs. However, the typical end-to-end delay in optical networks is very small due to the dedicated lightpaths provisioned for any end-to-end communication and thus network coding will not bring down the overall delay performance. Third, network coding introduces extra computation cost in application layer due to the encoding and decoding process performed in the electronic domain. However, compared to the saving in CAPEX cost of a LTE, which reaches tens of thousands of dollars, the additional computation cost is much less and almost negligible.

Based on the solution, we observe that network coding can save the number of LTEs for a given group only upon a condition, which is $\lceil (n-1)r/g \rceil < \lceil nr/g \rceil$, in single-hub rings. However, this condition can also apply to unhubbed rings if we use one-hub approach to provision traffic. Without explicitly reducing the number of LTEs, using network coding can always reduce the total amount of traffic communicated among group members. In a single-hub ring, we explored the minimum cost of all-to-all traffic in the cases when network coding was not applied and when it was applied, and from numerical results, we observed 10-20% cost savings with the deployment of network coding in two many-to-many communication scenarios, in which different groups are allowed and not allowed to have common nodes. In the un-hubbed unidirectional ring, we proposed the *multi-hub* scheme to derive the network cost

when network coding is not applied and *one-hub* scheme if network coding is applied for many-to-many communication. We also consider a more general case where different communication groups are not node disjoint. An heuristic algorithm, *HSM*, is proposed to address this issue. We evaluated the network cost using this algorithm and compared the cost under the two cases where network coding is applied or not. Based on the numerical results for different network scenario, the savings of LTEs by employing network coding is 1-5%, which is less significant than the saving in the single-hub ring case.

CHAPTER 3. TWO-LINK FAILURE PROTECTION IN WDM MESH NETWORKS WITH *P*-CYCLES

A paper accepted for publication in *Elsevier Computer Networks* [37]

Taiming Feng[1] [2], Long Long [3] [4], Ahmed E. Kamal[3] and Lu Ruan[1]

Abstract

In WDM networks, it is important to protect connections against link failures due to the high bandwidth provided by a fiber link. Although many *p*-cycle based schemes have been proposed for single-link failure protection, protection against two-link failures have not received much attention. In this paper, we propose *p*-cycle based protection schemes for two-link failures. We formulate an ILP model for the *p*-cycle design problem for static traffic. We also propose two protection schemes for dynamic traffic, namely SPPP (Shortest Path Pair Protection) and SFPP (Short Full Path Protection). Simulation results show that SFPP is more capacity efficient than SPPP under incremental traffic. Under dynamic traffic, SPPP has lower blocking than SFPP when the traffic load is low and has higher blocking than SFPP when the traffic load is high.

3.1 Introduction

Network survivability is an important requirement for WDM optical networks due to their ultra-high capacity. Various protection schemes have been developed for WDM networks.

[1]Department of Computer Science, Iowa State University, Ames, IA 50011.
[2]The major contributions are the the theoretical analysis and dynamic traffic protection.
[3]Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011.
[4]The major contributions are the theoretical analysis and static traffic protection.

Ring-based protection schemes enable traffic recovery to be completed in 50-60 ms, but require at least 100% capacity redundancy. On the other hand, mesh-based protection schemes are much more capacity efficient, attributed to diverse traffic routing and protection capacity sharing among different connections. However, more complicated protection switching process leads to much longer recovery time. p-Cycle is a promising protection technique as it achieves the speed of ring with the efficiency of mesh [47], [40]. p-Cycles are established by configuring the spare capacity into pre-cross-connected cycles. Upon a link failure, protection switching is performed at the two end nodes of the failed link. Therefore, traffic recovery is extremely fast. p-Cycle is also efficient in protection since it protects both on-cycle links and straddling links. As shown in Fig. 3.1, a-b-c-d-f-a is a p-Cycle. For the on-cycle span a-b, the p-cycle provides one protection path a-f-d-c-b. For the straddling span a-c, the p-cycle provides two protection paths: a-b-c and a-f-d-c. Thus, p-cycle can protect one unit of working capacity on every on-cycle span and protect two units of working capacity on every straddling span. Another advantage of using p-cycles is that any traffic pattern can be handled in a similar way, in which working traffic is provisioned first and p-cycles are then formed to cover all the links need to be protected without interfering with the routing of the working paths.

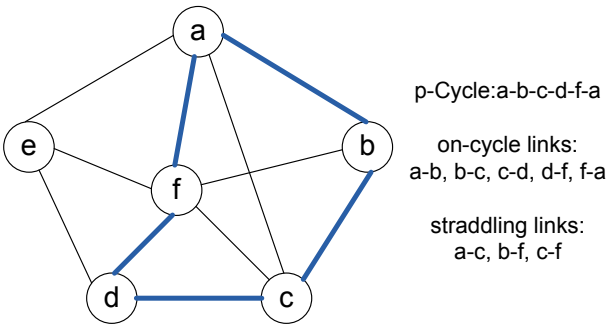


Figure 3.1 An Example of p-Cycle

The concept of p-cycle was first proposed in [39] and subsequently many works in literature study the p-cycle design problem for protecting against single-link failures. Most of these works assume the demands have been routed and seek to find the optimal set of p-cycles to protect the given working capacity [39], [41], [48]. The joint optimization of the working path routes

and the p-cycles is studied in [43]. An extension of the basic p-cycle concept called Failure Independent Path-Protecting (FIPP) p-cycle is proposed in [45], which leads to more capacity efficient network designs than link protecting p-cycle. Recently, the author of [66] introduced a new $1 + N$ protection scheme against single-link failures by combining network coding and p-cycles.

Although single-link failures are the most common failure scenarios, double-link failure can occur in some cases. First, after a link fails, a second link may fail while the first link is being repaired. Second, two fiber links may be physically routed together for some distance and a backhoe accident may lead to the failures of both links [53]. Third, if an optical switch with two links connected to it fails, then both links fail. Double-link failure protection has been addressed in some works. In [52], a p-cycle based scheme for double-link failure protection is proposed where p-cycles are reconfigured based on the remaining spare capacity after a link failure occurs and the corresponding working paths are rerouted. This scheme cannot deal with simultaneous two-link failures. In the scheme proposed in [53], two link-disjoint backup paths are computed for each link so that the network is two-link failure survivable. The similar scheme was also proposed in [50] and the problem was formulated as an Integer Linear Program. The scheme is slow in recovery because the backup paths are configured after link failure occurs. In [55], a p-cycle based multi-QoP (quality of protection) framework with five QoP service classes is proposed, where the platinum class is assured protection from all two-link failures. The protection for a platinum demand is achieved by routing it entirely over straddling links. There are also some work addressing multiple-link failure protection. The authors of [54] proposed algorithms to find k disjoint p-cycles to protect each link such that the network is k link-failure survivable. The author of [56] extended his work in [66] to protect multiple-link failures by using network coding and p-cycles.

In this paper, we consider the problem of protecting connections against two simultaneous link failures. Our basic idea is to use two p-cycles with link-disjoint protection segments to protect each working link. Since p-cycles are preconfigured using the spare capacity in the network, extremely fast recovery can be achieved. We formulate an ILP model for the p-cycle

design problem for static traffic model in which the set of connections to be established is given a priori. We also propose two protection schemes for dynamic traffic. In the dynamic traffic model, connection requests arrive at the network one by one and the network knows nothing about the bandwidth requirement, source and destination node of incoming requests. Thus, primary and backup lightpaths need to be computed online according to the utilization of the current network resources.

The rest of this paper is organized as follows. In Section 3.2, we present two theorems about double-link failure protection. An ILP model for the p-cycle design problem for static traffic is given in Section 3.3. In Section 3.4, we propose two double-link failure protection schemes for dynamic traffic. Numerical results are presented in Section 3.5 and conclusions are given in Section 3.6.

3.2 Preliminaries

We use a directed graph $G = (V, E)$ to represent a WDM optical network. A bidirectional communication link between nodes u and v are represented by two directed edges $u \rightarrow v \in E$ and $v \rightarrow u \in E$. Connections are unidirectional and each connection requires one unit of capacity (i.e., the capacity of a wavelength). We use unidirectional p-cycles to protect connections. A unidirectional p-cycle consumes one unit of capacity on each unidirectional on-cycle link; it can protect one unit of working capacity on any straddling link and any link in the opposite direction of an on-cycle link.

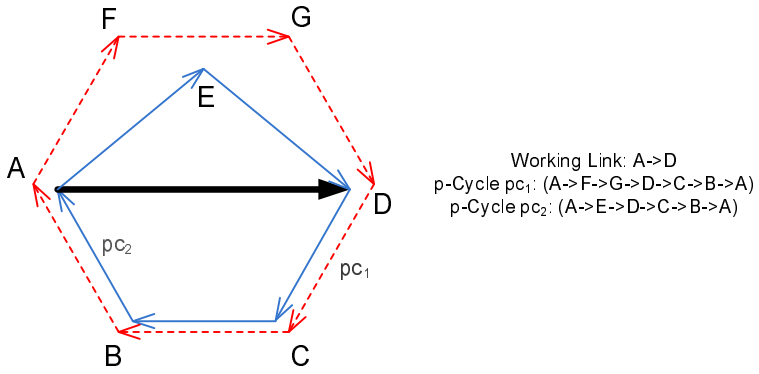


Figure 3.2 Two-Link Failure Protection for Link $A \rightarrow D$

In [54], two link-disjoint p-cycles are computed to protect a working link against two link failures. However, we do not have to enforce the link-disjoint requirement on the two p-cycles in order to protect a link against two link failures. In fact, when a link e is protected by a p-cycle pc, only part of the p-cycle is used for protection. We name the part of pc that carries the traffic when e fails as the *protection segment* for e on pc, which is denoted by $pc(e)$. Fig. 3.2 shows two p-cycles pc_1 and pc_2, both of which can protect link $A{\rightarrow}D$. $pc_1(A{\rightarrow}D){=}(A{\rightarrow}F{\rightarrow}G{\rightarrow}D)$ is the protection segment for link $A{\rightarrow}D$ on pc_1 and $pc_2(A{\rightarrow}D){=}(A{\rightarrow}E{\rightarrow}D)$ is the protection segment for link $A{\rightarrow}D$ on pc_2. Although pc_1 and pc_2 are not link-disjoint (they share links $D{\rightarrow}C$, $C{\rightarrow}B$, and $B{\rightarrow}A$), they can still protect link $A{\rightarrow}D$ against two link failures since $pc_1(A{\rightarrow}D)$ and $pc_2(A{\rightarrow}D)$ are link-disjoint.

The following theorem gives the sufficient condition for the traffic on a working link to be protected against any two-link failure.

Theorem 1. *A working link $A{\rightarrow}B$ can be protected against any two-link failure if there exist two p-cycles pc_1 and pc_2 such that the following conditions are met.*

1. *pc_1 can protect link $A{\rightarrow}B$;*
2. *pc_2 can protect link $A{\rightarrow}B$;*
3. *$pc_1(A{\rightarrow}B)$ is link-disjoint with $pc_2(A{\rightarrow}B)$.*

Proof. The three conditions ensure that there are three link-disjoint paths from A to B: one is the direct link from A to B, the other two are $pc_1(A{\rightarrow}B)$ and $pc_2(A{\rightarrow}B)$. When any two links in the network fail, there must exist at least one path from A to B that is intact. Therefore, link $A{\rightarrow}B$ is protected against any two-link failure. □

According to Theorem 1, we can use two protection-segment-disjoint p-cycles to protect a working link against two link failures. However, using two p-cycles to protect each working link requires a large amount of protection capacity. To reduce the capacity requirement, we allow two working links to share the protection of a common p-cycle. Let e_1 and e_2 be two working links. Let S_1 and S_2 be a pair of protection-segment-disjoint p-cycles for e_1 and e_2,

respectively. If $|S_1 \cap S_2| = 1$, then e_1 and e_2 share one p-cycle. If $|S_1 \cap S_2| = 2$, then e_1 and e_2 are protected by the same pair of p-cycles. When two links share one or two p-cycles, it is possible that the failure of these two links will leave one or both of them unprotected. In this case, we say the sharing is *invalid*. On the other hand, we say the sharing is *valid* if the two links are still protected when both of them fail simultaneously. In the following, we present a theorem that gives the sufficient condition for a valid sharing.

Theorem 2. *Let e_1 and e_2 be two working links that share one or two p-cycles (i.e., $S_1 \cap S_2 \neq \emptyset$). The sharing is valid if the following conditions are met.*

1. *For link e_1, there exists a p-cycle $pc_1 \in S_1$ such that $e_2 \notin pc_1(e_1)$.*
2. *For link e_2, there exists a p-cycle $pc_2 \in S_2$ such that $e_1 \notin pc_2(e_2)$;*
3. *$pc_1(e_1)$ is link-disjoint with $pc_2(e_2)$ if $pc_1 = pc_2$.*

Proof. Upon a two-link failure, we consider the case that only one of e_1 and e_2 fails, say e_1, so e_2 is still working and only e_1 needs to be protected. Based on Theorem 1, there must still exist one protection segment for e_1 that is not affected by another link failure, since e_1 is protected by two link-disjoint segments. Thus, both e_1 and e_2 can survive regardless of protection sharing.

Thus, we only need to focus on the case where both e_1 and e_2 fail. Conditions 1 and 2 ensure that both $pc_1(e_1)$ and $pc_2(e_2)$ are not affected by the failures. If $pc_1 \neq pc_2$, then e_1 and e_2 are protected by pc_1 and pc_2, respectively. In this case, the sharing is valid. If $pc_1 = pc_2$, then $pc_1(e_1)$ has to be link-disjoint with $pc_1(e_2)$ described in condition 3. Otherwise, one unit protection capacity provided by a p-cycle is not enough to protect two failed link at a time. Thus, $pc_1(e_1)$ and $pc_1(e_2)$ have to be link-disjoint to validate the sharing. □

Fig. 3.3 shows two examples of p-cycle sharing. In the example shown in Fig. 3.3(a), two working links e_1=A→B and e_2=C→D are protected by the same pair of p-cycles, pc_1 and pc_2, where both e_1 and e_2 are straddling links of pc_1 and on-cycle links of pc_2. That is, $S_1 = S_2 = \{pc_1, pc_2\}$. When both e_1 and e_2 fail, pc_2 can protect neither of them since

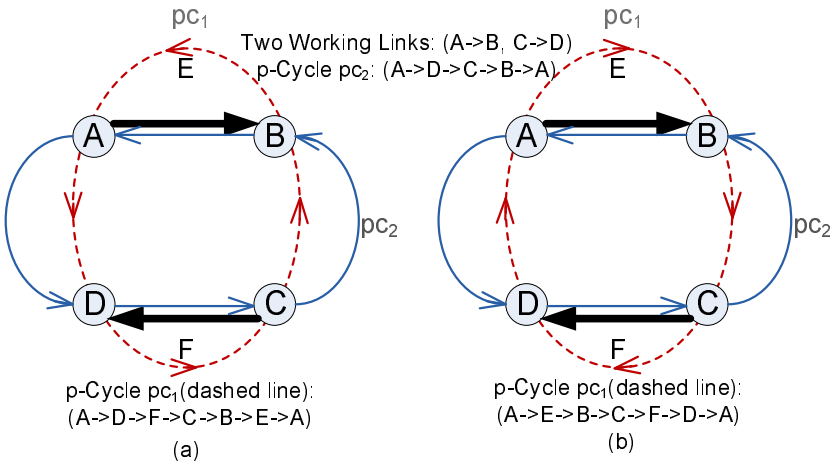


Figure 3.3 p-Cycle Sharing in Two-Link Failure Protection

$e_2 \in pc_2(e_1)$ and $e_1 \in pc_2(e_2)$. pc_1 can be used to protect either e_1 or e_2 but not both because $pc_1(e_1)=(A \rightarrow D \rightarrow F \rightarrow C \rightarrow B)$ and $pc_1(e_2)=(C \rightarrow B \rightarrow E \rightarrow A \rightarrow D)$ are not link-disjoint. Therefore, e_1 and e_2 cannot validly share the p-cycles pc_1 and pc_2. We now consider the example shown in Fig. 3.3(b), where everything is the same except that the direction of p-cycle pc_1 is reversed. In this case, $pc_1(e_1) = (A \rightarrow E \rightarrow B)$ does not contain e_2, $pc_1(e_2) = (C \rightarrow F \rightarrow D)$ does not contain e_1, and $pc_1(e_1)$ and $pc_1(e_2)$ are link-disjoint. According to Theorem 2, e_1 and e_2 can validly share pc_1 and pc_2.

3.3 An ILP Model for Static Traffic Protection

In this section, we present an ILP model for the following p-cycle design problem: given a network $G = (V, E)$, the working capacity d_{ab} on each link $(a \rightarrow b) \in E$, and the maximum number of p-cycles needed, compute a set of p-cycles to protect the working capacity against two-link failures such that the total capacity required by the p-cycles is minimized.

In the input parameters, P is the upper bound of the number of p-cycles needed and can be computed according to the static demands. Suppose the static demands need M links with one unit of capacity reserved on each link, a total of $2M$ p-cycles will be required in the worst case to protect any double-link failures since each link requires two p-cycles. In the results obtained by solving an ILP, variables e^p_{mn} $(\forall (m \rightarrow n) \in E)$ identify the configuration of those computed

p-cycles. Given a p-cycle pc_p, if all the corresponding variables e^p_{mn} equal 0, this p-cycle is not used to protect any link in the solution.

Input Parameters:

P	the maximum no. of p-cycles in the solution.
p	p-cycle index where $p \in \{1, 2, \ldots, P\}$.
d_{ab}	integer, total amount of working capacity on link $a \rightarrow b$.

Variable Notations:

e^p_{mn}	binary variable, 1 if p-cycle p uses link $m \rightarrow n$ as an on-cycle link.
z^p_n	binary variable, 1 if node n is on p-cycle p.
$x^p_{ab\_k}$	binary variable, 1 if p-cycle p protects the k^{th} working capacity on link $a \rightarrow b$.
$f^{p,(ab\_k)}_{mn}$	binary variable, 1 if p-cycle p protects the k^{th} working capacity on link $a \rightarrow b$ and the protection segment traverses link $m \rightarrow n$.
$v^{p,(ab\_k)}_{cd}$	binary variable, 1 if p-cycle p protects the k^{th} working capacity on link $a \rightarrow b$ and the protection segment does not use link $c \rightarrow d$ or $d \rightarrow c$.
$AB^{p,(ab\_k)}_{cd\_l}$	binary variable, it equals $\lvert v^{p,(ab\_k)}_{cd} - v^{p,(cd\_l)}_{ab} \rvert$.
$C^{p,(ab\_k)}_{cd\_l}$	binary variable, used in the absolute value constraints for $AB^{p,(ab\_k)}_{cd\_l}$.

Objective:

$$\text{Minimize} \sum_{p} \sum_{(m,n) \in E} e^p_{mn}$$

The objective function sums the total capacity used by all the active p-cycles.

1) Capacity Constraints:

$$\sum_{p} x^p_{ab\_k} \geq 2, \quad \forall (a \rightarrow b) \in E, \ \forall k \leq d_{ab}; \tag{3.1}$$

$$\sum_{k} x^p_{ab\_k} \leq 1, \quad \forall p, \ \forall (a \rightarrow b) \in E; \tag{3.2}$$

Equation (3.1) ensures that each working unit on a link should be protected by at least two p-cycles. Equation (3.2) ensures that a unidirectional p-cycle can protect only one unit capacity on a given link.

2) Cycle Constraints:

$$\sum_{(m \rightarrow n) \in E} e^p_{mn} = \sum_{(n \rightarrow l) \in E} e^p_{nl} = z^p_n, \ \forall p, \ \forall n \in V; \tag{3.3}$$

$$e^p_{mn} + e^p_{nm} \leq 1, \quad \forall p, \ \forall (m \rightarrow n) \in E; \tag{3.4}$$

Equations (3.3) is the flow conservation constraint for any simple cycle by ensuring that the in-degree and out-degree of every on-cycle node is 1. Equation (3.4) ensures that each unidirectional p-cycle p cannot traverse the same link twice in both directions.

3) Link Protection Constraints:

$$\sum_m f_{mn}^{p,(ab\_k)} - \sum_l f_{nl}^{p,(ab\_k)} = \begin{cases} x_{ab\_k}^p & \text{if } n = b \\ -x_{ab\_k}^p & \text{if } n = a \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

$$\forall p,\ \forall (a{\to}b) \in E,\ \forall n \in V,\ \forall k \le d_{ab};$$

$$\sum_m f_{ma}^{p,(ab\_k)} = \sum_n f_{bn}^{p,(ab\_k)} = 0, \quad \forall p,\ \forall (a{\to}b) \in E,\ \forall k \le d_{ab}; \tag{3.6}$$

$$f_{mn}^{p,(ab\_k)} \le e_{mn}^p, \quad \forall (a{\to}b), (m{\to}n) {\in} E,\ (a{\to}b) {\neq} (m{\to}n),\ \forall p,\ \forall k \le d_{ab}; \tag{3.7}$$

Equation (3.5) and (3.6) are the flow conservation constraints for each protection segment provided by a p-cycle p to protect the k^{th} working capacity on link $a{\to}b$. In this case, a unit of protection flow should be reserved from node a to b using the on-cycle links of p. But there is no incoming flow of source node a and outgoing flow of b along the protection segment. Equation (3.7) ensures that any link $(m{\to}n)$ used by a protection segment should be an on-cycle link of the protection p-cycle.

4) Protection Segment Disjointness Constraints:

$$f_{mn}^{p,(ab\_k)} + f_{mn}^{q,(ab\_k)} \le 1, \tag{3.8}$$

$$f_{mn}^{p,(ab\_k)} + f_{nm}^{q,(ab\_k)} \le 1, \tag{3.9}$$

$$\forall (a{\to}b), (m{\to}n) {\in} E, (a{\to}b) \neq (m{\to}n) \text{ or } (n{\to}m),\ \forall p, q, p{\neq}q,\ \forall k \le d_{ab};$$

Any link should be link-disjoint with its protection segment in any direction, which is guaranteed by Equation (3.8) and (3.9).

5) Absolute Value Constraints:

$$v_{cd}^{p,(ab\_k)} = (x_{ab\_k}^{p} - f_{cd}^{p,(ab\_k)} - f_{dc}^{p,(ab\_k)}), \tag{3.10}$$

$$\forall (a{\to}b),(c{\to}d) \in E,\ (a{\to}b) \neq (c{\to}d),\ \forall p, \forall k \leq d_{ab};$$

$$AB_{cd\_l}^{p,(ab\_k)} \geq v_{cd}^{p,(ab\_k)} - v_{ab}^{p,(cd\_l)}, \tag{3.11}$$

$$AB_{cd\_l}^{p,(ab\_k)} \geq -(v_{cd}^{p,(ab\_k)} - v_{ab}^{p,(cd\_l)}), \tag{3.12}$$

$$AB_{cd\_l}^{p,(ab\_k)} \leq v_{cd}^{p,(ab\_k)} - v_{ab}^{p,(cd\_l)} + 2C_{cd\_l}^{p,(ab\_k)}, \tag{3.13}$$

$$AB_{cd\_l}^{p,(ab\_k)} \leq -(v_{cd}^{p,(ab\_k)} - v_{ab}^{p,(cd\_l)}) + 2(1 - C_{cd\_l}^{p,(ab\_k)}), \tag{3.14}$$

$$\forall (a{\to}b),(c{\to}d) \in E,\ (a{\to}b) \neq (c{\to}d),\ \forall p,\ \forall k \leq d_{ab}, l \leq d_{cd}.$$

Equation (3.10) defines $v_{cd}^{p,(ab\_k)}$, which will be used for protection sharing between link ab_k and cd_l. The absolute value $AB_{cd\_l}^{p,(ab\_k)}$ is defined by equations (3.11) - (3.14). Eq. (3.11) and (3.12) make sure that the absolute value is always greater and equal to 0. However, they are not enough. When both v variables equal each other, the absolute value should be 0. But it can still be either 0 or 1. In order to make it equal 0, we have to introduce a new binary variable, $C_{cd\_l}^{p,(ab\_k)}$. Eq. (3.13) and (3.14) ensure that when both v variables equal 0 or 1 at the same time, the absolute value AB can only be 0 by randomly choosing C as either 0 or 1. Meanwhile, equation (3.11) and (3.12) are not violated.

6) p-Cycle Sharing Constraints:

$$\begin{aligned} & f_{mn}^{p,(ab\_k)} + f_{mn}^{p,(cd\_l)} + v_{cd}^{p,(ab\_k)} + v_{ab}^{p,(cd\_l)} \\ \leq & \sum_{p} v_{cd}^{p,(ab\_k)} + \sum_{p} v_{ab}^{p,(cd\_l)} + \sum_{p} AB_{cd\_l}^{p,(ab\_k)} + 1, \end{aligned} \tag{3.15}$$

$$\forall (a{\to}b),(c{\to}d){\in}E,\ (a{\to}b){\neq}(c{\to}d),\ \forall p,\ \forall (m{\to}n){\in}E,\ \forall \{k,l\}{\leq}d_{ab}.$$

Constraint (3.15) ensures that all p-cycle sharing are valid based on Theorem 2. It takes the following three cases into the consideration when both link $(a{\to}b)$ and $(c{\to}d)$ fail simultaneously.

If $\sum_p AB_{cd\_l}^{p,(ab\_k)} \geq 1$, link $(a{\to}b)$ and $(c{\to}d)$ can be protected by two different p-cycles, because there exist at least one p such that $|v_{cd}^{p,(ab\_k)} - v_{ab}^{p,(cd\_l)}| = 1$. Assume that $v_{cd}^{p,(ab\_k)} = 1, v_{ab}^{p,(cd\_l)} = 0$, then p can be used to protect link $(a{\to}b)$ without traversing link $(c{\to}d)$. Mean-

while, there must exist another p' that can protect link $(c{\rightarrow}d)$ without traversing link (a, b), since there are two link-disjoint protection segments for $(c{\rightarrow}d)$.

If $\sum_p AB_{cd\_l}^{p,(ab\_k)} = 0$, both link $(a{\rightarrow}b)$ and $(c{\rightarrow}d)$ share the same two p-cycles. There are two cases to be discussed as follows:

1) If $\sum_p v_{cd}^{p,(ab\_k)} + \sum_p v_{ab}^{p,(cd\_l)} = 4$, both link $(a{\rightarrow}b)$ and (c, d) are straddling links of the two protection p-cycles. In this case, one of the two p-cycles can protect $(a{\rightarrow}b)$ and the other one can protect $(c{\rightarrow}d)$ when both links fail.

2) If $\sum_p v_{cd}^{p,(ab\_k)} + \sum_p v_{ab}^{p,(cd\_l)} = 2$, one of the protection p-cycles actually traverses both links and cannot be used for protection anymore. Thus, only one p-cycle p can be used to protect them. In this case, we must have $f_{mn}^{p,(ab\_k)} + f_{mn}^{p,(cd\_l)} + v_{cd}^{p,(ab\_k)} + v_{ab}^{p,(cd\_l)} \leq 3$ to ensure that the protection segment $p(a{\rightarrow}b)$ and $p(c{\rightarrow}d)$ are link-disjoint.

Constraint (3.15) combines all three cases together to ensure that all p-cycle sharing are valid. Note that the condition $\sum_p v_{cd}^{p,(ab\_k)} + \sum_p v_{ab}^{p,(cd\_l)} \geq 2$ always holds. Thus, if $\sum_p AB_{cd\_l}^{p,(ab\_k)} \geq 1$, the sharing is valid. There is no need to address the remaining two cases and Eq. (3.15) should not be violated. If $\sum_p AB_{cd\_l}^{p,(ab\_k)} = 0$, Eq. (3.15) ensures that one of the last two cases will occur.

Current objective is to minimize the spare capacity required by the p-cycles. However, this objective could be easily modified to achieve different goals. For instance: 1. If the goal is to minimize the maximum total capacity (working and spare capacity) used on any link, we can introduce a new variable ζ and a new constraint: $\zeta \geq d_{mn} + \sum e_{mn}^{p},\ \forall(m{\rightarrow}n) \in E$. In this newly added constraint, ζ is the maximal aggregated amount of capacity reserved by working capacity and protection p-cycles on every link. Accordingly, the new objective will be: *Minimize* ζ. 2. In the second case, if the total amount of capacity provided by each link $(m{\rightarrow}n)$ is upper bounded by a number, denoted by λ_{mn}, we can introduce a new constraint: $\lambda_{mn} \geq d_{mn} + \sum e_{mn}^{p},\ \forall(m{\rightarrow}n) \in E$ to ensure that the total capacity reserved on each link will not exceed the limit. Therefore, our ILP model can be modified in a flexible fashion to adapt to various network scenarios with different design goals.

3.4 Protection Schemes for Dynamic Traffic

In this section, we study the problem of two-link failure protection for dynamic traffic. We assume that the working path for a connection is given. The problem is to compute a set of p-cycles to protect the working path against any two-link failure so that the total capacity required by the p-cycles is minimized. We present two heuristic algorithms for this problem. Both algorithms are designed to achieve efficient protection by employing p-cycle sharing. The notations and some functions used in the algorithms are explained in Table 3.1.

Table 3.1 Notations used in the algorithms

Notations	Meaning
$\mathbb{P}$	The set of links used by the working path of a given connection
pc_p	A p-cycle indexed by p
$pc_p(e)$	the protection segment on pc_p that actually protect link e
$\mathbb{C}$	The set of all the existing p-cycles in the network
$\mathbb{C}(e)$	The set of existing p-cycles that can protect link e
$\mathbb{C}_{temp}$	The set of protection segments that protect a set of links
$cycle\_build(e, n)$	Construct n new cycles that can protect e
$check\_share(pc_i, pc_j, e)$	Check whether p-cycle pc_i and pc_j can protect link e simultaneously in Algorithm 2
$check\_disjoint(pc_i, e)$	Check whether an existing p-cycle pc_i can protect a working link e in Algorithm 3
$check\_share2(e, e', pc_i)$	Check whether link e and e' can share the protection of pc_i validly in Algorithm 3

3.4.1 Shortest Path Pair Protection Scheme

We propose the Shortest Path Pair Protection (SPPP) scheme in this section. Given the working path $\mathbb{P}$ of a connection, SPPP computes a set of p-cycles to protect $\mathbb{P}$ as follows. For each link on $\mathbb{P}$, we compute two p-cycles to protect the link so that the two p-cycles are protection-segment-disjoint. Whenever possible, we reuse the p-cycles that have been previously provisioned to minimize the total protection capacity.

Fig.3.4 illustrates how SPPP protects a working path from s to d that traverses link 1 through link 4. For each link on the working path, SPPP computes two p-cycles with link-

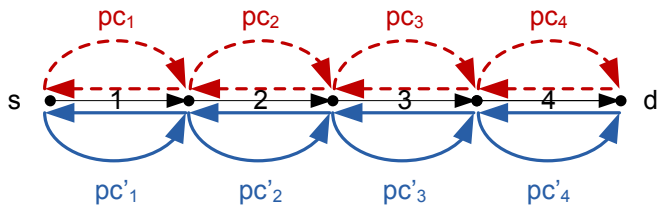


Figure 3.4 p-Cycles Used in SPPP Scheme.

disjoint protection segments to protect the link. As shown in the figure, pc_i and pc'_i are used to protect link i, for $1 \leq i \leq 4$. To save capacity, we allow a p-cycle to be shared by different working links if sharing is allowed according to Theorem 2. For example, suppose link 3 can share pc_2 with link 2, then $pc_3 = pc_2$ and only one new p-cycle (i.e., pc'_3) needs to be created for link 3; suppose link 4 can share pc_1 with link 1 and can share pc'_2 with link 2, and $pc_1(link4)$ is link-disjoint with $pc'_2(link4)$, then $pc_4 = pc_1$, $pc'_4 = pc'_2$, and no new p-cycle needs to be created for link 4.

SPPP uses a boolean function $check\_share(pc_1, pc_2, e)$ to check whether two p-cycles can be used to protect a working link e. The checking procedure consists of three steps. First, it checks whether e can be protected by pc_1 and has valid sharing with other links also protected by pc_1; Second, check whether e can be protected by pc_2 with valid sharing; Third, whether $pc_1(e)$ and $pc_2(e)$ are link-disjoint. $check\_share(pc_1, pc_2, e)$ returns true if all three steps are passed. The rules are actually based on Theorem 1 and 2. Given a working link e, the set of existing p-cycles that can protect e is denoted by $\mathbb{C}(e)$. That is, $\mathbb{C}(e)$ contains all existing p-cycles that have e as an on-cycle link or a straddling link. For each link $e \in \mathbb{P}$, SPPP computes two p-cycles for e as follows:

We try to use as many existing p-cycles as possible to maximize the sharing. We first check whether there exist two p-cycles in $\mathbb{C}(e)$ such that they can be reused to protect e. If so, no new p-cycle needs to be created for e. This check can be done by using the $check\_share$ function. If two p-cycles cannot be found in $\mathbb{C}(e)$, we try to reuse at least one p-cycle. By checking the protection condition of a p-cycle, say pc_i, we need to construct the second p-cycle pc_j for e such that $pc_i(e)$ and $pc_j(e)$ are link-disjoint. If $check\_share(pc_i, pc_j, e)$ returns true, then e is

protected against double-link failure. Finally, if none of the p-cycles in $\mathbb{C}$ can be reused to protect e, then we construct two new p-cycles for e such that the protection segments for e on these two p-cycles are link-disjoint.

The function $cycle\_build(e, n)$ is used to construct new cycles where n is the number of newly constructed cycles. If $n = 1$, the function finds the shortest path that is link-disjoint with e using Dijkstra's algorithm and then combines the same link e in the opposite direction to form a cycle, which can be used to protect e. If $n = 2$, we first use Bhandari's algorithm [57] to obtain a pair of link-disjoint paths between the two end nodes of e with minimum total length by temporarily marking it invalid. We can obtain two p-cycles for e by combining each path with e in the reverse direction. Clearly, these two p-cycles can provide link-disjoint protection segments for e.

Algorithm 2: SPPP Scheme

Input: Working Path $\mathbb{P}$, the set of existing p-cycles, $\mathbb{C}$
Output: A updated set $\mathbb{C}$

for $\forall e \in \mathbb{P}$ **do**
  flag=2;
  if $\exists\{pc_i, pc_j\}\in\mathbb{C}(e)$ *s.t.* $check\_share(pc_i, pc_j, e)==true$ **then**
    flag=0;
    $\mathbb{C}(e) = \mathbb{C}(e) - \{pc_i, pc_j\}$;
  end
  else if $(\exists pc_i \in \mathbb{C}(e)) \wedge (pc_j = cycle\_build(e, 1)) \wedge (check\_share(pc_i, pc_j, e))$ **then**
    flag = 1;
    $\mathbb{C}(e) = \mathbb{C}(e) - \{pc_i\}$;
    $\forall e' \neq e$ that can be protected by pc_j, $\mathbb{C}(e') = \mathbb{C}(e') \bigcup \{pc_j\}$;
    $\mathbb{C} = \mathbb{C} \bigcup \{pc_j\}$;
  end
  if *flag==2* **then**
    construct pc_1 and pc_2 for e by running $cycle\_build(e, 2)$;
    $\forall e' \neq e$ that can be protected by pc_1, $\mathbb{C}(e') = \mathbb{C}(e') \bigcup \{pc_1\}$;
    $\forall e' \neq e$ that can be protected by pc_2, $\mathbb{C}(e') = \mathbb{C}(e') \bigcup \{pc_2\}$;
    $\mathbb{C} = \mathbb{C} \bigcup \{pc_1, pc_2\}$;
  end
end

The pseudo-code of the SPPP scheme is shown in Algorithm 2. The input is a working path $\mathbb{P}$ and the set of the existing p-cycles, the output is a new set $\mathbb{C}$ of p-cycles that protect

$\mathbb{P}$. The algorithm computes two p-cycles for each link $e \in \mathbb{P}$ in the for loop from line 1 to line 19. The first step of the procedure is executed by Lines 3-6, in which we try to find a pair of existing p-cycles to protect a given link e. The second step is conducted by Lines 7-12, in which we reuse a p-cycle pc_i and construct a new one, pc_j to provide protection. The final step is to construct two new p-cycles such that their protection segments are link-disjoint, which is realized by Lines 13-18. It is worth noting that each p-cycle can only protect a link once. Once a link e has been protected by a given p-cycle, say pc_i, this p-cycle is valid to protect e any more in the future. This is the reason that in Lines 5 and 9, we need to remove the p-cycles that have been used to protect e from $\mathbb{C}(e)$.

We now analyze the time complexity of SPPP. First, we check the running time of function $check\_share(pc_i, pc_j, e)$ is $O(|E||V|^2)$ because it needs to check each working link protected by pc_i to see if it can share pc_i with e, and the time of the checking process is $O(|V|^2)$. For each $e \in \mathbb{P}$, the algorithm computes two p-cycles for e. The time of this computation is dominated by the computation in line 3. We assume $|\mathbb{C}(e)|$ is upper bounded by a constant. Since line 3 is executed for each edge in the working path $\mathbb{P}$ and the number of edges in $\mathbb{P}$ is upper bounded by $|V|$, the complexity of SPPP is $O(|E||V|^3)$.

The advantage of the SPPP scheme is that it can save plenty of protection capacity by exploiting p-cycle sharing. However, SPPP always creates short p-cycles, which are less efficient than long p-cycles as shown in [58] since short p-cycles tend to have less straddling links. In the next, we present another protection scheme that makes use of long p-cycles for connection protection.

3.4.2 Shortest Full Path Protection Scheme

In this section, we present the Shortest Full Path Protection (SFPP) Scheme. Given the working path $\mathbb{P}$ of a connection, SFPP computes a set of p-cycles to protect $\mathbb{P}$ as follows. First, we compute one short p-cycle for each link on $\mathbb{P}$. Next, we compute a long p-cycle that contains all links on $\mathbb{P}$ and is link-disjoint with the protection segments of all the working links computed in the first step. Clearly, the long p-cycle can protect every link in $\mathbb{P}$. Therefore,

each working link is still protected by two p-cycles (one short and one long) with link-disjoint protection segments. Like SPPP, SFPP reuses existing p-cycles whenever possible to save protection capacity.

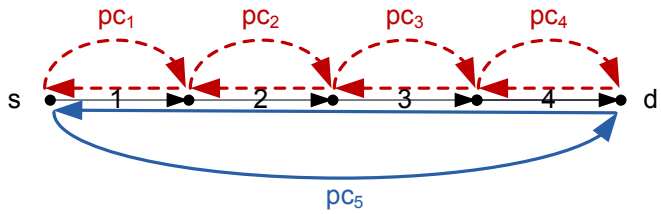


Figure 3.5 p-Cycles used in SFPP Scheme.

Fig. 3.5 illustrates how SFPP protects a working path from s to d that traverses link 1 through link 4. Four short p-cycles, pc_1 to pc_4, are first found to protect link 1 to link 4. These short p-cycles can be shared by the working links. For example, if link 3 can share pc_1 with link 1, then $pc_3 = pc_1$ and no new short p-cycle needs to be created for link 3. In the second step, we find a long p-cycle, labeled as pc_5, to cover the entire working path. pc_5 must be link-disjoint with the protection segments $pc_1(link1)$ to $pc_4(link4)$ to ensure that each working link is protected by two protection-segment-disjoint p-cycles.

We now explain the details of SFPP. We first find one short p-cycle for every link e on the working path $\mathbb{P}$. During this process, existing p-cycles will be reused if sharing is possible. Specifically, when we process link e, we first check whether there is a p-cycle in $\mathbb{C}(e)$ that can be reused to protect e. A p-cycle pc_i can be reused to protect e if 1) pc_i does not contain any edge $e' \neq e \in \mathbb{P}$, and 2) for every link $e' \neq e \in \mathbb{P}$ that is protected by pc_i, $pc_i(e)$ and $pc_i(e')$ are link-disjoint. The first condition is needed because if pc_i contains e', then pc and the long p-cycle will not be protection-segment-disjoint since they both contain e'. The second condition is needed for the following reason. When both e' and e fail, the long p-cycle can protect neither of them since the protection segment of one link contains the other link. So, both links have to be protected by pc_i. We define a function $check\_disjoint(pc_i, e)$ to check whether those two conditions given a link e and an existing p-cycle pc_i. It returns true if both conditions are satisfied.

If the function returns false, a new p-cycle should be constructed for e with the requirement that it does not contain any edge $e' \neq e$ in $\mathbb{P}$. After each link $e \in \mathbb{P}$ has been protected by a p-cycle, we construct a long p-cycle as follows. We first mark all the links in $\mathbb{P}$ as invalid as well as all links on the protection segments (provided by the short p-cycles) of all the working links in $\mathbb{P}$. We can simply substitute e in the function $cycle\_build(e, n)$ by $\mathbb{P}$ and then run $cycle\_build(\mathbb{P}, 1)$ to form a long p-cycle pc_f by combining two link-disjoint paths. One, denoted by $\mathbb{P}'$, starts from the source s to the destination d and the other one is $\mathbb{P}$ in the opposite direction. Hence, each link on the path $\mathbb{P}$ is protected against double-link failure.

However, constructing this new long p-cycle may destruct the validity of sharing between any link $e \in \mathbb{P}$ with any link $e' \notin \mathbb{P}$ if e and e' share a short p-cycle. We have to make sure that after the construction of pc_f, every link $e \in \mathbb{P}$ is still protected and every p-cycle sharing is valid. We define a function $check\_share2(e, e', pc_i)$ to perform the checking process. The function returns true if the sharing of a p-cycle pc_i by e and e' is valid based on Theorem 2. If the function returns false, we have to reconstruct the long p-cycle pc_f because it must contain link e' (We will explain why in the next.) We could make the sharing valid if the new p-cycle pc_f does not contain e'. Hence, we need to temporarily remove e' from G before construct the new pc_f. After all troublesome links are removed, we recompute a long p-cycle pc_f. We then repeat the process of checking p-cycle sharing validity and computing the long p-cycle until no invalid p-cycle sharing can be found.

We now explain why an invalid sharing of pc_i by e and e' is caused by the inclusion of e' in pc_f. Let pc'_f be the second p-cycle that protects e'. (The first p-cycle that protects e' is pc_i, which is shared by e.) In order for e and e' to validly share pc_i, we have to make sure that when both links fail, at least one of pc_i and pc'_f can protect e' and at least one of pc_i and pc_f can protect e. We know that at least one of the protection segments $pc_i(e')$ and $pc'_f(e')$ does not contain e since they must be link-disjoint. Therefore, there are three cases to consider as follows:

1. *Neither* $pc_i(e')$ *nor* $pc'_f(e')$ *contain* e: Clearly, e' can be protected by pc'_f since $pc'_f(e')$ is not affected by the failure. In addition, one of pc_i and pc_f can protect e because $pc_i(e)$

and $pc_f(e)$ are link-disjoint and therefore at least one of them does not contain e'. So, e and e' can validly share pc_i.

2. *$pc_i(e')$ contains e and $pc'_f(e')$ does not contain e*: e and e' can validly share pc_i for the same reason given in the previous case.

3. *$pc'_f(e')$ contains e and $pc_i(e')$ does not contain e*: e' has to be protected by pc_i when both e and e' fail. e can be protected by pc_f if $pc_f(e)$ does not contain e'. Therefore, the sharing is valid only if $pc_f(e)$ does not contain e'.

Algorithm 3: SFPP Scheme

Input: Working path $\mathbb{P}$, p-cycle set $\mathbb{C}$, $\mathbb{C}_{temp} = \phi$ and flag=1
Output: A updated set $\mathbb{C}$

```
1  for ∀e ∈ ℙ do
2      if ∃pc_i ∈ ℂ(e) and check_disjoint(pc_i, e)==true then
3          link e is protected once;
4      end
5      else
6          mark ℙ\e invalid and obtain pc_i by running cycle_build(e, 1);
7          ℂ = ℂ ⋃ {pc_i};
8          update ℂ(e') for all e' that can be protected by pc_i;
9      end
10     ℂ_temp = ℂ_temp ∪ pc_i(e);
11 end
12 mark e ∈ ℙ and e' ∈ ℂ_temp invalid in G;
13 construct pc_f by running cycle_build(ℙ, 1);
14 for ∀e ∈ ℙ and e' ∉ ℙ that share pc_i ∈ ℂ(e) do
15     if check_share2(e, e', pc_i) == false then
16         mark e' invalid and flag=0;
17     end
18 end
19 if flag == 0 then
20     repeat Line 13;
21     flag = 1 and goto Line 14;
22 end
23 Add pc_f to ℂ and ℂ(e') for all e' ∈ E that can be protected by pc_f;
```

As can be seen from the above three cases, if we know e and e' cannot validly share pc, then it must be the case that pc_f contains e'. And we can turn the sharing into a valid one by

making sure that pc_f does not contain e'.

The pseudo-code of the SFPP scheme is shown in Algorithm 3. The input is a working path $\mathbb{P}$ and the set of existing p-cycles $\mathbb{C}$, the output is the updated set $\mathbb{C}$ of p-cycles. Set $\mathbb{C}_{temp}$ stores the protection segments that are used to protect the links on $\mathbb{P}$. The first loop from line 1-11 tries to find an existing p-cycle to protect each link $e \in \mathbb{P}$. If no existing p-cycle in $\mathbb{C}$ can protect a given e, then construct a new p-cycle. We need to make sure that certain capacity sharing conditions have to be satisfied. The second part from line 12-13 is to construct the long p-cycle pc_f for the whole path $\mathbb{P}$. However, pc_f may cause invalid sharing between any link on $\mathbb{P}$ and the links not in $\mathbb{P}$ based on our previous analysis if it traverses any link in $\mathbb{C}_{temp}$. Hence, we need to use function $check\_share2$ to check the validity of pc_f. If the function returns false, we need to temporarily remove the link and reconstruct a new pc_f repeatedly until a valid long p-cycle is found. This process is described by line 14-22. After we find a pc_f that ensures all p-cycle sharing is valid, we add it into set $\mathbb{C}$ and also update $\mathbb{C}(e')$ for each edge $e' \neq e$ that can be protected by pc_f in line 23.

The time complexity of SFPP is dominated by the repeated construction process in lines 15-22. The complexity of function $check\_share2(e, e', pc)$ is $O(|V|^2)$, so the complexity of lines 15-18 is $O(|V||E||V|^2) = O(|E||V|^3)$. This block of code would be executed at most $|E|$ times because at most $|E|$ edges can be removed from G. Therefore, the complexity of SFPP is $O(|E||E|^*|V|^3) = O(|E|^2|V|^3)$.

Since SFPP makes use of long p-cycles, when failures occur in the network, some rerouted working paths may pass through redundant nodes and links since protection switching is done at the two endnodes of the failed link. This problem can be solved using the algorithm given in [59], which removes the loop backs and release the redundant capacity by reconfiguring the restored paths.

3.5 Numerical Results

3.5.1 Metrics and Methodology

We use ILOG CPLEX 10.1.0 to implement the ILP on a computer with four Intel Xeon 2.40GHz CPUs and 4G of memory. The ILP scheme provides the optimal solution for static traffic demands. The simulations of SPPP and SFPP are implemented on a computer with a Intel 3.0GHz CPU and 1.5G of memory. We measure the performance of our schemes from following metrics:

- *the protection redundancy:* defined as the ratio of protection capacity to working capacity.
- *the reject ratio:* defined as the ratio of the number of requests rejected to the number of requests received.
- *the number of wavelength channels:* one wavelength channel is defined as a wavelength on some link. For example, A 3-hop path uses 3 wavelength channels.
- *the number of XC:* this metric denotes the number of optical cross connects that need to be reconfigured upon a double-link failure.

When studying the performance of SPPP and SFPP, we first consider the incremental traffic, that is, a demand never terminates once it is satisfied. In this traffic model, the capacity of the network link is set to infinity. Then we study the performance of these two schemes in dynamic traffic model.

In both traffic models, a set of randomly generated connection requests are loaded to the network. For each connection request, the working path is routed along the shortest path between the source and the destination. For dynamic traffic, the arrival of traffic follows Poisson distribution with λ connection requests per second and the duration of the request is exponentially distributed with a mean of μ. The traffic load measured in Erlang is $\lambda\mu$.

3.5.2 Results for Static Traffic

We use ILOG CPLEX 10.1.0 to implement the ILP on a computer with four Intel Xeon 2.40GHz CPUs and 4BG of memory. A small test network with 6 nodes and 11 edges (shown in Fig. 3.6) is used. Table 3.2 shows the protection redundancy of different schemes and the running time of ILP for different number of connections. Each data point is the average of ten test cases.

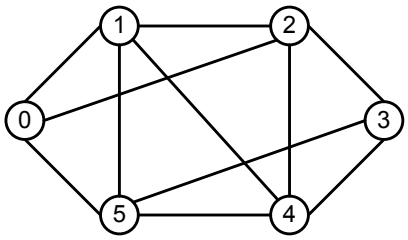


Figure 3.6 The 6-node 11-edge network.

Table 3.2 Redundancy and Computation time of ILP and Heuristic Algorithms

Number of connections:	1	2	3	4	5
ILP Run Times (s)	0.034	0.91	59.8	1304	11684.2
Protection Redundancy(ILP):	592%	448%	373%	306%	302%
Protection Redundancy(SPPP):	618%	546%	544%	534%	515%
Protection Redundancy(SFPP):	600%	561%	535%	498%	502%

The table shows that as the number of connections increases from 1 to 5, the protection redundancy decreases from 592% to 302%. This is expected because p-cycle sharing can be better exploited when more connections exist in the network. On the other hand, the running time increases exponentially as the number of connections increases. We also use the SFPP and SPPP under the same scenarios and calculate their redundancy performance. As expected, and as shown in Table 3.2, SFPP and SPPP are not as efficient as the ILP under static traffic because they deal with connection requests one by one and without considering the future incoming connection requests.

When ILP is used to find the optimal protection strategy for static demands, there will be sufficient time between the planning and provisioning processes even the ILP has long run-time. Moreover, ILP can provide the baseline to evaluate the performance of heuristic algorithms.

3.5.3 Comparison of SPPP and SFPP

We conduct simulations to compare the performance of SPPP and SFPP under incremental traffic and dynamic traffic. Two networks, the SMALLNET network and the COST239 network (Fig. 3.7), are used in the simulations. In each simulation run, a set of randomly generated connection requests are loaded to the network. For each connection request, the working path is routed along the shortest path between the source and the destination.

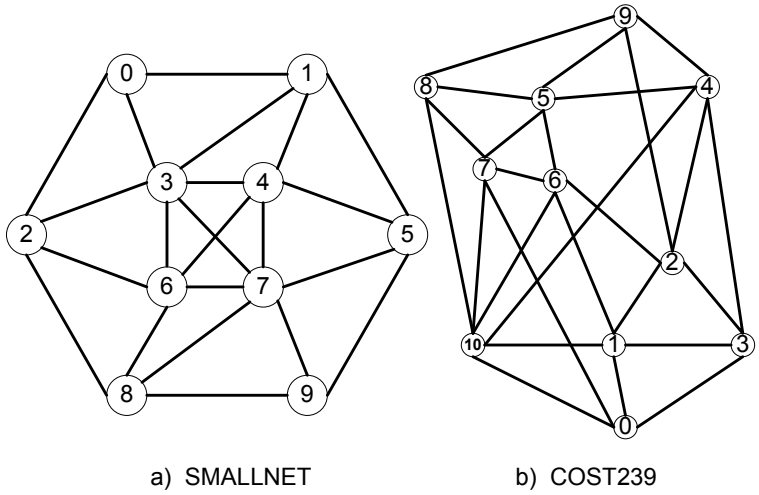


Figure 3.7 Two Networks in the Simulation.

In the first set of simulations, we consider incremental traffic. That is, a demand never terminates once it is satisfied. The capacity of the network link is set to infinity. The total number of wavelength channels used by all the working paths and by all the p-cycles are recorded for each simulation run. In Fig. 3.8, we show the performance of SPPP and SFPP under different traffic load in SMALLNET network. The results shows that SFPP uses less wavelength channels for protection than SPPP under all traffic loads. Specifically, SFPP achieves a 16.4%-18.3% reduction in wavelength usage over SPPP. The reason for SFPP to ourperform SPPP is that SFPP uses long p-cycles that have more straddling links so that

higher protection efficiency can be achieved.

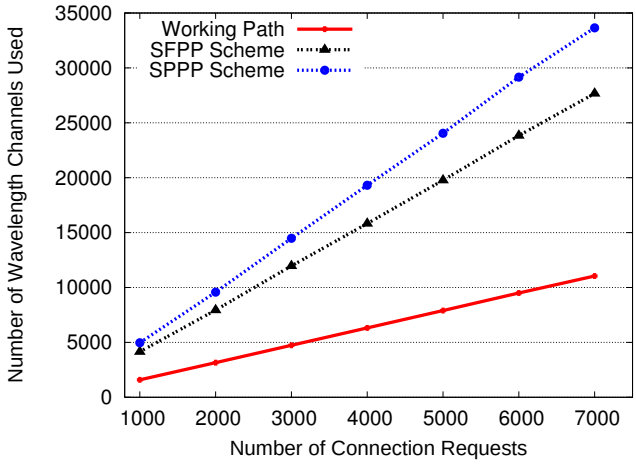


Figure 3.8 Wavelength usage of SPPP and SFPP in SMALLNET.

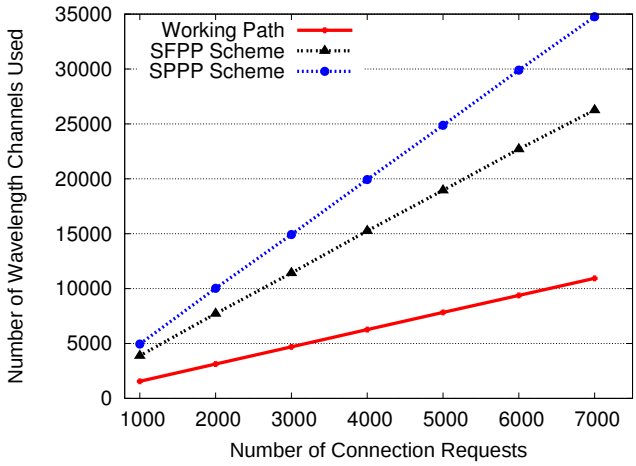


Figure 3.9 Wavelength usage of SPPP and SFPP in COST239.

In Fig. 3.9, we show the performance of SPPP and SFPP in COST239 network. Again, SFPP uses less wavelength channels for protection than SPPP under all traffic loads. Specifically, SFPP achieves a 21.5%-24.5% reduction in wavelength usage over SPPP. The improvement of SFPP over SPPP is bigger than that in SMALLNET network. This is because the COST239 network is denser. So, long p-cycles tend to have higher protection efficiency due to the inclusion of more straddling links.

Fig. 3.10 and Fig. 3.11 compare the protection redundancy of SPPP and SFPP for SMALL-

NET and COST239, respectively. Both figures show that the protection redundancy of SPPP and SFPP drop slightly as the number of connections increases, which is consistent with the ILP results. The redundancy of SFPP is much lower than that of SPPP. For SMALLNET, SFPP achieves 16.4%-18.3% reduction in redundancy over SPPP; For COST239, SFPP achieves 23.0% -24.5% reduction in redundancy over SPPP.

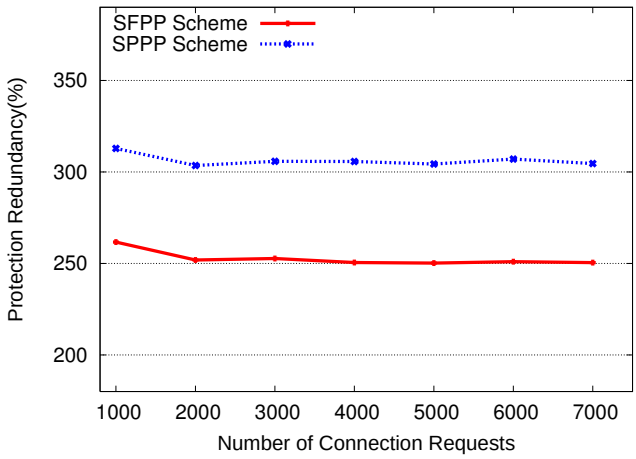


Figure 3.10 Protection redundancy of SPPP and SFPP in SMALLNET.

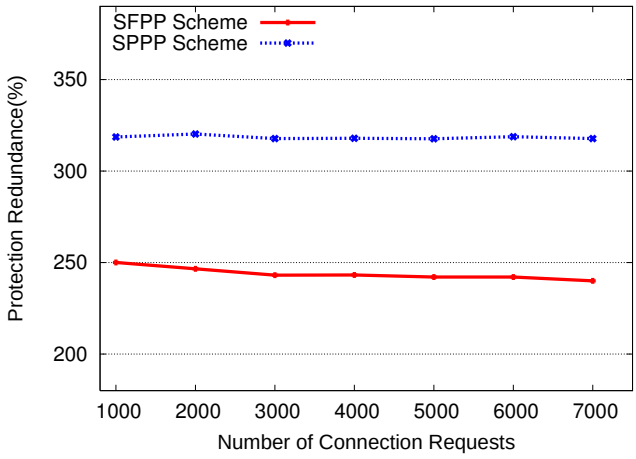


Figure 3.11 Protection redundancy of SPPP and SFPP in COST239.

As shown in Table 3.3, we also study the efficiency of p-Cycles constructed by SFPP and SPPP under incremental traffic in two different networks. The p-Cycle efficiency of p-Cycle pc_i is defined as the ratio of working wavelength channels protected by pc_i over the wavelength

channels reserved on p-Cycle pc_i. SFPP has better efficiency performance than SPPP because SFPP tends to use longer p-cycles that can protect more straddling links. Thus, each p-cycle has higher capacity efficiency in average and this also explains why SFPP has lower redundancy.

Table 3.3 Comparison of p-Cycle Efficiency

Networks	SMALLNET	COST239
SFPP Efficiency	0.75	0.78
SPPP Efficiency	0.65	0.63

In the second set of simulations, we consider dynamic traffic. In each simulation run, 5000 randomly generated connection requests are loaded to the network and the reject ratio is recorded. The arrival of traffic follows Poisson distribution with λ connection requests per second and the duration of the request is exponentially distributed with a mean of μ. The traffic load measured in erlangs is $\lambda\mu$. The capacity of the network link is set to 10 wavelengths.

In Fig 3.12, we compare the reject ratio of SFPP and SPPP under different traffic loads (in erlangs) in SMALLNET network. The results show that SFPP performs worse than SPPP when traffic load is low. This can be explained as follows. When the traffic load is low, there is not enough connections to fully utilize the protection capacity provided by the longer p-cycles. We also observe that the longer p-cycles can be fully utilized and they can provide more efficient protection than those p-cycles created by SPPP when the traffic load becomes high. According to our simulation, SFPP performs better than SPPP when traffic load is above 32 erlangs. However, the reject ratio is high and it is not practical.

In Fig 3.13, we compare the reject ratio of SFPP and SPPP under different traffic loads in COST239 network. Again, the results show that SFPP performs worse than SPPP under low traffic loads. Similarly, SFPP will perform better than SPPP when the traffic load is above 30 erlangs.

In Table 3.4, we compare the average computation time for a single demand using SPPP and SFPP in two test networks. As the table shows, it only takes SPPP 1.55 milliseconds on average to computer the p-Cycles for a demand in SMALLNET. Meanwhile, SPPP runs faster

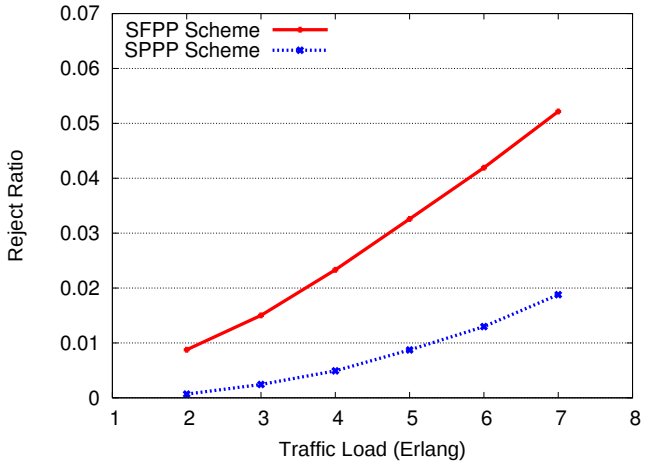


Figure 3.12 Reject ratio of SPPP and SFPP in SMALLNET.

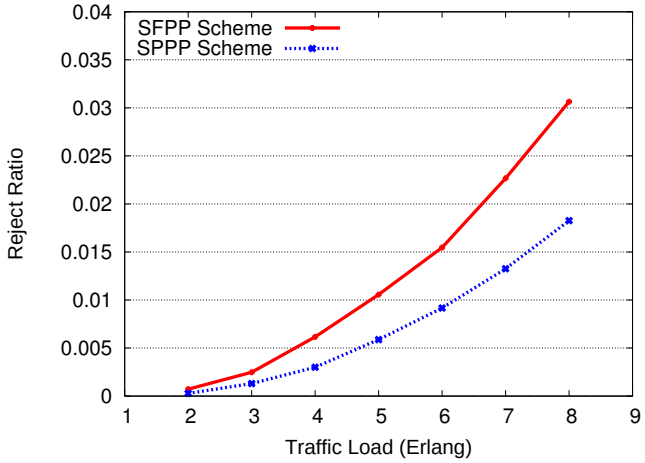


Figure 3.13 Reject ratio of SPPP and SFPP in COST239.

Table 3.4 Average Computation Time (milliseconds)

Networks	SMALLNET	COST239
SPPP	1.55	1.9
SFPP	2.1	2.45

than SFPP in both networks which corresponds well with previous time complexity analysis. SFPP just needs 2.45 milliseconds to compute the p-Cycle in COST239. Thus, the proposed schemes are suitable for dynamic demands.

3.5.4 Comparison of SPPP and the Algorithms in [53]

Table 3.5 Comparison of Algorithms in ARPANET

Algorithm	I	II	MADPA	SPPP
Protection ratio	100%	100%	98.8%	100%
Protection redundancy	200%	200%	200%	259%
XC_{max} with signaling	26	18	N/A	6
XC_{avg} with signaling	9.34	8.64	N/A	4.4
XC_{max} w/o signaling	N/A	N/A	24	4
XC_{avg} w/o signaling	N/A	N/A	7.3	4

We compare SPPP with the three approaches–Method I, Method II, and MADPA–proposed in [53] as shown in Table 3.5. The network topology used is the 20-node 32-link ARPANET network. Protection ratio is the percentage of double-link failures that can be protected. Protection redundancy is the ratio of the total protection capacity to the total working capacity. XC_{max} and XC_{avg} denote the worst-case and average number of optical cross connects that need to be configured upon a double-link failure. In [53], the authors assume the working capacity reserved on each link is one unit such that the schemes proposed in the paper require fixed protection capacity on each link, which is 200%.

When a link fails, Methods I and II require that all nodes in the network are informed of the failure through signaling. However, this is not required for MADPA. SPPP can operate with or without signaling of the failure event. If, upon a link failure, the traffic on the link is sent on both p-cycles simultaneously, then signaling is not required. In this case, a total of 4 cross connections are needed to recover from any double-link failure because the two endnodes of each failed link need configure their cross connects to direct the traffic onto the p-cycles. On the other hand, if only one p-cycle is used to restore the traffic upon a link failure, then

signaling of failure is required and a total of 6 cross connections are needed to recover from a double-link failure in the worst case.

The worst case occurs when the second failure affects the p-cycle used to protect the first failure. In this case, when the first link fails, both end nodes configure their cross connects to direct the traffic onto the first p-cycle for this link. When the second link fails, the end nodes of the link configure their cross connects to direct the traffic onto one of the two p-cycles that is not affected by the first link failure. After the end nodes of the first failed link learn that the second failure affects the p-cycle being used, they reconfigure their cross connects to direct the traffic onto the second p-cycle for this link. Thus, a total of 6 cross connections are needed.

The results in Table 3.5 show that while SPPP has higher protection redundancy than the other three methods, the number of cross connections required is much less. Since p-cycles are pre-configured, SPPP requires only the end nodes of the failed links to configure their cross connects. On the other hand, cross connects have to be configured by every node along the protection path in the other three methods. Thus, SPPP is much faster in restoration than the other methods. Basically, SPPP trades off protection redundancy for restoration speed. Compared with the other methods, SPPP's gain in restoration speed is much larger than its loss in protection redundancy.

3.6 Conclusion

In this paper, we consider the problem of protecting connections against two-link failures. The basic idea is to protect each working link with two p-cycles with link-disjoint protection segments. We present an ILP model to compute the optimal set of p-cycles for protecting a set of static demands. The ILP can provide the optimal solution for static demands and provide a baseline for evaluating the performance of heuristic algorithms. Realizing that ILP is not suitable for online provisioning process, we also propose two protection schemes, SPPP and SFPP, for dynamic demands. The numerical results show that SFPP is more capacity efficient than SPPP under incremental traffic. Under dynamic traffic, SPPP has lower blocking than SFPP in most practical cases. In addition to the time complexity analysis, we run

simulations to show that the average computation time for a single demand is at the milisecond-level. The time complexity analysis also shows the good scalability of our proposed SFPP and SPPP schemes. Compared with the algorithms proposed in [53], SPPP trades off protection redundancy for fast restoration speed. In the future, we plan to develop heuristic algorithms to solve the static traffic provisioning problem and design more efficient algorithms for dynamic traffic in terms of running time and protection redundancy. We will also consider multiple-link failure protection.

CHAPTER 4. P^2-CYCLES: P-CYCLES WITH PARASITIC PROTECTION LINKS

A paper ready for submission [1]

Long Long and Ahmed E. Kamal

Abstract

The p-cycle and its Failure Independent Path Protection (FIPP) extension are known to be efficient and agile protection strategies. The p-cycle is pre-configured such that if there is a failure, only the switches at two end nodes need to be reconfigured. In this paper, we extend the p-cycle by allowing cycles to have attached links, called Parasitic Protection Links (PPL), in order to protect paths whose source and destination nodes are not only located on the cycle but also connected by a PPL to the cycle. A p-cycle with PPL is named p^2-cycle.

We address the unicast service protection problem against single-link failures by using p^2-cycle in mesh networks for both static and dynamic traffic scenarios. In the static case, the problem is formulated as an Integer Linear Program (ILP). We further propose two p^2-cycle based heuristic algorithms, Strict Routing Protection (SRP) and Flexible Routing Protection (FRP), to address the dynamic traffic case. The numerical results show that the p^2-cycle scheme provides better capacity efficiency than the FIPP p-cycle scheme in all the traffic scenarios considered and achieves only less than 3% extra total cost over the optimum, provided by Shared Backup Path Protection (SBPP) approach when the traffic load is high. We also study the failure recovery performance in terms of average number of switch reconfigurations

[1] Part of the work has appeared in the proceedings of the *IEEE International Conference on Communications* (ICC) 2010 [38].

(NOR), and show that the performance of the p^2-cycle becomes much better than that of SBPP and gets close to FIPP as the traffic demand increases. In the dynamic case, both SRP and FRP outperform FIPP p-cycle schemes in terms of blocking probability in most scenarios considered. In general, the p^2-cycle protection scheme outperforms the p-cycle based in terms of capacity efficiencies which being slightly slower in terms of traffic recovery speed.

4.1 Introduction

Network survivability, defined as the ability of networks to continue to function properly in the presence of the failures of network components [3], is an important requirement for WDM optical networks due to their ultra-high capacity. A single failure can disrupt millions of applications and users. Ring-based networks and resilience schemes are prevalent due to the simple manageability and fast recovery mechanism, in which the traffic recovery process can be completed within 50-60 ms, but require 100% capacity redundancy [60]. As mesh-based networks emerged, more capacity efficient protection schemes were proposed which allow backup capacity sharing. These schemes fall into three categories: link-based, segment-based and path-based [29].

Link-based protection schemes produce the fast traffic recovery speed but suffer from the worst resource efficiency [31]. Path-based protection schemes usually achieve the best resource efficiency. Among them, a path protection scheme, namely, Shared Backup Path Protection (SBPP), was shown to be the most capacity efficient protection scheme [29]. However, it suffers from long traffic recovery time upon a network failure. Segment-based protection schemes lie between the link-based and path-based schemes, and offer a better combination of bandwidth efficiency and recovery time [61, 62].

The pre-configured protection cycle approach, referred to as p-cycle, combines the merits of both ring-based and mesh-based protection schemes and achieves the recovery speed of ring-based with the capacity efficiency of mesh protection [39]. P-cycle has been proven theoretically to be the most efficient pre-configured protection scheme in terms of capacity efficiency and recovery speed [63]. A thorough survey of p-cycle-based survivability techniques was conducted

by Grover in [64]. Since the concept of p-cycle was first introduced in [39], a large amount of work in the literature studied the p-cycle design problem with unicast traffic against a single-link failure. The authors in [39, 41] introduced a tractable solution by solving the problem in two steps: by first routing the connections, and then selecting the best p-cycles candidates from the enumeration of all the cycles to protect the established connections. In this approach, the optimality of the solution is compromised by solving two subproblems. In [42, 43], the problem were also solved jointly by minimizing the total capacity cost used by both primary paths and protection p-cycles.

Besides link protection, p-cycles has been extended to protect segments and paths in [44, 45]. Reference [45] proposed a Failure Independent Path-Protecting (FIPP) p-cycle which is a more capacity efficient protection strategy than link protecting p-cycle, with a little relaxation on the recovery time. Recently, the author of [66] introduced a new 1+N protection scheme against single-link failures by combining network coding and p-cycles. Besides p-cycles, other pre-configured structures are also used for fast recovery, such as p-trails [67] and p-trees [68, 69]. In [69], the authors extended p-tree by adding links to form a more flexible protection pattern, such as cycles or trees, but it can provide higher protection capacity than link-protecting p-cycles only in a network with non-uniformly distributed spare capacity.

Regardless of the protection schemes, the trade-off between the capacity efficiency and failure recovery speed always exists [70]. Since the p-cycle has a good combination of capacity and time efficiency, we attempt to further increase the capacity efficiency of FIPP p-cycles without sacrificing too much of its fast recovery property. In this paper, therefore, we extend the FIPP p-cycle paradigm to a new one in which each p-cycle may be augmented with a number of protection links that are attached to the cycle, called "Parasitic Protection Links (PPL)".

An example is shown in Figure 4.1 to illustrate the concept of the p^2-cycle. In Fig. 4.1(a), a p-cycle ($A-B-C-D-E-F-A$) is used to protect two bidirectional paths, P_1 and P_2, where path P_1 traverses on-cycle span (D, E) and (E, F) and is protected by on-cycle segment ($F-A-B-C-D$) and path $P_2(A-C)$ is a straddling path that is protected by on-cycle

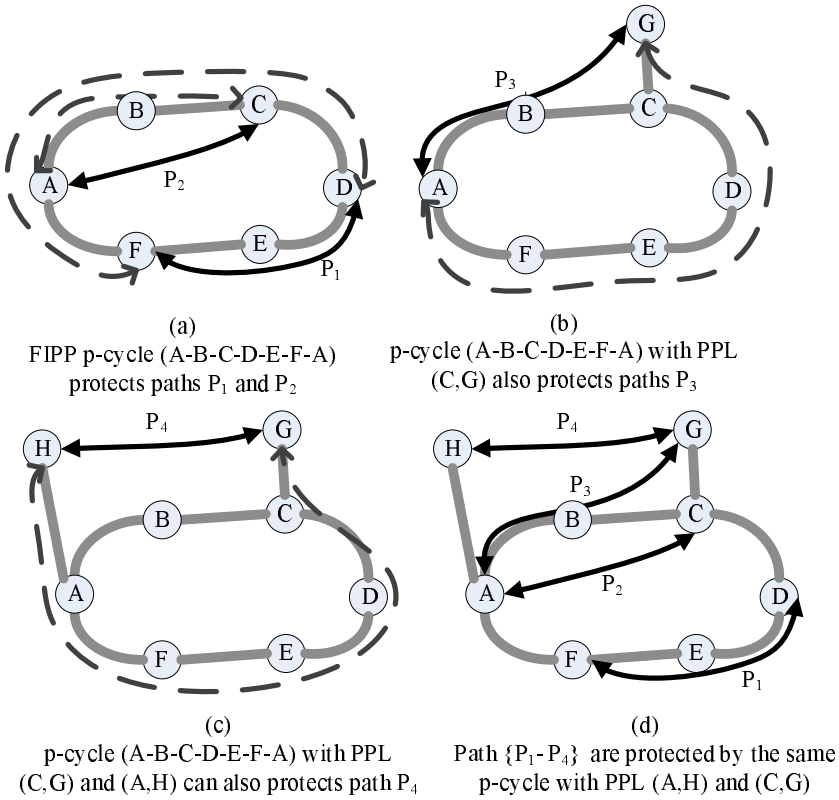


Figure 4.1 An example of a p-cycle with PPLs

segment $(A{-}B{-}C)$. Working paths are denoted by solid lines and protection paths are represented by dashed lines. Assuming we have another working path P_3 (shown in Fig.4.1(b)) traversing on-cycle span (A, B) and non-cycle span (B, G), the original p-cycle cannot protect it, since the end node G is not on the cycle. We then extend the p-cycle to have a PPL (C, G) and hence protect P_3 by using the path $(A{-}F{-}E{-}D{-}C{-}G)$, which is partly on-cycle and partly on PPL. The idea can also be applied to a path whose two end nodes are not on the cycle, such as path p_4 shown in Fig.4.1(c). Two PPLs (A, H) and (C, G) can be used to construct the protection path $(H{-}A{-}F{-}E{-}D{-}C{-}G)$. Therefore, the augmented p-cycle with the two links (A, H) and (C, G) can protect four paths (shown in Fig.4.1(d)). Hence, augmenting a p-cycle to have PPLs enhances the flexibility of protection and thus may decrease spare capacity redundancy and reduce overall capacity cost.

The rest of the paper is organized as follows: In Section 4.2, we analyze the p^2-cycle protection scheme in more detail. In Section 4.3, we consider unicast protection problem with

static traffic demands using p^2-cycles as the protection method. The problem is formulated as an Integer Linear Program (ILP). In Section 4.4, we further consider dynamic traffic scenarios, in which two heuristic algorithms are proposed. Performance evaluation of multiple criteria for both static and dynamic traffic scenarios will be presented in Section 4.5. Finally, we conclude the paper in Section 4.6.

4.2 Overview of p^2-Cycles

In this section, we provide an overview of p^2-cycle protection scheme and elaborate the details of protection mechanism and traffic recovery time.

4.2.1 Protection Mechanism

The protection ability of a p^2-cycle is an enhancement to that of the p-cycle by adding attached spans to the cycle, which enables the cycle to provide protection to the connections whose end nodes are one hop away from the cycle. All the nodes on the cycle still remain pre-configured. For the nodes that also connect to PPLs, they only reconfigure the switches when the attached PPLs are activated to provide protection upon a network failure. Given a unicast session, the primary path and the corresponding protection path, which may consist of an on-cycle segment and one or two PPLs, will be determined in advance regardless of the location of the failure. Hence, the p^2-cycle protection scheme is also failure independent.

Upon a link failure, the failure will be detected by the end nodes of the failed span and the corresponding signals will transmitted to the source and destination nodes of the path. The distinction between a p^2-cycle and an FIPP p-cycle here is that the source or the destination may not be on the cycle. Therefore, in order to reroute the traffic onto the backup path, the source and destination nodes need to reconfigure their switches, as do the end nodes on the protection cycle that connect to an activated PPL.

Let us review the examples in Figure 4.1. In Figure 4.1(b), if a failure happens to span (A, B) or (B, G), both end nodes A, G and on-cycle node C will reconfigure their switches to reroute the traffic through the backup path. Similar reconfigurations should be done by node

H, G, A and C in Figure 4.1(c) upon any span failure on the primary path P_4.

4.2.2 Traffic Recovery Time

In general, the traffic recovery process consists of three phases: failure detection, fault signaling and switch reconfigurations. However, switch reconfigurations are usually the most time consuming phase during the process, since each reconfiguration takes 10 - 20s ms [101] depending on the technology used. More node reconfigurations on the protection path will result in longer traffic recovery. Thus, the average *Number of Node Reconfigurations* (NOR) is a key factor to inspect the traffic recovery speed of a given protection scheme.

It is apparent that NOR of FIPP p-cycle scheme is always equal to two, one at the source and the other at the destination, and rest of the nodes are pre-configured on the cycle. However, for a path protected by a p^2-cycle, NOR can be two, three or four depending on how many PPLs are used by a protection path. In Fig. 4.1, path 1 and 2 are protected by the cycle without usage of any PPL and thus the NOR of them is two. However, the NOR of protection path 3 and 4 equals three (A,C and G) and four (A,H,C and G), respectively.

It is worth noting that the distance between any end node of a connection and its protection cycle does not have to be fixed by one hop, such as PPL (A, H) and (C, G) in Fig. 4.1. Instead, we can extend the capability of a p-cycle to protect the connections whose end nodes are not only one hop but k hops away where $k > 1$. The overall protection capacity efficiency may be enhanced. However, each PPL will easily grow to be a segment that may have multiple links in order to protect more connections. In that case, the length of each protection path may increase. But more critically, the NOR required on a protection path may increase significantly since every node on a stretched "PPL" may potentially become a reconfiguration node. In addition to the original reconfiguration nodes, such as end nodes or nodes on the cycle that are connected to PPLs, the traffic recovery time may soon become unacceptable due to a large number of reconfigurations. Therefore, in order to achieve great improvement in capacity efficiency without sacrificing too much recovery time, we limit the number of hops of any "PPL" to one link for every p^2-cycle.

4.3 Static Traffic Scenarios

In this section, we address the static unicast services protection problem using p^2-cycle protection scheme in WDM networks against single-link failure scenarios. The problem will be defined first and then formulated as an Integer Linear Program (ILP).

4.3.1 Problem Statement

Given a number of static traffic requests, the problem can be solved in two ways. The first method is to divide the problem into two sub-problems: working paths provisioning and protection cycles provisioning. Then the two sub-problems are solved sequentially. However, the optimality of the solution may be compromised. In order to study the overall optimal performance of the p^2-cycle scheme, we address the joint capacity placement (JCP) problem in which working paths and protection cycles are provisioned jointly such that the minimum total cost is achieved.

A number of assumptions are given as follow, in which a session refers to a provisioned traffic demand or request:

1. Each unicast session is bidirectional with a unitary traffic rate (one wavelength) and the traffic in both directions has to be routed through the same paths and protected by the same p^2-cycle.

2. Each p^2-cycle is also bidirectional and has unitary capacity on both on-cycle spans and PPLs.

3. Each span has enough wavelengths and each node is equipped with wavelength converters over all wavelengths, such that wavelength continuity is not required in the network.

We now state the JCP problem formally: Given a bidirectional unicast traffic matrix D where $D = d_l(s_l, t_l)$, $(0 \leq l < M)$ where M is the number of connections, and a weighted undirected graph $G = (V, E)$ in which each span $e \in E$ has a cost c_e, provision and protect all the unicast sessions with minimal total cost (the cost consumed on each span is the product of the total number of wavelengths used over the span and the unit cost over the span).

Input Parameters:

P	The maximum number of p^2-cycles in the solution
p	an index that refers to the pth p^2-cycle where $1 \leq p \leq P$
l	an index that refers to the lth session where $1 \leq l \leq M$

Variables Notation:

f_{mn}^{l} :	A binary variable, equals 1 if the primary path of session l traverses span $(m,n) \in E$
q_{mn}^{l} :	A binary variable, equals 1 if the protection flow of session l traverses span $(m,n) \in E$
Zf_{n}^{l} :	A binary variable, equals 1 if the primary path of session l passes node n
Zq_{n}^{l} :	A binary variable, equals 1 if the protection path of session l passes node n
e_{mn}^{p} :	A binary variable, equals 1 if p^2-cycle p traverses span (m,n)
z_{n}^{p} :	a binary variable, equals 1 if p^2-cycle p traverses node n
b_{mn}^{p} :	A binary variable, equals 1 if span (m,n) is a PPL of p
$B_{mn}^{p,l}$:	A binary variable, equals 1 if PPL (m,n) is used by p to protect session l
x_{l}^{p} :	A binary variable, equals 1 if p protects session l
$X_{l_1 l_2}^{p}$:	A binary variable, equals 1 if session l_1 and l_2 are both protected by p
$\phi^{l_1 l_2}$:	A binary variable, equals 1 if session l_1 and l_2 share protection of any p^2-cycle
$\gamma_{mn}^{l_1 l_2}$:	A binary variable, equals 1 if the primary paths of both session l_1 and l_2 use span (m,n)
$\Gamma^{l_1 l_2}$:	A binary variable, equals 1 if the primary paths of session l_1 and l_2 use at least one common span
μ_{u}^{p} :	A binary variable, equals 1 if node u is the master node of p
$\alpha_{mn,v}^{p}$:	A binary variable, equals 1 if span (m,n) is used to reach node v from the master node of p
$\beta_{n,v}^{p}$:	A binary variable, equals 1 if node n is traversed by the flow from the master node to node v through the cycle of p
ϵ :	A small positive constant (0.0001)

4.3.2 ILP Formulation

We formulate the JCP problem as an ILP. Since the number of cycles increases exponentially with network sizes, we do not enumerate all the cycles in a given network in the formulation. Instead, the flow variables will form the cycles in the solution. The input parameters and decision variables used in the ILP are defined in the table.

The objective function is:

$$\text{Minimize:} \quad \sum_{(m,n)\in E} \left(\sum_{0\le l<M} f_{mn}^{l} + \sum_{0\le p<P} (e_{mn}^{p} + b_{mn}^{p}) \right)$$

Each span between node m and n is denoted by (m, n) where $m<n$ in the network $G=(V, E)$. The objective function minimizes the total cost consumed by the primary paths (first term) and the p^2-cycles used to protect them. Each p^2-cycle is composed of on-cycle spans e (second term) and parasitic protection spans b (third term).

The constraints are such that:

1. **Flow Conservation Constraints:**

For $n \in V\backslash\{s_l, t_l\},\ \forall l$:

$$\sum_{n:(s_l,n)\in E} f_{s_l,n}^{l} = \sum_{m:(m,t_l)\in E} f_{m,t_l}^{l} = 1; \tag{4.1}$$

$$\sum_{n:(u,n),(n,u)\in E} f_{u,n}^{l} = 2Z f_{n}^{l}, \tag{4.2}$$

$$\sum_{n:(s_l,n)\in E} q_{s_l,n}^{l} = \sum_{m:(m,t_l)\in E} q_{m,t_l}^{l} = 1; \tag{4.3}$$

$$\sum_{n:(u,n),(n,u)\in E} q_{u,n}^{l} = 2Z q_{n}^{l}; \tag{4.4}$$

Equations (4.1)-(4.4) ensure that each session l has a primary and a protection path. The source and destination nodes of the session connect to only one span used by each path, but each intermediate node is connected by two adjacent spans.

2. **Protection Constraints:**

$$\sum_{p} x_l^p = 1, \quad \forall l; \tag{4.5}$$

$$e_{mn}^p \geq x_l^p \wedge q_{mn}^l, \quad \forall p, l, \ \forall (m,n) \in E, \ m,n \neq \{s_l, t_l\}; \tag{4.6}$$

$$B_{mn}^{p,l} = x_l^p \wedge q_{mn}^l \wedge (1 - e_{mn}^p), \quad \forall p, l, \ \forall (m,n) \in E; \tag{4.7}$$

$$b_{mn}^p \geq \epsilon \left(\sum_{l} B_{mn}^{p,l} \right), \quad \forall p, \forall (m,n) \in E; \tag{4.8}$$

Equation (4.5) ensures that each session is protected exactly once by a p^2-cycle. If session l is protected by each p and the protection flow uses span (m,n), then span (m,n) should be an on-cycle span of p except that m or n is the source or the destination of session l. In that case, (m,n) can be a PPL. This constraint is ensured by equation (4.6), in which the symbol $\wedge$ denotes a conjunction operation. A conjunction expression $X = \bigwedge_{1 \leq i \leq N} x_i$ can easily be represented by two linear equations $X \leq \frac{1}{N}(\sum_i x_i)$ and $X \geq \sum_i x_i - N + 1$, given binary variables X and x_i.

Equation (4.7) ensures that if a span (m,n) is used by a protection flow to protect session l, it should be a part of p^2-cycle p. However, if it is not an on-cycle span, it must be a PPL, which is denoted by $B_{mn}^{p,l}$. A PPL can be used to protect multiple connections. As long as there exists at least one connection using span (m,n) as a PPL of p, the span (m,n) is counted as a PPL of p. Equation (4.8) ensures this constraint.

3. **Link Disjointness Constraints:**

$$f_{mn}^l + q_{mn}^l \leq 1, \quad \forall l, \ \forall (m,n) \in E; \tag{4.9}$$

The working and backup paths of any session l should be link-disjoint to survive any single-link failure. This is ensured by equation (4.9).

4. **Protection Capacity Sharing:**

For $\forall p,\ \forall l_1, l_2, l_1 < l_2 < M,\ \forall (m,n) \in E$:

$$X^p_{l_1 l_2} = x^p_{l_1} \wedge x^p_{l_2}; \tag{4.10}$$

$$\phi^{l_1,l_2} \geq \epsilon \left(\sum_p X^p_{l_1 l_2} \right); \tag{4.11}$$

$$\gamma^{l_1,l_2}_{mn} = p^{l_1}_{mn} \wedge p^{l_2}_{mn}; \tag{4.12}$$

$$\Gamma^{l_1,l_2} \geq \epsilon \left(\sum_{(m,n)\in E} \gamma^{l_1,l_2}_{mn} \right); \tag{4.13}$$

$$q^{l_1}_{mn} + q^{l_2}_{mn} \leq 2 - \left(\phi^{l_1,l_2} \wedge \gamma^{l_1,l_2} \right); \tag{4.14}$$

Equations (4.10) ensures that if two different session l_1 and l_2 are protected by the same p, then $X^p_{l_1,l_2} = 1$. If they share any p^2-cycle p, then $\phi^{l_1 l_2} = 1$ as shown in equation (4.11). Equation (4.12) and (4.13) make sure that $\Gamma^{l_1 l_2} = 1$ if the primary paths of session l_1 and l_2 are not link disjoint. In this case, if l_1 and l_2 also share the protection of the same p, the protection flow of l_1 and l_2 cannot traverse the same span, which is ensured by equation (4.14).

5. **Cycle Constraints:**

$$\sum_{n:(m,n)\in E} e^p_{mn} = 2z^p_n,\ \forall n \in V,\ \forall p; \tag{4.15}$$

$$|z^p_m - z^p_n| \geq b^p_{mn},\quad \forall (m,n) \in E,\ \forall p; \tag{4.16}$$

The cycle constraints make sure that each node on the cycle is passed twice by on-cycle spans, as described in equation (4.15). If span (m,n) is a PPL of p, then one of m and n must lie on the cycle while the other not, which is given by Equation (4.16). A new variable can be introduced to replace the absolute expression with two linear equations. For instance, $Z^p_{m,n}$ can be used to replace $|z^p_m - z^p_n|$ with two new constraints: $Z^p_{m,n} \geq z^p_m - z^p_n$ and $Z^p_{m,n} \geq z^p_n - z^p_m$. The minimization of the cost of connection provisioning in the objective function will ensure that Z^p_{mn} is equal to $|z^p_m - z^p_n|$.

6. **Cycle Uniqueness:**

$$\sum_{u \in V} \mu_u^p = 1, \quad \forall p; \tag{4.17}$$

$$\sum_{n:(m,n)\in E} \alpha_{mn,v}^p = 2\beta_{m,v}^p - \mu_m^p, \quad \forall p, \ \forall \{m,n,v\} \in V, \ m \neq v; \tag{4.18}$$

$$\sum_{m:(m,v)\in E} \alpha_{mv,v}^p = 1 - \mu_v^p, \ \forall p, \forall v \in V; \tag{4.19}$$

$$e_{mn}^p \geq \alpha_{mn,v}^p - \epsilon\beta_{m,v}^p, \quad \forall p, \ \forall (m,n) \in E, \forall v \in V; \tag{4.20}$$

However, equation (4.15) is not enough to guarantee that there is only one cycle with index p, since multiple cycles can be formed with the same index p while still complying with constraint (4.15). Some work has been done to address this issue. The method proposed in [42] is simple and the number of introduced variables is linear in the size of the network. However, it can only apply to unidirectional cycles. Hence, we use the approach proposed in [65]. This approach picks a node on each cycle randomly and defines it as the master node such that there must exist a flow from the master node to every other on-cycle node through the cycle.

Equation (4.17) ensures that there is only one unique master node for each p^2-cycle where node u is the master node of p. Equations (4.18) and (4.19) ensure the flow conservation between the master node and all other on-cycle nodes. Equation (4.18) guarantees that the flow uses one span connected to the master node but two connected to any intermediate node passed by the flow and Equation (4.19) ensures that only one span connecting to the destination node is used by the flow on the cycle. Equation (4.19) also guarantees that if a node v is on the cycle p, then all the spans traversed by the flow from the master nodes to node v should be on-cycle spans of p. Therefore, the uniqueness of one cycle for each index p is ensured. The number of variables introduced to guarantee a single cycle for each p is $P|E||V| + P(|V|+1)|V|$.

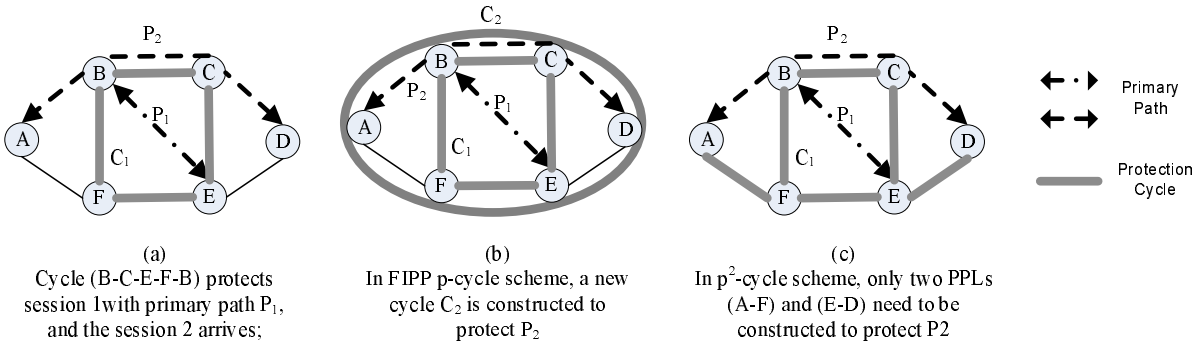
The total number of variables used in the ILP formulation is dominated by $M^2(|E|+P) + P|V|(|E|+|V|)$ and the total number of constraints can be denoted by $O(M^2(|E|+P) + P(|E||V| + |V|^2 + M|E|))$.

In addition, this ILP formulation can be easily extended to address other network scenarios or design goals. For instance, instead of minimizing total capacity usage, it can be used to minimize only the spare capacity used by protection p^2-cycles given the primary path for each session. Accordingly, all the primary path variables f_{mn}^{l} become input parameters and will be removed from the objective function. Besides, if the design goal is to balance the traffic load by minimizing the maximum capacity consumed on any span in a network, we just need to introduce a new variable, say ζ, to substitute the original objective function such that $\zeta \geq \sum_{0 \leq l < M} f_{mn}^{l} + \sum_{0 \leq p < P} (e_{mn}^{p} + b_{mn}^{p})$. In this case, ζ denotes the maximum capacity required on any link.

In terms of the change of network scenarios, for example, if each span has an upper bound on the number of wavelengths, denoted by λ_{mn}, we can add a constraint: $\lambda_{mn} \geq \sum_{0 \leq l < M} f_{mn}^{l} + \sum_{0 \leq p < P} (e_{mn}^{p} + b_{mn}^{p}),\ \forall (m,n) \in E$ to ensure that the total capacity reserved does not violate the upper bound. Therefore, our ILP model can be flexibly modified to adapt it to various network scenarios and different design goals.

4.4 Dynamic Traffic Scenarios

In this section, the dynamic traffic scenario will be addressed. We first demonstrate the motivation to apply p^2-cycles in the dynamic case and then formally state the problem. Finally, we propose two heuristic algorithms to study the problem.



(a) Cycle (B-C-E-F-B) protects session 1with primary path P_1, and the session 2 arrives;

(b) In FIPP p-cycle scheme, a new cycle C_2 is constructed to protect P_2

(c) In p^2-cycle scheme, only two PPLs (A-F) and (E-D) need to be constructed to protect P2

Figure 4.2 Demonstration of p^2-cycles in dealing with dynamic traffic

4.4.1 Motivation

p-Cycle is very capacity efficient and suitable with static traffic scenarios since the traffic requests are known in advance and do not vary with time. However, this may not be the case when dealing with dynamic traffic without the prior knowledge of arrival time of future requests. Due to the pre-configuration property of traditional p-cycles, it is extremely difficult to re-provision all the protection cycles whenever a new session arrives in order to minimize overall cost. Each provisioning takes large computation cost and complex network reconfiguration. Therefore, most of the work in the literature assume that established p-cycles should not vary with time or traffic. The comparison of a number of resilience approaches in protecting dynamic traffic was conducted in [71]. The authors in [72] proposed three different routing algorithms along with link-based p-cycle protection scheme to deal with dynamic traffic. The results indicate that the proposed p-cycle based design performs better than SBPP in dense networks but worse in sparse networks. Protected Working Capacity Envelopes (PWCE) is another method to address dynamic traffic scenarios [73]. It divides the entire network into two partitions: working and protection. Both static and dynamic traffic can be accommodated as long as the total traffic do not exceed the limit of working envelopes.

Although some decent results have been shown in the literature, p-cycles still have such intrinsic weakness in dealing with dynamic traffic. If an incoming session whose end nodes do not lie on any cycle, it cannot be protected and a new cycle has to be constructed to protect this session, or the existing cycles must be reconfigured. An example shown in Figure 4.2 illustrates such weakness and also reveals the advantage of p^2-cycles. In Fig. 4.2(a), session 1 has been provisioned and protected by cycle $C_1(E{-}C{-}B{-}F{-}E)$. As session 2 arrives, the primary path of session 2 is provisioned as $P_2(A{-}B{-}C{-}D)$. Under FIPP p-cycle scheme, cycle C_1 cannot protect it and thus a new cycle $C_2(A{-}B{-}C{-}D{-}E{-}F{-}A)$ is constructed to protect it as shown in Fig.4.2(b). However, instead of building a new cycle, using p^2-cycle approach we can add two PPLs (A,F) and (D,E) to connect the end nodes of P_2 such that C_1 can also provide a protection segment $(A{-}F{-}E{-}D)$ for P_2 as shown in Fig.4.2(c). Therefore, both sessions are protected by a p^2-cycle with much less cost (a total of 6 links compared to

10 links).

4.4.2 Problem Statement

In dynamic traffic scenarios, a WDM mesh network is given with network resources, such as the maximum number of wavelengths and the cost on each span. Each traffic request arrives to the network in a dynamic fashion such that it needs to be considered individually based on the current network status. The network status consists of the detailed working and available wavelengths on each span as well as all the accepted sessions and p^2-cycles provisioned in the network. Hence, the dynamic traffic protection problem can be defined as follows:

Given a network modeled as an undirected graph $G = (V, E)$ where each undirected span $e \in E$ has a cost c_e, the current network which includes the currently used and available wavelengths on each span e, each accepted session l and their protection p^2-cycles. Provision incoming unicast sessions against any single-link failure with the minimum overall blocking probability by using p^2-cycle scheme. The assumptions required in this dynamic traffic case are the same as that in the static case defined in Section 4.3.

Table 4.1 Notations used in the algorithms

Notations	Meaning
$\mathbb{D}$	The set of sessions that are active in the network
$\mathbb{C}$	The set of existing p^2-cycles in the network
$\mathbb{P}$	The set of primary paths of the active sessions in the network
$\mathbb{T}$	A temporary set of the combination of $< c_p, q_l, f_l >$
$d_l(s_l, t_l)$	An incoming traffic request l with end nodes s_l and t_l, stored in $\mathbb{D}$ if it is accepted
$c_p \in \mathbb{C}$	The pth p^2-cycle in $\mathbb{C}$
$f_l \in \mathbb{P}$	The primary path of traffic request l
q_l	The protection path of traffic request l
$\delta(d_l, c_p)$	Integer, the distance between the end nodes of a connection d_l to the cycle of c_p.

4.4.3 Heuristic Algorithms

We design two heuristics to address the dynamic traffic case. In the first method, named Strict Routing Protection (SRP), the primary and protection path for each incoming session are computed separately. The primary path is firstly provisioned using Dijkstra's shortest routing algorithm. Based on the primary path, either an existing p^2-cycle or a new cycle is found to protect it. In the second method, named Flexible Routing Protection (FRP), the primary and protection paths of an incoming session are constructed jointly. The existing p^2-cycles will be preferred to being used first. If no existing one is able to protect the session, a new cycle will be formed. We allow spare capacity sharing between different sessions to increase the capacity efficiency. The notations used in the algorithms are explained in Table 4.1.

4.4.3.1 Strict Routing Protection (SRP)

The motivation of SRP is to always choose the shortest path to route the primary traffic in order to leave more spare capacity for protection, since the capacity used for primary path cannot be shared among different sessions. And then we check whether any available p^2-cycle can be exploited to protect this newly established session. Once being set up, the cycle for a p^2-cycle can not be changed. However, PPLs may be added for protecting the connections whose end nodes are one hop away from the cycle. The detail of the algorithm SRP is shown in Algorithm 4, in which the process can be described in following steps:

1. As a new session $d_l(s_l, t_l)$ arrives, establish the primary path f_l between s_l and t_l under current network status by using Dijkstra's algorithm. If it fails, the session is blocked;

2. Sort all the existing p^2-cycles, $c_p \in \mathbb{C}$, in the increasing order of $\delta(d_l, c_p)$, which is

Algorithm 4: Strict Routing Protection (SRP) Scheme

Input: $G(V, E), \mathbb{D}, \mathbb{P}, \mathbb{C}$
Output: Accepted or Blocked?

Given a new session l, find the shortest path f_l in G;
sort $c_p \in \mathbb{C}$ in the increasing order of $\delta(t_l, c_p)$;
for $c_p \in \mathbb{C}$ *and* $\delta(d_l, c_p)<3$ **do**
    construct a temporary graph $G'=(V', E')$ where $V' = \{\forall v \in c_p\} \vee \{s_l, t_l\}$ and $E' = \{\{\forall e \in c_p\} \vee \{(s_l, v), (v, d_l)\}\} \backslash \{\forall e \in f_l\}$;
    for $d_i \in \mathbb{D}$ *protected by* c_p **do**
        if f_l *and* p_m^i *are not link disjoint* **then**
            $E' = E' \backslash \{e \in p_t^i\}$;
        end
    end
    Run Dijkstra's algorithm to find a protection path q_l between s_l and t_l in G';
    if *Succeed* **then**
        accept session l and update c_p by adding PPLs $e \in q_l$ but $e \notin c_p$;
        update $\mathbb{P}$ and G and exit;
    end
end
if f_l *can not protected by any* c_p **then**
    establish form a new cycle, $c_{|\mathbb{C}|+1}$, to protect f_l;
    if *Succeed* **then**
        add f_l to $\mathbb{P}$ and $c_{|\mathbb{C}|+1}$ to $\mathbb{C}$;
    else
        the request l is blocked;
    end
end

computed as follows:

$$\delta(d_l, c_p) = \begin{cases} 0, & \text{if both } s_l \text{ and } d_l \text{ are the on-cycle nodes of } c_p; \\ 1, & \text{if one of } s_l \text{ and } d_l \text{ is on the cycle and the other is one hop away from the cycle;} \\ 2, & \text{if both } s_l \text{ and } d_l \text{ are one hop away from the cycle of } c_p; \\ +\infty, & \text{otherwise.} \end{cases}$$

One hop indicates that there exists a span in the network that connects a node to the cycle. If $\delta(d_l, c_p) = +\infty$ for all $c_p \in \mathbb{C}$, then no existing cycle is able to protect this new

session. Thus, a new cycle needs to be constructed to protect d_l.

3. For each existing protection cycle, c_p, we construct a temporary graph G', consisting of only the cycle spans of c_p and all the spans connecting the source and destination nodes of l to the cycle (line 4). All the spans used by f_l should be removed to ensure that its protection path is link-disjoint. Then, all the sessions protected by c_p are checked and if an existing session in $\mathbb{D}$ can share the same c_p with the new session l, we should make sure that either their primary paths or their protection paths are link-disjoint. In lines 5-9, we remove the protection paths of all the sessions in $\mathbb{D}$ whose primary paths are not link-disjoint with f_l. If a protection path can still be found in the remaining G' (line 10), this protection path will be q_l for l. Accordingly, the protection cycle is also determined, which should be updated if some PPLs are also used (lines 11-14).

4. If every existing c_p fails to protect d_l, a new cycle will be constructed to protect it. We first attempt to find two diverse paths to form a cycle that is link-disjoint to f_l. If such cycle cannot be found, then we find a path, q_l, link-disjoint to f_l and the cycle is formed by combining q_l with f_l. This last part of the algorithm is described by lines 16-22.

In the worst case, every existing sessions d_i will be examined as to whether its primary path is link-disjoint with the primary path of the new session. It takes total of $O(|\mathbb{D}||E|)$ times for the checking process. All existing p^2-cycles may also be checked. For each c_p, the computation cost of graph and path construction is $O(|E| + |V|^2)$. Therefore, the time complexity of SRP algorithm in the worst scenario is dominated by $O(|\mathbb{C}|(|\mathbb{D}||E| + |V|^2))$.

4.4.3.2 Flexible Routing Protection (FRP)

Different from SRP, the flexible routing protection scheme considers primary and protection paths jointly for each arriving session. Instead of determining the primary path in advance, we examine each existing p^2-cycle and find each potential protection path along the cycle that can connect the source and destination. For each potential protection path, we try to

discover a primary path for it. If it succeeds, the session is accepted. Otherwise, a new cycle is constructed to protect the session. The detailed procedure is described in Algorithm 5.

Algorithm 5: Flexible Routing Protection (FRP) Scheme

Input: $G(V, E), \mathbb{D}, \mathbb{P}, \mathbb{C}, \mathbb{T} = \emptyset$
Output: Accepted or Blocked?

Given a new session $d_l(s_l, t_l)$, sort $c_p \in \mathbb{C}$ in the increasing order of $\delta(d_l, c_p)$;
for $c_p \in \mathbb{C}$ *and* $\delta(d_l, c_p) < 3$ **do**
  if s_l *or(and)* t_l *is not on the cycle of* c_p **then**
    construct each candidate q_l by combining PPL (s_l, u) or(and) (v, t_l) and an on-cycle segment between on-cycle node u and v;
  else
    construct candidate q_l by using an on-cycle segment of c_p between s_l and t_l;
  end
  for *each candidate* q_l **do**
    for $d_i \in \mathbb{D}$ *protected by* c_p **do**
      remove the primary path f_i from G if q_i is not link-disjoint with q_l;
    end
    remove q_l and run Dijkstra's Algorithm to find f_l between s_l and t_l in G;
    if *Succeed* **then**
      add the combination $< c_p, q_l, f_l >$ to $\mathbb{T}$;
    end
  end
end
if $\mathbb{T}$ *is not empty* **then**
  Session l is accepted and choose $< f_l, q_l, c_p >$ with the minimum cost of f_l;
  Add f_l to $\mathbb{P}$ and $e \in q_l$ but $e \notin c_p$ to c_p;
end
else
  Run Bhandari's Algorithm[74] to find two link-disjoint paths between s_l and t_l;
  Choose shorter one as f_l and combine them as a new p^2-cycle $c_{|\mathbb{C}|+1}$;
  if *Failed* **then**
    The request is blocked;
  end
end

Algorithm FRP is explained in following steps:

1. Given a new session $d_l(s_l, t_l)$, all the available p^2-cycles $c_p \in \mathbb{C}$ are sorted in the increasing order of $\delta(d_l, c_p)$.

2. For each available c_p, list all the possible protection paths for d_l. If the end nodes s_l and

t_l are on the cycle, there are two possible segments along the cycle. If s_l or(and) t_l is not on the cycle, the path will be composed of parasitic links connecting s_l or t_l to the cycle and an on-cycle segment.

We assume the average node degree in a given network is denoted by θ. Each cycle can provide two on-cycle segments between any pair of on-cycle nodes. Each end node, s_l or t_l, can be connected to the cycle by at most θ PPLs given the node degree θ. Hence, the average number of candidate protection paths provided by any p^2-cycle for d_l is $2\theta^2$. Lines 3-7 are used to construct all candidate protection paths for d_l.

3. For each candidate q_l, run Dijkstra's algorithm to find a primary path f_l in G that is not only link-disjoint to q_l but also link-disjoint with other primary paths protected by the same cycle if their protection paths are not link-disjoint. If it succeeds, we store the combination $< c_p, q_l, f_l >$ in a temporary set $\mathbb{T}$, which is initialized as $\emptyset$. After checking all the existing p^2-cycles, we check set $\mathbb{T}$ and find the combination $< c_p, q_l, f_l >$ with minimum cost of f_l. We recover the spans removed from G and update the network status. The process is described by lines 8-18.

4. If no existing p^2-cycle can be used to protect session d_l, we use Bhandari's algorithm to find two link-disjoint paths between s_i and t_i to form a new p^2-cycle. If it fails, the session is blocked. Otherwise, the session is accepted and one of the paths (usually the shorter one) is used as the primary path f_l, and the network is updated.

In the worst case, all the existing p^2-cycles will be examined and the total number of protection path candidates provided by a cycle is $2\theta^2$. It takes $2|V|$ operations to check all the candidates of protection path q_l, from line 3 to line 7. For each candidate, a total of $O(|\mathbb{D}||E|)$ time is consumed by checking link-disjointness between q_l provided by each c_p and the protection paths of other sessions. In addition to the time of running Dijkstra's algorithm, $O(|V|^2)$, lines 8-18 consume a total time; that is $O(2\theta^2(|\mathbb{D}||E| + |V|^2))$. This part of the algorithm actually dominates the running time. Therefore, adding up the cost for each iteration of c_p, the total time complexity of FRP algorithm is $O(2\theta^2|\mathbb{C}|(|\mathbb{D}||E| + |V|^2))$. As we

can see, FRP has higher time complexity than SRP due to multiple options of each primary path.

4.5 Performance Evaluation

In this section, we investigate the performances of the proposed p^2-cycle protection scheme under both static and dynamic traffic scenarios. In the static traffic case, the unicast traffic requests are given in advance. We compare p^2-cycle performance with two other path protection schemes, the SBPP and FIPP p-cycle, in terms of two criteria: total capacity cost and average number of reconfigurations (NOR). We also study the performance of p^2-cycle scheme in handling dynamic traffic demands by using the proposed heuristic algorithms, SRP and FRP, and compare them with FIPP p-cycle in terms of overall blocking probability and average NOR.

4.5.1 Results for Static Traffic

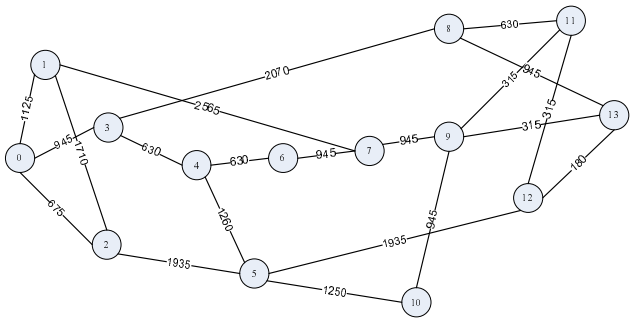


Figure 4.3 NSFNET network (14 nodes, 21 spans)

With the prior knowledge of all the traffic information, we obtain the optimal solutions of the JCP problem by formulating it as an ILP and solving it by a commercial software - ILOG CPLEX 10.1.0 on a Linux server with four Xeon 2.4GHz CPU and 4 GB of RAM. The ILP for SBPP is obtained from [30]. Since the ILP for FIPP p-cycle proposed in [45] does not address JCP problem but only spare capacity assignment, we use the ILP proposed in this paper without using PPLs.

The experiments are conducted on two practical networks, NSFNET and the pan-European COST 239, shown in Fig. 4.3 and 4.4, respectively. Both networks have similar numbers of nodes, but COST239 has a larger average node degree (4.72) than NSFNET (2.7). Each span in the two networks has a cost, which is the actual distance between the two end nodes in kilometers. We assume that the networks have wavelength conversion capabilities and unlimited wavelengths on each span.

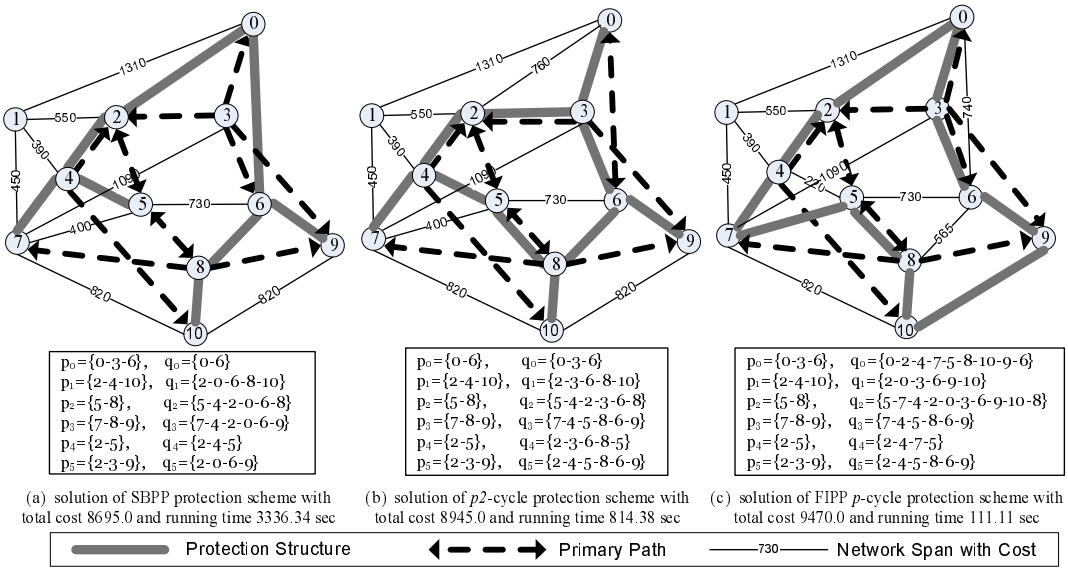


Figure 4.4 Comparison of the total cost of six unitary unicast sessions: $\{(0,6), (2,10), (5,8), (7,9), (2,5), (2,9)\}$ in COST239 network using: (a) SBPP, (b) p^2-cycle and (c) FIPP

4.5.1.1 Total Capacity Cost

We first study a special case in COST239 network in which six unicast traffic requests with unitary traffic rate (one wavelength) need to be provisioned, and we obtain the solutions of each protection scheme, as shown in Fig. 4.4. The source and destination of each session is depicted in a pair of parenthesis, indexed from 0 to 5, counted from left to right. The routes of the primary path, denoted by p, and the corresponding protection path, denoted by q, of each session are described in the boxes.

The optimal solution obtained by employing the SBPP, p^2-cycle and FIPP p-cycle are

presented in Fig. 4.4(a), (b), (c), respectively. One wavelength assigned for protection on each span can be shared by multiple sessions. For instance, in Fig. 4.4(a), the spare capacity on span $(6, 8)$ is shared by sessions 1 and 2 and $(0, 6)$ is shared by sessions 0,1,2,3 and 5. This feature makes SBPP the most capacity efficient scheme but also takes the longest time to obtain the optimal solution due to the high complexity. The p^2-cycle scheme uses more capacity than SBPP but less than FIPP p-cycle. The FIPP scheme uses 8.9% more capacity over the optimal solution achieved by SBPP whereas the p^2-cycle reduces this number to 2.9%, which is very close to the optimal solution, because it utilizes a combination of a smaller cycle and a number of PPLs compared to a large cycle provisioned by FIPP.

We also studied the average performance of each scheme in both NSFNET and COST239 networks and the results are presented in Table 4.2 and 4.3, respectively. Six different traffic scenarios are simulated, in which a total of 2 to 7 sessions are provisioned. In each scenario, we ran 50 independent cases and then took the average value of the total cost. The end nodes of each session were randomly chosen, but the three schemes use exactly the same traffic demands in each case in order to make a fair comparison.

Table 4.2 Comparison of average total capacity cost in NSFNET

No. Sess.	SBPP	p^2-cycle (extra cost(%))	FIPP (extra cost(%))
2	10734.8	12335.4 (14.9)	13310.1 (24)
3	14774.7	16060.5 (8.7)	17515.8 (18.6)
4	19146.1	19759.5 (3.2)	21185.1 (10.6)
5	21818.3	22624.2 (3.7)	24122.7 (10.6)
6	25539.2	26395.2 (3.4)	27514.8 (7.7)
7	29525.2	30327.5 (2.7)	31518.5 (6.8)

Table 4.3 Comparison of average total capacity cost in COST239

No. Sess.	SBPP	p^2-cycle (extra cost(%))	FIPP (extra cost(%))
2	3602.3	4261.7 (18.3)	4669.6 (29.6)
3	4888.3	5496.3 (12.4)	5820.8 (19.1)
4	6268.2	6794.6 (8.4)	7094.5 (13.2)
5	7530.0	7963.9 (5.8)	8277.6 (9.9)
6	8468.8	8674.8 (2.4)	8947.8 (5.7)

The first column denotes the number of sessions. In the third and fourth column, the extra cost over the optimum is calculated as $(cost - optimum)/optimum$, where optimum is achieved by SBPP. We can observe that the p^2-cycle always achieves better results than FIPP in each scenario. The capacity efficiency of cycle-based protection schemes increases as the number of sessions increases, since there are not enough connections to share the protection of cycles when the traffic is low. As the number of sessions increases, a cycle is more likely to protect multiple connections and become more capacity efficient. When the number of sessions is large in both networks, the p^2-cycle becomes extremely efficient and only uses less than 3% extra cost over the optimal solution. Hence, one may conclude that p^2-cycle will be extremely close to the optimal solutions as the traffic keeps on increasing.

4.5.1.2 Average Number of Reconfigurations

We also compare the traffic recovery performance of p^2-cycles to the other two protection schemes in terms of the average number of reconfigurations (NOR) per connection. It is straightforward to obtain the NOR for each connection protected by FIPP p-cycle and p^2-cycle schemes, respectively, given the primary and protection paths for each connection. But it is not as straightforward for SBPP scheme due to the complex protection structure and capacity sharing. Given a session with its primary and protection paths, if there are any links on the protection path whose the protection wavelengths are shared with other sessions, the nodes adjacent to these links on the protection path are potential reconfiguration nodes. In order to further decide whether such nodes need reconfiguration upon the failure on the primary path, we combining all the protection paths that share the same wavelengths into a protection structure. A potential node requires reconfiguration upon a network failure on the primary path if its nodal degree is greater than 2 in this protection structure, since this node needs to reconfigure its switch to establish protection paths for different sessions that share the same backup resource on the same link. We obtain the NOR under SBPP by assigning a specific wavelength for each protection unit.

The results of the average NOR in both NSFNET and COST239 networks under SBPP,

Table 4.4 Comparison of average NOR per connection in NSFNET

No. Sess.	SBPP	p^2-cycle	FIPP
2	2.48	2.47	2
3	2.77	2.55	2
4	2.84	2.42	2
5	3.01	2.43	2
6	3.09	2.34	2
7	3.19	2.31	2

Table 4.5 Comparison of average NOR per connection in COST239

No. Sess.	SBPP	p^2-cycle	FIPP
2	2.36	2.87	2
3	2.43	2.76	2
4	2.66	2.63	2
5	2.80	2.52	2
6	2.89	2.42	2

p^2-cycle and FIPP are presented in Table 4.4 and 4.5. The results are obtained by taking the average value over 50 independent cases in each traffic scenario. Clearly, FIPP achieves the best solution since it always takes only two end nodes to reconfigure upon a failure. On the other hand, the average NOR of SBPP increases as the number of connections increases, since the structures gets more complex due to sharing backup resources and results in more potential reconfiguration nodes. On the contrary, the average NOR of the p^2-cycle scheme actually decreases. One of the reasons is that a larger number of connections usually results in cycles with larger size. Hence, more nodes will be covered by the cycle such that fewer connections will use PPL as a part of protection path. Thus, more sessions require only two node reconfigurations upon a failure.

Based on the results, except for the first two scenarios Table 4.5, the p^2-cycles always perform better than SBPP in terms of NOR and the advantage becomes more significant and gets closer to that of FIPP as the number of sessions increases. In particular, when there are 7 connections in NSFNET, the NOR of p^2-cycles is equal to 2.31, which is only 15% more than the optimal number of 2, achieved by the FIPP p-cycle, compared to 3.19, obtained from

SBPP. In practice, networks usually accommodate much more than three connections. Thus, we can predict that p^2-cycle will perform very close to FIPP p-cycle in terms of NOR when the traffic demands become large.

4.5.2 Results for Dynamic Traffic

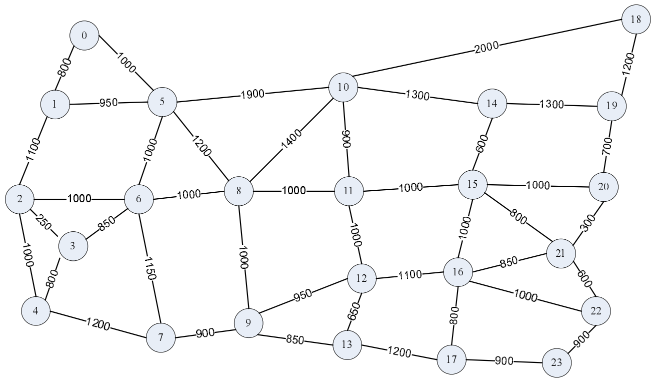


Figure 4.5 USNET network (24 nodes, 43 spans)

Based on two p^2-cycle protection algorithms, SRP and FRP, proposed for provisioning dynamic requests, we conduct a simulation study to compare the performance of these algorithms under dynamic traffic. The networks used in the simulations are NSFNET, COST239 and USNET, in which USNET network, shown in Fig. 4.5 has 24 nodes and 43 edges and the average node degree is 3.58.

In each simulation run, 1000 randomly generated unicast requests are loaded to the network sequentially and the reject ratio is recorded. The arrival of traffic follows Poisson distribution with λ requests per second and the duration of an accepted connection is exponentially distributed with a mean of μ. The traffic load measured in Erlangs is $\lambda\mu$. Each connection requires an entire wavelength to transmit the traffic. The maximum capacity on each network link is set to 16 wavelengths.

Figures 4.6, 4.7 and 4.8 show the blocking probability of dynamic traffic using SRP, FRP and FIPP p-cycle in NSFNET, USNET and COST239 networks, respectively. Each point in the figures is the average value of 200 simulation runs for each traffic load. For FIPP p-cycle

scheme, the primary path of each arriving connection is provisioned first by using Dijkstra's algorithm, and then protected by a p-cycle.

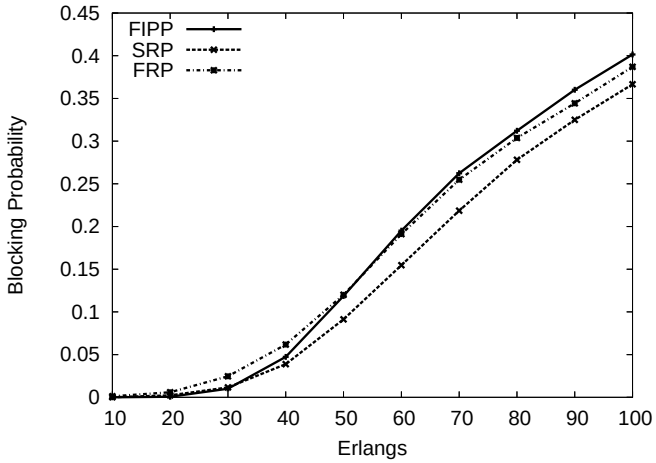


Figure 4.6 Comparison of blocking probability in NSFNET (W=16)

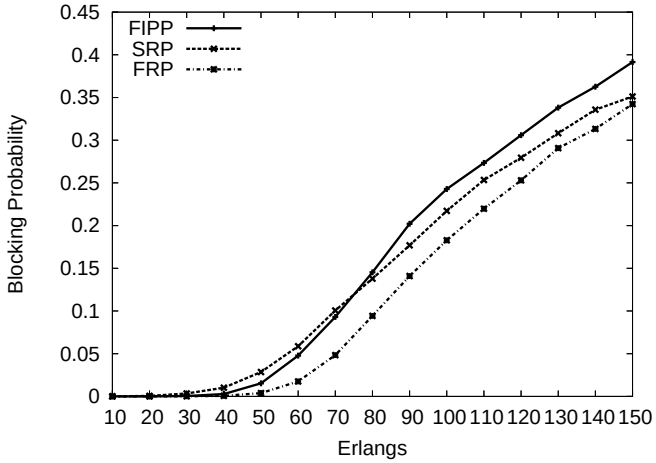


Figure 4.7 Comparison of blocking probability in USNET (W=16)

The results show that both SRP and FRP achieve lower blocking probability than FIPP under most of the network scenarios. In NSFNET, SRP achieves better performance than the other two schemes. In USNET, FRP outperforms SRP and FIPP under every scenarios. In COST239, however, SRP and FIPP achieves the same session blocking ratio, which is better than FRP, when the traffic load is relatively low. As the traffic load increases where the network is very saturated, FRP turns to perform better than SRP and FIPP.

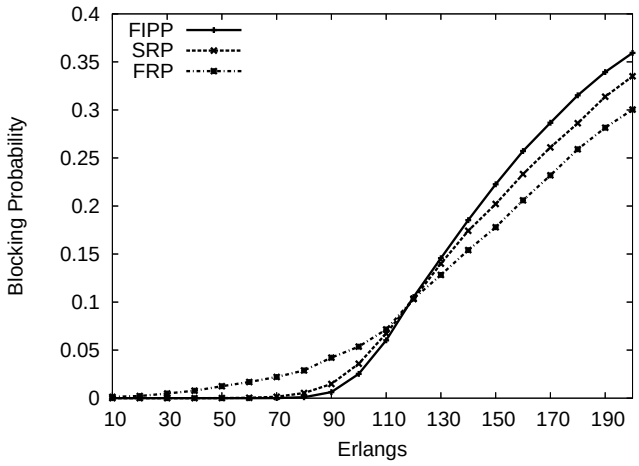


Figure 4.8 Comparison of blocking probability in COST239 (W=16)

The advantage of our schemes over FIPP is denoted by $BP_{FIPP} - BP_{scheme}/BP_{scheme}$ where BP denotes blocking probability. In NSF, the advantage of SRP over FIPP is between $8\% - 23\%$ except for the first two traffic scenarios. When traffic load is very low, traffic requests are rarely blocked for any scheme. In USNET, the advantage of FRP over FIPP is more significant comparing to the advantage of SRP in NSFNET, which lies between 13% and 63% when the BP becomes substantial. In COST239, FIPP actually performs the best and SRP always as the same as FIPP when the traffic load is low. Once the traffic load reaches the threshold where traffic load equals 120 in Erlangs and the BP of all the schemes are always the same, FRP begins to outperform other two. Hence, FRP can provide high probability to protect a given session using existing cycles. However, it may end up with using a long primary path or protection path such that the resources may not be utilized the most efficiently in a long run.

Based on the results, SRP performs better than other two schemes in relatively small and sparse networks at a low level of traffic load. FRP achieves the best performance in larger and denser networks, especially when the network is very saturated. One of the reason that SRP performs better in small and sparse networks, such as, NSF, is that to provision a session always using the shortest path will save some capacity for protection in a long run. Hence, more capacity can be used for protection such that more cycles can be established. However,

in a network with high nodal degree, a cycle is more likely to reach a large group of nodes compared with a sparse network. In this case, FRP has a higher chance to protect a given session by using existing p^2-cycles when network load is very high and the network is over saturated.

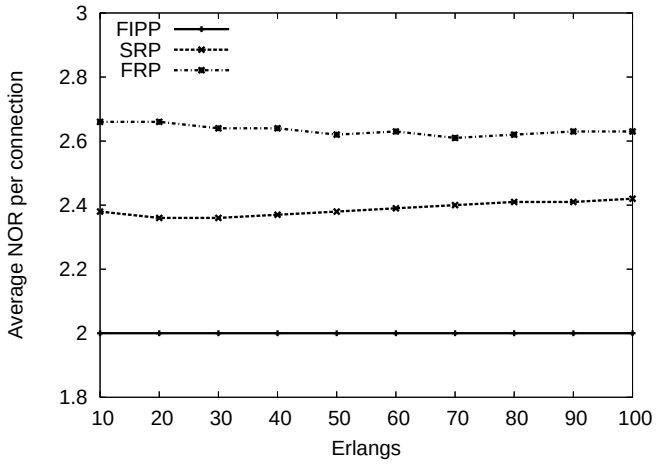


Figure 4.9 Comparison of NOR in NSFNET (W=16)

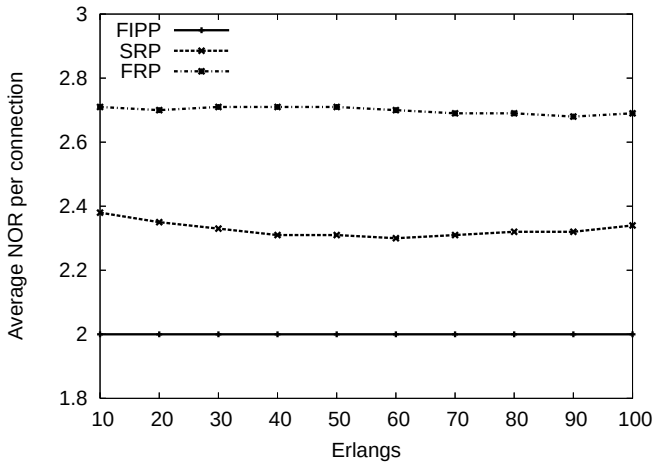


Figure 4.10 Comparison of NOR in USNET (W=16)

We also studied the average NOR of each accepted connection as in dynamic traffic scenarios and the results are shown in Figures 4.9, 4.10 and 4.11. As expected, FIPP achieves the best solution with exact two node reconfigurations for each connection. Meanwhile, SRP also performs better than FRP in three networks. This reveals that connections protected by FRP

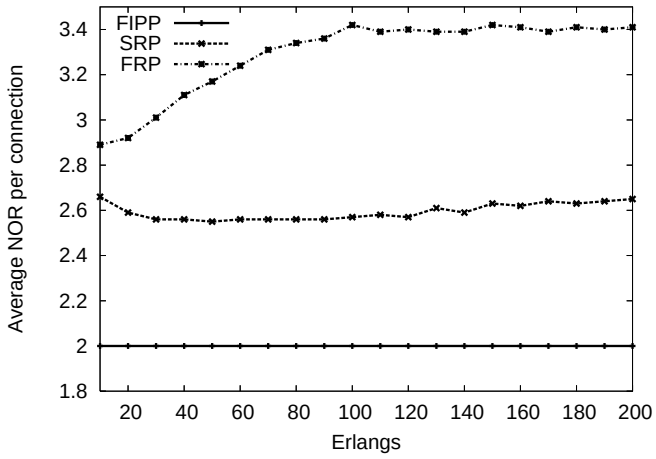


Figure 4.11 Comparison of NOR in COST239((W=16)

use more PPLs than those used by SRP, which follows from the basic concept on which the two algorithms are based. It is worth noting that the average NOR achieved by SRP is almost stable below 2.4 in NSF and USNET and 2.7 in COST239. This indicates that most of the connections only need two no reconfigurations upon a network failure, especially in NSF and USNET. FRP has larger average NOR because it iterates every existing p-cycle in the network to protect each session and choose the one with minimum cost but not the one with minimum NOR. Shorter primary paths always results in longer protection paths such that more PPLs are used to protect each session.

Therefore, based on the simulation results, SRP and FRP both achieves the lowest blocking probability than FIPP in most of the network scenarios considered and each scheme has advantage over the other in different network scenarios. SRP has better failure recovery performance than FRP. Considering both static and dynamic traffic scenarios, the p^2-cycle protection scheme is a much faster protection scheme than SBPP and provides an enhancement of capacity efficiency over the FIPP p-cycle with a small increase in the recovery time.

4.6 Conclusions

In this paper, we proposed a new p-cycle based protection scheme in mesh network, named p^2-cycle, by augmenting the FIPP p-cycle with attached parasitic protection links (PPL) in

order to enhance the protection ability by protecting the paths whose end nodes are not located on the cycle but only one hop away from the cycle. We studied both static and dynamic traffic scenarios. In static traffic case, we solved the joint capacity placement (JCP) problem optimally by formulating the problem as an ILP such that the total working and protection capacity used for all the sessions are minimized. In the dynamic traffic case, we proposed two heuristic algorithms, Strict Routing Protection (SRP) and Flexible Routing Protection (FRP), to handle dynamic traffic demands in order to minimize the total number of blocked sessions.

Based on the numerical results, the p^2-cycle protection scheme is a more capacity efficient than the FIPP p-cycle scheme in both static and dynamic traffic scenarios. In static scenario, it achieves a cost close to the optimal solution, achieved by SBPP, given a large number of session as a priori. Meanwhile, the p^2-cycle has much better recovery performance than SBPP in terms of NOR as traffic demands increase. In the dynamic traffic scenario, the p^2-cycle based approach, SRP and FRP achieve better blocking probability than FIPP does in most of the scenarios considered. Considering the trade-off between capacity efficiency and recovery speed, the p^2-cycle protection scheme is a more effective alternative of existent p-cycle-based and path-based protection schemes.

CHAPTER 5. TREE-BASED PROTECTION OF MULTICAST SERVICES IN WDM MESH NETWORKS

A paper ready for submission [1]

Long Long and Ahmed E. Kamal

Abstract

In this paper we address the multicast survivability problem of using minimum resources to provision a multicast session and its protection paths (trees) in a network such that the session is protected against any single-link failure. We propose a new protection scheme, namely, Segment-based Protection Tree (SPT). In SPT scheme, a given multicast session is first provisioned as a primary multicast tree, and then each segment on the primary tree is protected by a multicast tree instead of a path, as in most existing approaches. We also analyze the recovery performance of SPT and design a Reconfiguration Calculation Algorithm to compute the average number of reconfigurations upon any link failure. By extending SPT to address dynamic traffic scenarios, we also propose two heuristic algorithms, Cost-based SPT (CB_SPT) and Wavelength-based SPT (WB_SPT). We study the performance of the SPT scheme in different traffic scenarios. The numerical results show that SPT outperforms the best existing approaches, optimal path-pair-based shared disjoint paths (OPP_SDP). SPT uses less than 10% extra resources to provision a survivable multicast session over the optimal solution and up to 4% lower than existing approaches under various traffic scenarios and has an average number of reconfigurations 10-86% less than the best cost efficient approach. Moreover,

[1] Part of this work has been published in *IEEE Globecom* 2009 [75].

in dynamic traffic cases, both CB_SPT and WB_SPT achieves overall blocking probability with 20% lower than OPP_SDP in most network scenarios.

5.1 Introduction

Wavelength-division multiplexing (WDM) technology allows an aggregate traffic on the order of Tbps to be carried on a single fiber, with each wavelength carrying traffic in the tens of Gbps order. Such advances meet the explosive increase of bandwidth demand in the Internet and enable a greater variety of network applications to be served [76]. Several of these applications employ the multicast service mode, such as video distribution, online gaming and so on. To implement multicasting, a node should have the capability to replicate an incoming packet into multiple copies. In the context of optical networks, there are two ways to implement the multicast function at a node, unicast and multicast. In unicast mode, traffic duplication can only be implemented in the electronic domain, whereas in multicast mode, traffic duplication can be done in the optical domain by using optical splitters [80]. If a multicast session is provisioned as a tree in the optical domain, it is called a "light-tree" which originates at a source node and delivers the same data to a number of destination (leaf) nodes [78].

As the capacity of fibers keeps on increasing, a fiber cut caused by an accident or a failure of a switch port or a node interface may lead to loss of tremendous amounts of data. In the scenario of multicast service, data loss on one fiber may cause the disruption of delivery to multiple nodes. Therefore, efforts have been exerted to deal with protection of a multicast session against single link failures. A straightforward method proposed in [77] is to find two link disjoint light trees and both of them start from the source and end at the destination nodes. It is clear that this method is not capacity efficient since it is not always possible to find two link disjoint trees in a network. In [79], the authors introduced a number of protection schemes: link-based, segment-based and path-based. In link-based and segment-based approaches, a multicast session is routed first to construct a multicast tree, and then each link or segment on the tree is protected by a path starting at the tail node and finishing at the head node of the link or segment it protects. Alternatively, a path-based protection scheme, named

optimal path-pair-based shared disjoint paths (OPP_SDP) algorithm, achieves the best result in terms of network resource consumption in [79] by self-sharing primary and spare capacity [81]. The idea is to find two shortest link disjoint paths for each source and destination pair [100]. Recently, a couple of new technologies were applied to the survivability problem, p-cycle [83] and network coding [84]. These techniques do have some nice features such as the fast recovery of p-cycles or high bandwidth utilization of network coding. However, p-cycle-based schemes are not efficient and flexible to protect dynamic traffic, especially multicast traffic, while network coding introduces extra computational cost as well as O-E-O conversion since network coding can only be performed in the electronic domain in current optical networks, which may introduce an additional expense.

A path-based scheme, called multicast protection through spanning paths (MPSP), proposed in [82], outperforms OPP_SDP under both static and dynamic traffic patterns. It first provisions a primary multicast tree and then establishes a number of paths to protect each path between any pair of leaf nodes on the primary tree, called spanning path. Each path is link disjoint from the spanning path it protects. However, this scheme relies on the assumption that wavelengths reserved in a fiber can be used in two opposite directions by reconfiguring the switches at two end nodes. However, this feature cannot be achieved in practice. Between two connected nodes, there are usually two physical fibers set up and each of them works in one direction. The switches at end nodes use input and output ports to connect incoming and outgoing fibers, respectively [79, 80]. Reserved capacity (wavelength) in a fiber cannot be used in both directions by simply reconfiguring the switches at end nodes due to the fixed switching ports. One way to enable this feature is to change the physical infrastructure by deploying a pair of circulators between two nodes as shown in Fig. 5.1. The fiber is connected to the circulators instead of switching ports on the switches. The circulators connect to both input and output ports on the nearby switches and can configure the fiber to connect to either input port or output port. Only changing the configuration of both switches and circulators will make the transmission in both directions on the same fiber possible such that one unit of capacity reserved in a directed link can be shared by primary and protection paths in MPSP

scheme. Due to the infrastructure of current backbone networks, the lack of support for this functionality and the restrictions this imposes on other modes of communication, we do not take this assumption into consideration in our proposed scheme.

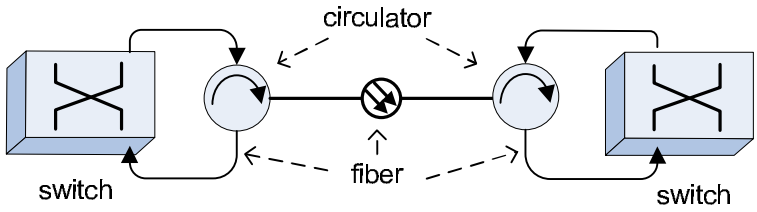


Figure 5.1 Additional depolyment of circulators enables capacity sharing in opposite directions of a fiber

A tree-based protection scheme, segment-based protection tree (SPT) algorithm, is proposed in this paper to provision a multicast request and protect it against any single link failure. We first provision the multicast session on a light tree and then construct protection multicast trees instead of paths to protect the primary light tree. Each protection tree, similar to primary tree, is rooted at the source and reaches every destination in the session. Each segment on the primary tree is protected by a protection tree. A protection tree can share any link with the primary tree as well as other protection trees. The uniqueness of our schemes is that each protection tree is a complete multicast tree from source to destinations. It does not have to traverse the end nodes of a segment it protects. In this case, multiple segments may share one protection tree, which potentially improves the efficiency of the bandwidth utilization.

The rest of this paper is organized as follows. In Section 5.2, we present the assumptions and statement of the problem addressed. The proposed scheme, SPT, will be introduced in Section 5.3. The method of computing the average number of reconfigurations will be presented in Section 5.4. We further study the dynamic multicast cases by proposing two heuristic algorithm extended from SPT in Section 5.5. Numerical results will be presented and explained in Section 5.6. Finally, we conclude this paper in Section 5.7.

5.2 Preliminaries

In this section, we first describe multicast protection problem addressed in the paper with the corresponding assumptions. We then summarize a number of multicast provisioning methods, some of which will be used in the scheme proposed in the paper.

5.2.1 Multicast Protection Problem

A typical multicast session is unidirectional whereas the links of a typical WDM mesh network are bidirectional, since each link has two optical fibers transporting signals in two opposite directions with the same capacity. Each directed fiber is also called "an arc" in [79]. Meanwhile, each arc is assigned a value to indicate the cost of transmitting the data from one end to the other. The cost usually refers to the length of the physical fiber.

We make the following assumptions and present the formal statement of the multicast protection problem:

1. Given a weighted directed connected graph $G = (V, E)$ in which each directed link[2] $e = (u, v) \in E$ where $u, v \in V$ is assigned a weight (cost) c_e and a capacity with W wavelengths. The graph, G, is at least 2-connected.

2. Given a directed multicast request d with a source node s and a set of destinations $\{t_1, t_2, ..., t_M\}$ where $s, t_i \in V$ and M is the number of destination nodes. The traffic requirements of the session is equal to one wavelength. d is expressed as $(s, \{t_1, t_2, ..., t_M\})$.

3. A single link failure will cut off the links in both directions such that traffic delivered in both fibers will be lost. Thus, when we claim two link-disjoint paths (trees) in this article, it indicates that two paths (trees) do not travel the links with the same end nodes in any direction.

4. In this article, we assume that each network node is equipped with an optical switch, optical splitters and wavelength converters if necessary.

[2]Here we use "link" to represent "arc" similar to [79] and therefore links (u, v) and (v, u) are two different links but have the same cost and capacity.

The multicast protection problem is described as follow:

Given a weighted graph $G = (V, E)$ and a multicast request d, find a provisioning of the multicast session d such that the multicast service is survivable against any single link failure in G using the minimum cost.

5.2.2 Multicast Provisioning Methods

In order to provision a survivable multicast session with minimum cost, it is essential to study how to provision a multicast request. In optical transport networks, multicast provisioning problem can be referred to as finding a light-tree that delivers data from the source to all the destinations with the minimum cost. Deployment of optical splitters at each network node enables multicast implementation in the optical domain. Thus, this problem turns out to be a classic graph theory problem, "Steiner tree problem", which has been proven *NP-complete* [97]. Hence, the multicast protection problem is also *NP-complete* in the general case and this is why we develop heuristic solution approaches in this paper.

Many approximate algorithms have been proposed in the literature such as Nearest Participant First (NPF) algorithm [98], KMB algorithm [99], pruned Prim′s heuristic [100], referred to as PPH and so on. We actually consider three multicast schemes in the construction of multicast tree: NPF, pruned Prim′s heuristic and simply using Dijkstra's Shortest Path algorithm, namely, DST, to find the shortest path from the source to each destination and combining all the paths to construct a multicast tree.

The heuristic NPF is a greedy-based algorithm with time complexity $O(M|V|^2)$. The procedure is explained as follow:

1. start from the source node;
2. find a destination node that is closest to the current tree;
3. connect the closest destination node to the closest part of the tree;
4. repeat until all the destinations are connected in the tree.

Prim′s algorithm is a well known approach of finding the minimum spanning tree with time complexity $O(|V|^2)$. Based on the minimum spanning tree obtained, PPH trims the unwanted branches such that the resulting multicast tree only reaches the given destinations. The total time complexity is $O(|V|^2 + M|V|)$.

The algorithm DST, with time complexity $O(M|V|^2)$, is straightforward and is actually a special case of NPF by assuming that the source is the only node on the current tree. Thus, a multicast tree produced by DST always has equivalent or higher cost than what NPF produces.

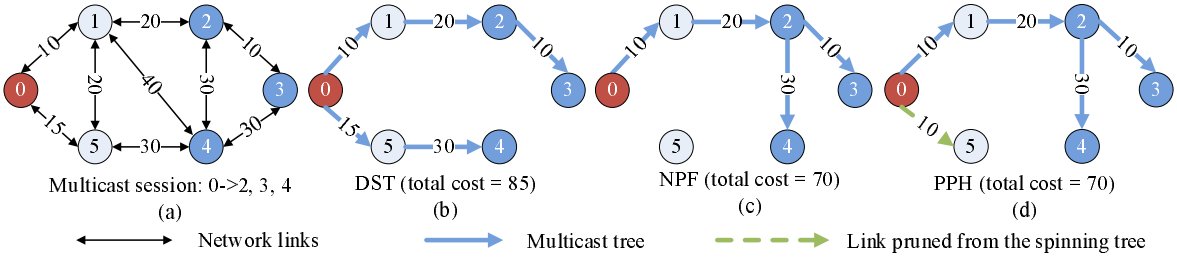


Figure 5.2 Demonstration of various multicast provisioning algorithms

An example shown in 5.2 illustrates the multicast trees constructed by employing various algorithms described above. Given a multicast session that sourced at node 0 and destined to node $\{2, 3, 4\}$, NPF and PPH construct the same multicast tree and achieve less cost than DST does.

5.3 Tree-based Protection Scheme

In this section, we present the tree-based protection scheme, SPT, to provision a multicast request against any single-link failure using minimum cost.

SPT scheme consists of two phases. The first phase is to construct three primary multicast trees by using the three methods, NPF, PPH and DST, repectively, introduced in the section 5.2. The second phase is to provision a protection structure to protect each primary multicast tree established in the first phase. A final survivabile multicast session is established by combining the primary multicast tree and its corresponding protection structure. We choose the final topology with the minimal total cost among three methods. Although the multicast tree

obtained by using DST has the same or higher cost than that achieved by NPF. The reason to consider DST in only phase one is that the objective considered in our problem is the total cost of the final survivable multicast session other than the primary multicast tree only. A primary multicast tree with higher cost in phase one may end up with requiring a protection structure with lower cost to protect it.

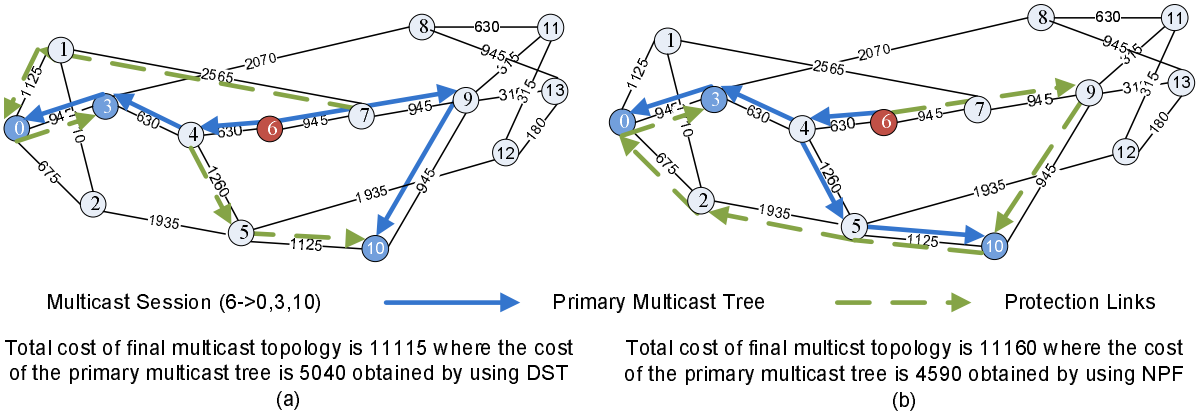


Figure 5.3 Comparison of the final topologies with the primary multicast trees constructed by using DST and NPF, respectively.

An example shown in Figure 5.3 illustrates this property. We have a multicast session sourced at node 6 and destined at node 0, 3 and 10. The primary multicast tree constructed by using DST has cost 5040, which is higher than that achieved by NPF, which is 4590. However, the final survivable topology of this multicast session with the primary tree constructed by DST has lower cost than that with NPF, since it uses the protection link with lower cost. Therefore, the total cost of the combination of primary tree and protection structure can still be lower than the structure by using other multicast schemes in phase one.

In the second phase, we find a protection structure for each primary tree obtained in phase one. Each primary multicast tree is decomposed into a number of segments. Following the definition in [79], a segment is defined as the sequence of links from the source or any branch node (on a tree) to a leaf node or to a downstream branch node. For each segment of the tree, the SPT scheme establishes another multicast tree to protect it, called "protection tree". A protection tree is generated by running both NPF and PPH and selecting the one with the less

network cost. We do not consider DST here because DST is a special case of NPF and can never produce better solution than NPF. Each protection tree must not traverse the segment it protects. However, it is not necessary for it to pass two end nodes of any segment it protects either. Any protection tree is a complete multicast tree rooted at the source and destined to all the destinations regardless of which segment it protects.

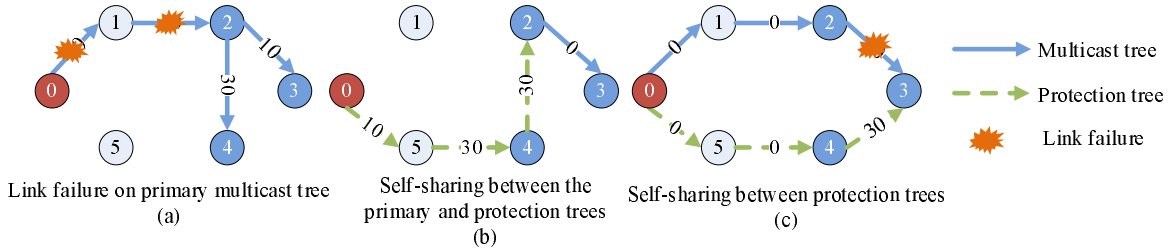


Figure 5.4 An example of Self-sharing

We apply the self-sharing introduced in [81] to our protection scheme. Self-sharing means that a backup path/tree can share capacity not only with other backup paths/trees but also with other edges on the primary tree. An example of self-sharing is illustrated in Fig. 5.4. If a link failure occurs on either link $(0,1)$ or $(1,2)$ as shown in Fig. 5.4(a), we need a protection tree to protect the segment $\{(0,1),(1,2)\}$, and the tree is shown in Fig. 5.4(b). Three links used by the protection tree are newly reserved and they are $\{(0,2),(2,4),(4,3)\}$. However, one capacity reserved on link $(2,3)$ for primary tree can be shared with this protection tree and thus the cost of this link in the protection tree is 0. The self-sharing between two protection trees are shown in Fig. 5.4(c). To protect against the failure of link $(2,3)$ in the primary tree, the second protection tree is constructed. However, four links used by this tree have already been reserved by the primary and the previous protection trees, which can be shared. Therefore, only one link should be newly reserved, which is link $(4,3)$. Accordingly, the cost of rest links are equal to 0, too.

Based on the self-sharing rule, we construct the algorithm, Segment-based Protection Tree (SPT). Before describing the algorithm, we introduced symbols used in the algorithm are explained in Table 5.1:

Given a multicast session, we call the combination of primary and all the protection trees as

Table 5.1 Notations used in the Algorithms

Notation	Meaning
T_m^k	the kth primary multicast tree obtained by heuristic k, where $0 \leq k < 3$ and $0, 1$ and 2 represents heuristic algorithm NPF, PPH and DST, respectively
$T_{p_i}^k$	the ith protection tree for primary multicast tree k
$\mathbb{P}_k$	the union of all the protection trees for kth primary tree, denoted by $\bigcup_i T_{p_i}^k$
$\mathbb{R}_k$	the union of all links used for the multicast session generated by heuristic k, denoted by $T_m^k \bigcup \mathbb{P}_k$
c_e	the cost of link $e \in E$

final survivable topology. Working traffic will be transmitted through the primary tree under the normal condition. The primary tree is divided into a number of segments and each of them is a basic protection unit. All the links reserved for protection other than the primary tree in the final topology is called "pure protection links". The algorithm SPT is to find the final topology with a cost that is as low as possible.

The SPT is presented in Algorithm 6. Each segment in the primary tree is denoted by $l \in T_m^k$. If any existing protection tree established earlier does not traverse l and its counterpart in the opposite direction[3], then l is protected by this tree upon any failure of link $e \in l$. If no such protection tree exists, a new protection tree needs to be provisioned. However, the new tree can share any link with all the established trees in $\mathbb{P}_k$ as well as the primary tree T_m^k in the modified graph G' with removal of l. Hence, we set the cost of all links available for sharing as 0. Then, algorithm NPF and PPH are executed to obtain the new protection trees $T_{p_i}^k$ and $T_{p_{i'}}^k$ and the one with the less link cost will be selected and added into the protection tree set $\mathbb{P}_k$ in which the links that do not exist in the final set $\mathbb{R}_k$ will also be added. In the final step, three final sets with three different primary trees are compared and we choose the one with the minimum cost, $\mathbb{R}_{\min}$, as the final survivable topology.

Since the number of links of a tree is less than $|V|$, in the worst case, the number of segments on a primary tree cannot exceed $|V|$. Therefore, the time complexity of the heuristic SPT is

[3]In the rest of the paper, when we say a tree does not travel a link or segment, it indicates that the tree does not travel the link or the segment in either direction

Algorithm 6: Segment-based Protection Tree Algorithm (SPT)

Input: $G(V, E), d = \{s, t_i\}$ $(1 \leq i \leq M)$
Output: $\mathbb{R}_{\min}$

```
for k = 0; k < 3; k++ do
    construct T_m^k by running kth heuristic;
    foreach segment l ∈ T_m^k do
        if ∃T_{p_i}^k ∈ ℙ_k, s.t. l ∉ T_{p_i}^k then
            continue;
        end
        else
            remove e ∈ l from E;
            set c_e = 0, ∀e ∈ ℝ_k;
            run NPF and PPH to obtain protection trees T_{p_i}^k and T_{p_i'}^k, respectively in G;
            select the T_{p_i}^k with less cost and add it to ℙ_k;
            add e to ℝ_k, ∀e ∈ T_{p_i}^k and e ∉ ℝ_k;
            recover c_e where e ∈ l;
        end
    end
    if the cost of ℝ_k is less than that of ℝ_min then
        ℝ_min = ℝ_k;
    end
end
```

$O(M|V|^3)$.

5.4 Reconfiguration Calculation

Besides the network cost, the recovery time, referred to as the time period from the occurance of the failure to the restoration of the traffic, is another important criterion to evaluate the performance of a protection approach. The recovery process consists of several stages: failure detection, signaling transmission and switch reconfiguration, in which switch reconfiguration process consumes the most part of recovery time, since each reconfiguration takes 10 - 20 ms [101] depending on the technology used. Therefore, it is essential to figure out the average reconfiguration time upon any link failure in a network.

Based on the SPT approach proposed in Section 5.3, a multicast tree is provisioned first and then each segment on the tree will be protected by a protection tree. Thus, given a failure

in a network, if this link happens to be used by the multicast tree, a protection tree will be activated to protect it. Accordingly, some nodes on the protection tree may be required to reconfigure the switches to reroute the traffic. The rule to determine whether a node needs to reconfigure its switch is whether this node receives the incoming traffic from a different node or forwards it to a different output node in the protection tree compared to that in the primary multicast tree.

In order to obtain the average number of reconfigurations upon any link failure that disrupts a given multicast service, we assume that the primary tree T_m consists of L links and upon the failure of link $e \in T_m$, a protection tree T_{p_i} is activated and r_i nodes on T_{p_i} will reconfigure the switch. Therefore, the average number of reconfigurations given any link failure is denoted by:

$$R_{\text{avg}} = \frac{\sum_{e \in T_m} r_i}{L} \text{ , where } T_{p_i} \text{ protects } e \tag{5.1}$$

Based on the previous analysis, we propose Algorithm 7 to compute the average reconfiguration time with the application of SPT approach. Several symbols used in the algorithm are explained in Table 5.2:

Table 5.2 Symbols used in the Reconfiguration Calculation

Symbol	Meaning
L	total number of links in the primary tree T_m
X	the set of nodes that consists of $\{s, t_i\}$ and the nodes that have node degree more than 2 in the final survivable topology $\mathbb{R}$
R_{avg}	the average number of reconfigurations given any single link failure

In the Algorithm 7, the set X maintains all the potential nodes that may reconfigure the switch upon a link failure. Any node in the final survivable topology $\mathbb{R}$ has node degree at least two, since $\mathbb{R}$ is 2-connected. Except the source s, every node has at least a parent. If a node has nodal degree 2, the incoming and outgoing links that the traffic passes through will always be fixed and there is no need for reconfiguration. Therefore, we only consider the nodes with node degree at least 3 along with the source and the destinations as the potential nodes. In the algorithm, line 5 checks whether node v needs reconfiguration or not. If yes, line 6 increases the total number of reconfigurations. Therefore, the average number of reconfigurations is

Algorithm 7: Reconfiguration Calculation Algorithm of SPT

Input: $T_m, \{T_{p_i}\}, X$
Output: R_{avg}

```
1  R_avg = 0;
2  for e ∈ T_m do
3      if ∃T_{p_i} protects e then
4          for ∀v ∈ X do
5              if ∃(u, v) or (v, u) ∈ T_{p_i} but ∉ T_m then
6                  R_avg++;
7              end
8          end
9      end
10 end
11 R_avg = R_avg/L;
```

obtained by the total number divided by the total number of the links in the primary tree shown in line (11). The time complexity of Algorithm 7 is $O(L|V|^2)$.

5.5 Dynamic Traffic Protection

In this section, we extend SPT algorithm to address dynamic traffic provisioning problem. We introduce another sharing method, cross-sharing, introduced by [81]. By applying both self-sharing and cross-sharing to deal with dynamic traffic, we propose two algorithms, Cost-based SPT and Wavelength SPT, by extending SPT algorithm such that not only the resources can be shared within a session but also among different sessions.

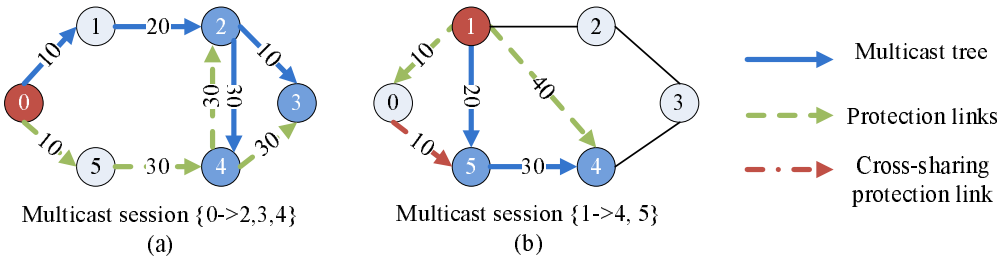


Figure 5.5 An example of Cross-sharing

The basic idea of cross-sharing is to share pure protection links among different multicast sessions. An example is shown in Figure 5.5 to illustrate cross-sharing. Two sessions arrive one

after the other and the session $\{0->2,3,4\}$ comes ahead of the session $\{1->4,5\}$. The final topology of the first session is shown in Fig. 5.5(a). As the second session comes, the primary multicast tree is $\{(1,5),(5,4)\}$ and the pure protection links are $\{(1,0),(1,4),(0,5)\}$. Since the primary trees of session one and two are link disjoint, so the pure protection link $(0,5)$ is shared such that only one protection capacity is required to protect both sessions. This feature will be used to provision dynamic traffic.

Based on the rule of cross-sharing, we combine both self-sharing and cross-sharing to provision each dynamic multicast session and protect them against single-link failure. We propose two heuristic algorithms, Cost-based SPT (CB_SPT) and Wavelength-based SPT (WB_SPT). Each heuristic is still based on SPT algorithm proposed in the section 5.3 for multicast protection. However, we also introduce cross-sharing such that each pure protection link can be used to protect multiple sessions. The distinction between CB_SPT and WB_SPT is that we use different measurements to choose the final topology for each incoming session. The former chooses the final topology with the minimum overall cost, whereas the latter chooses the final topology with the minimum number of wavelengths used.

The first heuristic, CB_SPT, is shown in as Algorithm 8. We still use three multicast algorithm, DST, NPF and PPH, to obtain three primary multicast trees. For each segment in a multicast tree, we try to find a protection tree to protect it as SPT does. The difference here is that any protection tree can not only use links in the primary tree and other established protection trees, but also pure protection links from other established multicast sessions.

In the algorithm, lines 1 and 2 construct three primary trees. For each segment, line 4-6 check whether this segment is protected already by an existing protection tree. If yes, continue to the next segment. Otherwise, a new protection tree that is link disjoint to the segment has to be provisioned. In order to cross share as many links as possible. We need to record every session in the network and the detailed information of which protection link is used to protect which segments. Thus, we iterate every session and every pure protection link in it. If the segment protected by a protection link is link-disjoint to the segment we are trying to protect, this link is a potential free link to use. We set its cost to 0. The process is described by lines

Algorithm 8: Cost-based SPT Algorithm (CB_SPT)

Input: $G, d = \{s, t_i\}$ $(1 \leq i \leq M)$, $\text{Cost}_{\mathbb{R}_{\min}}=\infty$
Output: accept or block?

```
1  for k = 0; k<3; k++ do
2      construct T_k^m by running kth heuristic;
3      for each segment l ∈ T_m^k do
4          if ∃T_{p_i}^k ∈ ℙ_k, s.t. l ∉ T_{p_i}^k then
5              continue;
6          end
7          else
8              remove e∈l from E and set c_e = 0, ∀e∈ℝ_k;
9              check all the active sessions d_j in G;
10             for each protection link e_j on wavelength λ_j do
11                 if the segments protected by e_j are link disjoint with l then
12                     set c_{e_j} = 0;
13                 end
14             end
15             run NPF and PPH to obtain protection trees T_{p_i}^k and T_{p_i'}^k;
16             select the T_{p_i}^k with less cost and add it to ℙ_k;
17             add e to ℝ_k, ∀e ∈ T_{p_i}^k and e ∉ ℝ_k;
18             update the segments protected by each e_l where e_l ∈ T_{p_i}^k;
19         end
20     end
21     if Cost_{ℝ_k} < Cost_{ℝ_min} then
22         Cost_{ℝ_min} = Cost_{ℝ_k};
23     end
24 end
25 return (Cost_{ℝ_min} == ∞? block : accept);
```

9-14. Then we run PPH and NPF to obtain the protection tree with the lower cost. After each protection tree is established, we update the final topology and overall cost, which is depicted by lines 15-18. By comparing three topologies, we choose the one with the minimum cost as the final survivable topologies.

At any given time, we assume the average number of sessions in the network is denoted by $|d_j|$, which can be calculated by $\lambda\mu$, in which λ is the average arrival rate of multicast sessions and μ is the average holding time of each session. To check each pure protection link of each session in d_j, it takes $O(|V|^2)$. Therefore, the total time complexity of CB_SPT is $O((|V|^2 + |V||d_j||V|^2)) = O(\lambda\mu|V|^3)$.

Algorithm 9: Wavelength-based SPT Algorithm (WB_SPT)

Input: $G, d = \{s, t_i\}$ $(1 \leq i \leq M), W_k, W_{\min}{=}\infty$
Output: accept or block?

```
1  for k = 0; k<3; k++ do
2      construct T_k^m by running kth heuristic and set W_k=0;
3      for each segment l ∈ T_m^k do
4          construct a protection tree T_{p_i}^k for each segment l, similar to line 2-14 in
           Algorithm SPT;
5      end
6      if ∃T_m^k and T_{p_i}^k then
7          set W_k = |e|, e ∈ R_k;
8      end
9      for each protection link e ∈ R_k do
10         if ∃ protection link e_j ∈ d_j and segments protected by e_j are link disjoint with the
           segments protected by e then
11             W_k−−;
12         end
13     end
14     if W_k < W_min then
15         W_min = W_k;
16     end
17 end
18 return (W_min == ∞? block : accept);
```

The second heuristic is WB_SPT shown in Algorithm 9. Instead of looking for the survivable topology with the minimum cost, we try to minimize the total traffic flow and choose each session with the minimal number of wavelengths actually reserved every time. Thus, by extending SPT, the first several steps remain the same, in which we use three different multicast algorithm to construct three primaries trees and obtain the segments for each tree. The difference from SPT starts from line 6. Line 6-8 actually summarizes the total number of links used by the combination of primary and all the protection trees. The next step is to find all the pure protection links in the survivable topology, which can be cross sharing with other pure protection links in any existing sessions. If this link is found, the corresponding link in the new session does not need to be provisioned, which can be subtracted from the total number of wavelengths. This process is described by line 9-13. We choose the final topology out of the three as the one with the minimum number of wavelengths used. The total time

complexity is the same as CB_SPT, which is $O(\lambda\mu|V|^3)$.

5.6 Numerical Results

In order to investigate the overall performance of the proposed multicast protection schemes in our study, we consider two network topologies: NSF network [102] and USNET [79]. Each link is assigned a certain cost determined by the distance between two end nodes. USNET has a greater number of nodes, links and average node degree than NSF network.

The results consist of three parts. In the first part, we calculate the average cost of provisioning a given multicast session by using SPT in both network topologies. We will compare them with the best existing heuristics, OPP_SDP, as well as the optimal solution developed in [79]. In the second part, we compare the average number of reconfigurations between SPT and OPP_SDP upon any single-link failure. In the last part, we study the performance of two extended heuristic algorithms, CB_SPT and WB_SPT, for dynamic traffic scenarios and compare them with OPP_SDP in terms of blocking probability in various traffic scenarios.

We investigate the total link cost to route one multicast session and its protection trees in this part, under the following assumptions:

1. A network scenario is defined by one source and M destinations and the source and destinations are randomly generated for each network scenario;

2. The bandwidth requirement of each multicast session is one wavelength and the links of network topologies are uncapacitated;

3. For each network scenario, we run the simulation 200 times and take the average value.

Given fixed traffic pattern, we compare the average cost achieved by SPT scheme to that obtained by with OPP_SDP algorithm as well as the optimal solution solved by formulating the problem using Integer programming, which is also proposed in [79]. Tables 5.3 and 5.4 illustrate the average cost of provisioning a multicast session obtained by the different approaches in NSF and USNET networks, respectively, in which the session size denotes the number of destinations

in a session and saving ratio reflects the cost saving ratio of heuristic SPT over OPP_SDP and is defined by $(C_{\text{OPP\_SDP}} - C_{\text{SPT}})/C_{\text{SPT}}$.

5.6.1 Single Multicast Session

Table 5.3 The comparison of average network cost of provisioning a survivable multicast session in NSF network

Session Size	2	4	6	8	10	11	13
Optimal	8835.5	12537.2	15097.2	17152.1	18984.2	19720.4	21164.9
SPT	8904	13274.1	15833.7	17871.3	19876.5	20938.9	22491.9
OPP_SDP	8922.2	13383	16262.6	18432	20572.7	21648.6	23351.6
Saving Ratio (%)	0.20	0.82	2.71	3.14	3.50	3.39	3.82

Table 5.4 The comparison of average network cost of provisioning a survivable multicast session in USNET network

Session Size	2	6	10	14	18	20	23
Optimal	10839	19696.7	25518.4	30491.4	35209.61	37461.0	40838.6
SPT	11076	19974	27212.5	32199	36607.5	39493	42761
OPP_SDP	11393	20319.5	27649.5	32830.5	37366.5	39770.6	43307.5
Saving Ratio (%)	2.86	1.73	0.92	1.96	2.07	0.70	1.28

It is clear that results produced by both SPT and OPP_SDP are close to the optimal solutions within 10% in NSF network and 15% in USNET. However, SPT produces lower total cost than OPP_SDP approach in both network topologies. The saving ratio of SPT over OPP_SDP in NSF network is between 0.2% and 4%, and the most saving ratios in USNET fluctuate between 1% and 2% with various session sizes. In NSF network, the advantage of SPT over OPP_SDP gradually increases as the session size increases, which is not the case in USNET. One of the reasons is that NSF network has a smaller average nodal degree such that finding two link disjoint paths for each pair of source and destination conducted in OPP_SDP scheme may end up with long paths. However, SPT is not affected as much since different segments may share the same protection tree. The larger the session size is, the higher the

possibility that segments will share protection with one another. However, this feature cannot be applied to OPP_SDP scheme.

In USNET, the average nodal degree is higher and the distances between different pairs of nodes do not vary as much as in the NSF network. The shortest path pair established earlier in OPP_SDP scheme may be shared by other source and destination pairs with higher probability. Therefore, the advantages of SPT scheme is not as significant as that in NSF network.

5.6.2 Average Number of Reconfigurations

We also studied the failure recovery performance in terms of average number of switch reconfigurations given any link failure in both NSF and USNET network topologies. The method of calculating the number of reconfigurations in the SPT scheme has been presented in Section 5.4. In OPP_SDP scheme, the shortest pair of paths between the source and each destination is constructed. We consider one as the primary path and another as the protection path. The combination of all the primary paths construct a primary multicast tree. We assume that when a link on the multicast tree fails, all the disrupted primary paths will be rerouted from the source to the corresponding destinations through the protection paths. Accordingly, the same reconfigurations rule described in STP can be applied here. Hence, we obtained the average number of reconfigurations of both protection schemes in NSF and USNET networks as shown in Fig. 5.6 and 5.7. Each value is obtained by taking the average over 200 independent cases for each network scenario.

It is obvious that the average number of reconfigurations increases as the session size increases in both topologies due to the fact that final topology gets larger and denser and link sharing becomes more prevalent between different source and destination pair. Therefore, the average nodal degree of the survivable multicast session gets higher and more nodes will become potential switch nodes. Thus, more nodes will actually reconfigure their switch upon a link failure. However, the increase in the number of reconfigurations under the OPP_SDP approach is much faster than SPT as the increase of the session size, because the larger number of destinations in the session results in a greater number of path pairs in the multicast topology

and one link capacity may be shared by a large number of primary paths. Therefore, one link failure will disrupt more primary paths and cause more reconfigurations. As we can see in the figures, the performance of SPT and OPP_SDP are close when there are only two destinations. However, when they provision broadcast services, the advantages of SPT over OPP_SDP reaches almost 30% in NSF network and 86% in USNET.

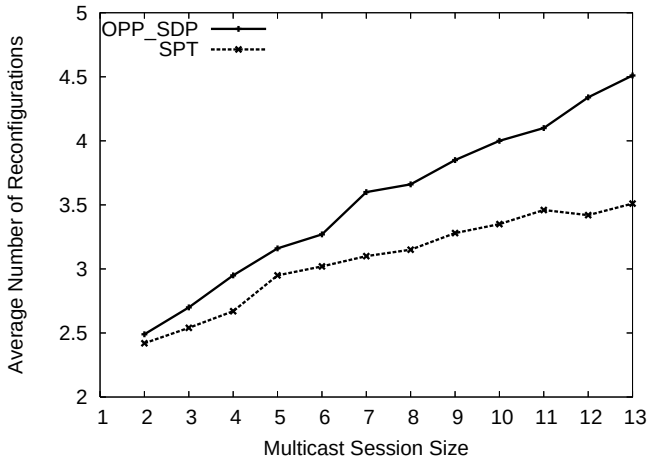


Figure 5.6 Comparison of Average Number of Reconfigurations in NSF network

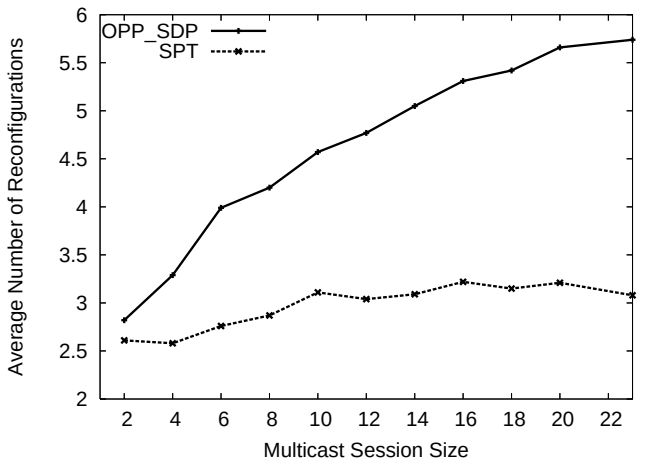


Figure 5.7 Comparison of Average Number of Reconfigurations in USNET network

However, SPT performs very well in USNET since the average number of reconfigurations grows very slowly as the session size increases. Since each protection tree is independent from

one another and also from the multicast tree, each protection tree can share a large number of links with the primary tree except the segment it protects, which means any link failure will not result in a significant change between the multicast tree and the corresponding protection, especially when the session size is very large. Therefore, only a limited number of nodes may need reconfiguration differing significantly from OPP_SDP scheme in the same scenario. In summary, SPT outperforms OPP_SDP in terms of the configuration time in all the network scenarios in our study and the advantages vary from 10% to 86%.

5.6.3 Dynamic Multicast Sessions

We also study the performance of two extended heuristic algorithms, CB_SPT and WB_SPT, and compare them with OPP_SDP for dynamic traffic scenarios. The simulation is conducted on two realistic networks, NSFNET and USNET. In each simulation run, 1000 randomly generated multicast requests are loaded to the network sequentially and the reject ratio is recorded.

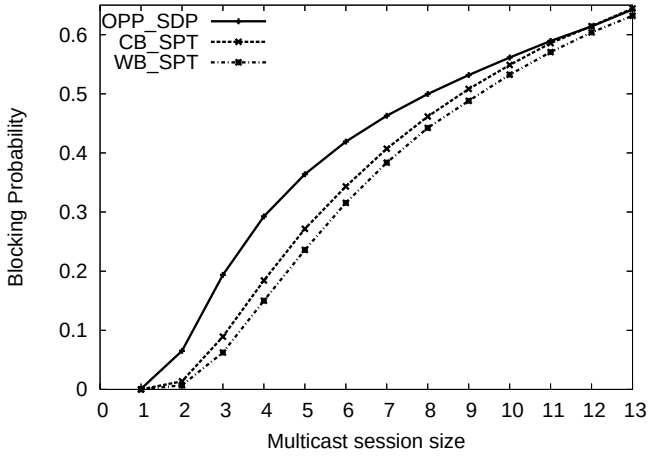


Figure 5.8 Blocking Probability of dynamic multicast sessions with Erlang=100 in NSFNET

Figure 5.8 and 5.9 show the comparison of blocking probability in NSFNET with various traffic scenarios. Each value in the figures is calculated by averaging the results of 200 independent cases at each traffic scenario. In Fig. 5.8, we simulate multicast sessions with different size, which vary from 2 to 13, but with the fixed traffic load, which is represented by 100

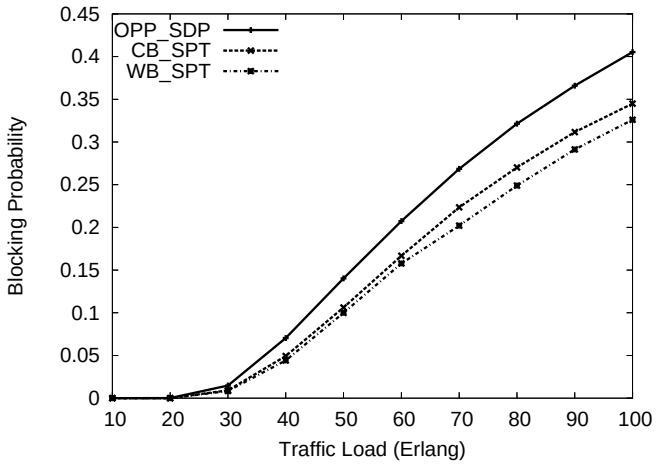


Figure 5.9 Blocking Probability of dynamic multicast sessions whose session sizes are uniformly distributed in [2, 12] in NSFNET

Erlangs. Since the traffic load in Erlangs = $\lambda\mu$, which means in a fixed time slot, the number of arrival sessions is 100 times of that of departure sessions. The blocking probability increases as the session size increases, since the larger the multicast session is, the more resources will be consumed. In Fig. 5.9, x axis represents the load in Erlangs which increases from 10 to 100 and we uniformly distribute the multicast session size between 2 and 12 for each scenario. In both figures, our proposed schemes, CB_SPT and WB_SPT, achieves lower reject ratio than OPP_SDP does, which was claimed as the most capacity efficient multicast protection scheme. Among them, WB_SPT achieves the best solution, since we try to use the minimum number of wavelengths to provision each session regardless of the cost of the session. In this case, more resources can remain for future sessions. It is worth noting that when the multicast session size reaches 13 in Fig. 5.8, the traffic pattern becomes multicasting. More than half the sessions are rejected and three schemes behave almost the same. Due to the large size of each session, the network cannot benefit enough from cross-sharing to accept more multicast sessions.

We also study the dynamic traffic scenarios in USNET network as shown in Figure 5.10 and 5.11. In Fig. 5.10, the multicast session size is uniformly distributed between 2 and 20. In Fig. 5.11, the traffic load in terms of Erlang is fixed at 100. We can observe that comparison to the OPP_SDP without cross-sharing, both CB_SPT and WB_SPT have significant advantages over

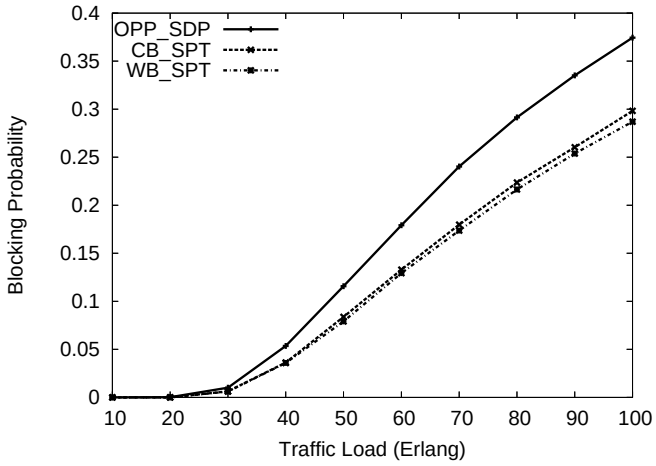


Figure 5.10 Blocking Probability of dynamic multicast sessions where session sizes are uniformly distributed in [2, 20] in USNET

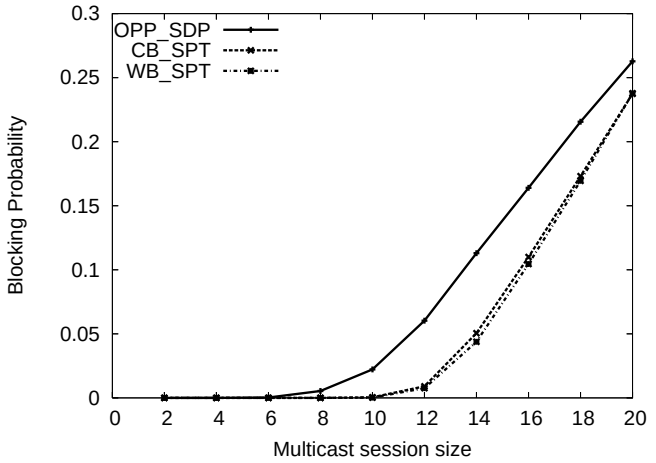


Figure 5.11 Blocking Probability of dynamic multicast sessions with Erlang=100 in USNET

OPP_SDP in terms of blocking probability. The advantage reaches 20% as the traffic load or the multicast size increases. In such a dense network, WB_SPT still behaves better than CB_SPT, but the advantage is almost invisible. By combining the results from two networks, we can observe that WB_SPT achieves the best blocking probability in all the scenarios considered, which means minimizing the total number of wavelengths for each arrival multicast session is more efficient than minimizing the total link cost of each session. In fact, by making the cost of each link in the network equal to each other, WB_SPT is equivalent of CB_SPT. This means WB_SPT can be considered as a special case of CB_SPT.

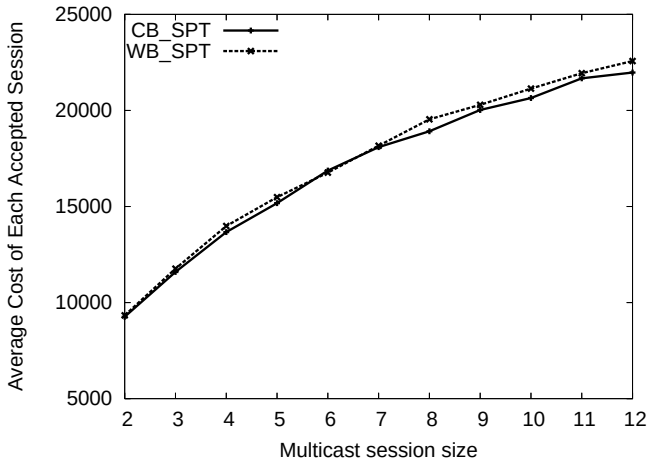


Figure 5.12 Average cost of each accepted session in NSFNET where traffic load equals 100 Erlangs and the number of wavelengths on each link equals 32

In addition, we further compare CB_SPT and WB_SPT protection schemes in terms of the average cost of each accepted multicast session. In practise, each multicast session may come from an independent user. The network operator may charge each user by the total cost of each session. Therefore, we study the average cost of each multicast session in NSFNET and USNET and results are shown in Figure 5.12 and 5.13, respectively. For each traffic scenario with a unique session size, we take the average value over 1000 independent cases. We can observe that CB_SPT achieves slightly lower cost than WB_SPT does. This is predictable because CB_SPT chooses the final topologies with the minimum cost out of three multicast provisioning schemes. However, the advantages are very small in both networks, especially

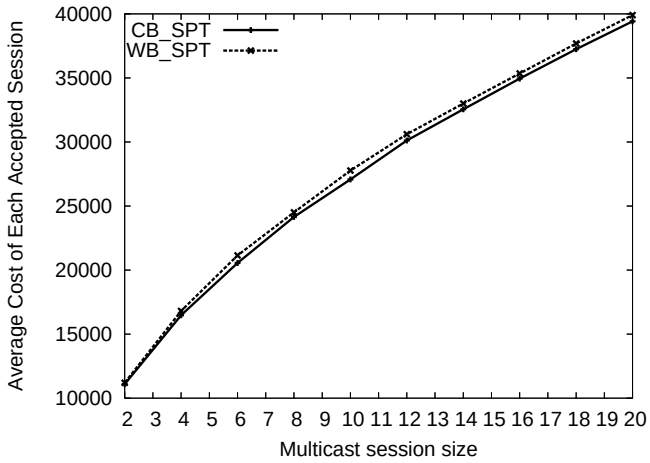


Figure 5.13 Average cost of each accepted session in USNET where traffic load equals 100 Erlangs and the number of wavelengths on each link equals 64

in USNET. Consider the overall performance of both overall blocking probability and average cost, CB_SPT and WB_SPT perform very close in relatively large network with high nodal degree such as USNET. In sparse and medium size network, such as, NSFNET, WB_SPT can achieve better capacity efficient in terms of blocking probability, but CB_SPT has a small advantage in terms of the average cost of each accepted session.

5.7 Conclusions

We studied the problem of provisioning survivable multicast sessions with protection against single link failures in a network with minimum resources, and proposed a heuristic algorithm, Segment-based Protection Tree (SPT), to provision and protect a multicast session. In the SPT scheme, three primary multicast trees are established first by three different multicast provisioning approaches, NPF, PPH and DST, respectively, and then each segment of each primary tree is protected by a multicast tree, called protection tree, which is selected out of two candidates produced by NPF and PPH, respectively. Each primary tree and its corresponding protection trees compose a survivable topology. We choose the one with minimum network cost as the final topology. By extending SPT, we also proposed two schemes, CB_SPT and

WB_SPT, to protect dynamic multicast sessions, in which we utilize the feature of self-sharing and cross-sharing to enable maximum protection capacity sharing within a multicast session as well as among different multicast sessions.

We studied the performance of SPT in terms of network cost and average number of reconfigurations. SPT uses no more than 10% extra cost over the optimal solution under all network scenarios considered and only 5% extra cost over the optimum when the session size is very small or large, such as unicast or broadcast, respectively. In terms of both cost and recovery performance, SPT achieves better than OPP_SDP, which was considered as the best capacity efficient scheme. We also studied the dynamic traffic scenarios, and the results show that our proposed schemes, CB_SPT and WB_SPT, also achieve better overall blocking probability than OPP_SDP in various network scenarios, in which WB_SPT achieves the better capacity efficiency but CB_SPT achieves lower average cost of each session.

CHAPTER 6. GENERAL CONCLUSIONS

In the thesis, we addressed a number of problems in optical networks, which fall into two areas: traffic grooming and network survivability. In the traffic grooming part, we addressed the many-to-many traffic grooming problem on unidirectional ring networks using network coding in order to reduce cost comparing to traditional traffic grooming schemes. In the network survivability part, we studied both unicast and multicast protection problems.

For many-to-many traffic grooming, we addressed static many-to-many traffic in two different unidirectional ring networks, single-hub and unhubbed and considered the total number of LTEs as the objective to be minimized. We first proved that the general many-to-many traffic grooming problem in single-hub ring networks is NP-complete. Then we considered two operational scenarios, namely, node-disjoint many-to-many groups and non-disjoint groups. In single-hub rings, we proved that if the traffic rate, r, and the group size, n, of a given group satisfy the condition: $\lceil (n-1)r/g \rceil < \lceil nr/g \rceil$, where g is the grooming factor, then applying network coding to the provisioning of low granularity traffic flow of this group will save a LTE at each group node. This condition can also apply to unhubbed rings if one-hub approach is used. We studied four different cases that consist of many-to-many traffic grooming with node disjoint groups and non-disjoint groups in both single-hub and unhubbed ring networks and addressed the complexities of the problems. We have investigated the benefits of applying network coding to many-to-many traffic grooming in all the cases and showed that using network coding will either save LTEs or total traffic flow transmitted within each group. In the case where there is a large number of many-to-many groups coexisting in the network, the total cost of network provisioning can be indeed reduced by employing network coding.

The thesis also deals with network survivability, which is an important problem both in

theory and practice. However, in the general case, this problem is also hard. In this thesis, we addressed three different survivability problems.

We first studied the unicast protection problem against double-link failure, which is the second most frequent failure scenario after single-link failures. We chose p-cycle problem due to its great performance in both capacity efficiency and failure recovery speed. In this work, we first studied the static traffic provisioning problem by formulating it as an integer program, which is the candidate's major contribution to the paper in chapter 3. And then we proposed two p-cycle-based heuristic algorithms, SPPP (Shortest Path Pair Protection) and SFPP (Short Full Path Protection), to address dynamic traffic scenarios. The results reveal that SFPP achieves better capacity efficiency than SPPP in most scenarios. However, SPPP has slightly better failure recovery performance than SFPP.

We also extended traditional p-cycles by adding protection links connected to the cycles, which we refer to as parasitic protection links (PPL). A p-cycle with PPL is therefore called a p^2-cycle. The motivation of having PPL is to extend the capability of a p-cycle to protect connections whose end nodes do not only lie on the p-cycle but also those whose end nodes can be reached by PPLs. The use of p^2-cycle to protect both static and dynamic traffic scenarios was studied. The static traffic protection problem was formulated as an integer program, and the dynamic case was addressed by proposing two heuristic algorithms, Strict Routing Protection (SRP) and Flexible Routing Protection (FRP). We studied the performance of p^2-cycles in terms of capacity efficiency and speed of recovery, and compare it to other p-cycle based schemes. We found out that p^2-cycle improve capacity efficiency with a small increase in failure recovery speed.

The last problem we addressed was the multicast protection problem. Given a multicast session, we need to provision and protect this session with minimum overall cost. We propose a new protection scheme, namely, Segment-based Protection Tree (SPT). In order to handle dynamic traffic, we extend SPT to protect dynamic multicast sessions by utilizing both self-sharing and cross-sharing such that protection resources can be shared within a multicast session as well as among different sessions. We studied the performance of SPT in terms

of network cost and the average number of reconfigurations (NOR). We also extend SPT to address dynamic traffic scenarios, in which we propose two heuristic algorithm, Cost-base SPT (CB_SPT) and Wavelength-based SPT (WB_STP). The results showed that SPT and its extensions achieves better solutions than the existing schemes in both static and dynamic traffic scenarios. Meanwhile, the failure recovery speed of SPT is significantly less than one of the most cost efficient scheme, namely, OPP_SDP. This has shown that SPT and its extension possess a good overall performance among all multicast protection schemes.

We propose to extend the work of this thesis in a number of directions. One of the straightforward extensions is to extend multicast protection schemes proposed in this thesis to protect many-to-many sessions. This work can be further extended by combining traffic grooming and survivability problems in the many-to-many context. Although both problems are hard, combining them will be a practical and valuable contribution.

In terms of survivability, the concept of differentiated survivability has been recently developed due to the fact that different users running different applications may have different requirements on the quality of protection. In this sense, our p^2-cycle scheme can be further extended to adapt to the different users requirements by balancing the capacity efficiency and protection speed. By relaxing the constraint on the length of PPL, the capacity efficiency of p^2-cycles can improve further. However this will cause more complicated traffic recovery mechanism and longer recovery speed. Based on the different protection requirements, we may flexibly control the length of PPL to provide different levels of protection.

Another area of survivability is multi-domain protection. Current optical networks are managed and operated by different telecom carriers and are composed of several independent optical domains based on diverse technologies. Up to date global network information may not always be available to provide efficient provisioning and protection for traffic requests that traverse multiple domains. Thus, traditional survivability techniques suitable for single-domain networks may not be applicable to multidomain networks directly. Therefore, how to apply the protection schemes proposed in this thesis across multiple domains is another interesting, and practical extension.

BIBLIOGRAPHY

[1] R. Ramaswami and K. N. Sivarajan, "Optical Networks: A Practical Perspective," 2nd ed. San Mateo, CA: Morgan Kaufman, Nov. 2001.

[2] K. Zhu, B. Mukherjee, "Traffic grooming in an optical WDM mesh network," *IEEE J. Select. Areas Commun.* 20(1) (2002) 122-133.

[3] D. Zhou and S. Subramanian, "Survivability in optical networks," *IEEE Networks,* 2000.

[4] Arun K. Somani, "Survivability and Traffic Grooming in WDM Optical Networks," Cambridge University Press, 2006.

[5] E. Modiano and P. J. Lin, "Traffic grooming in WDM networks", *IEEE Commun. Mag.,* vol. 39, no. 7, pp. 124-129, Jul. 2001.

[6] R. Dutta and G. N. Rouskas, "On optimal traffic grooming in WDM rings", *IEEE J. Sel. Areas Commun.,* vol. 20, no. 1, pp. 110-121, Jan. 2002.

[7] O. Grestel, R. Ramaswami, and G. Sasaki, "Cost effective traffic grooming in WDM rings", *IEEE/ACM Trans. Netw.,* vol. 8, no. 5, pp. 618-630, Oct. 2000.

[8] X.Zhang and C.Qiao, "An effective and comprehensive approach for traffic grooming and wavelength assignment in SONET/WDM rings," *IEEE/ACM Trans. Networking*, vol. 8, no.5, pp. 608-617, Oct 2000.

[9] J. Wang, W. Cho, V. Vemuri and B. Mukherjee, "Improved approaches for cost-effective traffic grooming in WDM ring networks: ILP formulations and single-hop and multihop connections," *Journal of Lightwave Technology,* Vol. 19, No. 11, pp.1645-1653, 2001.

[10] Long, L. and A. E. Kamal, "Reducing network cost of many-to-many communication in unidirectional WDM rings with network coding," *IEEE/OSA Journal of Lightwave Technology,* Vol. 27, No. 19, Oct. 2009, pp. 4209-4221.

[11] A. L. Chiu and E. H. Modiano, "Traffic grooming algorithms for reducing electronic multiplexing costs in WDM ring networks," *IEEE/OSA Journal of Lightwave Technology*, 18(1): 2-12, January 2000.

[12] J-C. Bermond and D. Coudert, "Traffic grooming in unidrirectional WDM ring networks: the all-to-all unitary case," *in Proc. 7th IFIP Working Conference on Optical Network Design and Modeling*, Feb. 2003.

[13] J. Q. Hu, "Optimal traffic grooming for WDM rings with all-to-all uniform traffic," *Journal of Optical Networks,* Vol. 1, 32-42, 2002.

[14] P. J. Wan, G.Galinescu, L.Liu and O.Frieder, "Grooming of arbitrary traffic in SONET/WDM BLSRs," *IEEE Journal Select. Areas Commun.*, Vol. 18, no.10, pp.1995-2003, Oct.2000.

[15] G. Feng, C. Siew, and T-S. Yum, "Architectural design and bandwidth demand analysis for multiparty videoconferenceing on SONET/ATM rings," *IEEE J. Select. Areas Commu.*, vol. 20, no. 8, pp.1580-1588, 2002.

[16] X. Y. Li, L. Liu, P. J. Wan, and O. Frieder, "Practical traffic grooming scheme for single-hub SONET/WDM rings," in *Journal of High Speed Networks*, Vol 11, Issue 2, pp.103-119, 2002.

[17] H. V. Madhyastha, G. V. Chowdhary, N. Srinivas, and C. S. R. Murthy, ""Grooming of multicast sessions in metropolitan WDM ring networks," *Computer Networks*, vol. 49, no. 4, pp. 561-579, 2005.

[18] Anuj Rawat, Richard La, Steven Marcus, and Mark Shayman, "Grooming Multicast Traffic in Unidirectional SONET/WDM Rings," *IEEE J. Select. Areas Commun.* 20(6) August, 2007.

[19] H. Zhu, H. Zang, K. Zhu, and B. Mukherjee, "A novel, generic graph model for traffic grooming in heterogeneous WDM mesh networks," *IEEE/ACM Trans. Netw.,* vol. 11, no. 2, pp. 285-299, Apr. 2003.

[20] J. Q. Hu, Brett leida, "Traffic Grooming, Routing, and Wavelength Assignment in Optical WDM Mesh Networks," *Proceedings of IEEE Infocom,* 2004.

[21] A. Khalil, A. Hadjiantonis, C. M. Assi, A. Shami, G. Ellinas and M. A. Ali, "Dynamic Provisioning of Low-Speed Unicast/Multicast Traffic Demands in Mesh-Based WDM Optical Networks," *IEEE/OSA Journal of Lightwave Technology*, Vol. 24, No. 2, pp. 681-693, Feb. 2006.

[22] Kamal, A. E., "Algorithms for Multicast Traffic Grooming in WDM Mesh Networks," *IEEE Communications*, Volume 44, No. 11, Nov. 2006.

[23] Ul-Mustafa, R. and A. E. Kamal, "Design and Provisioning of WDM Networks with Multicast Traffic Grooming," *IEEE Journal on Selected Areas in Communications, Part II: Optical Communications and Networking*, Vol. 24, No. 4, Apr. 2006.

[24] Ul-Mustafa, R. and A. E. Kamal, "Many-to-one Traffic Grooming with Aggregation in WDM Networks," *IEEE Journal on Selected Areas in Communications, Part II: Optical Communications and Networking*, Vol. 24, No. 8, Aug. 2006, pp. 68–81.

[25] Saleh, M. and A. E. Kamal, "Many-to-Many Traffic Grooming in WDM Networks," IEEE/OSA Journal of Optical Communications and Networking, Vol. 1, No. 5, Oct. 2009, pp. 376-391.

[26] B. Doshi et al., "Optical network design and restoration," *Bell Labs Tech. J.,* vol. 4, no. 1, pp. 58-84, 1999.

[27] C. H. Yang and S. Hasegawa, "FITNESS: Failure Immunization Technology for Network Services Survivability," *Proc. IEEE GLOBECOM 88,* Hollywood, Fla., Nov. 1988, pp. 1549-1554.

[28] K. Murakami and H.S. Kim, "Optimal Capacity and Flow Assignment for Self-Healing ATM Networks Based on Line and End-to-End Restoration," *IEEE/ACM Trans. on Networking,* Vol. 6, No. 2, 1998, pp. 207-221.

[29] S. Ramamurthy and B. Mukherjee, "Survivable WDM mesh networks. Part I-protection," *in Proceedings of IEEE INFOCOM,* vol. 2, pp. 744-751, 1999.

[30] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee, "Survivable WDM mesh network," *Journal of Lightwave Technology,* vol. 21, no. 4, pp. 870-883, 2003.

[31] P. H. Ho and H. T. Mouftah, "A framework of service guaranteed shared protection for optical networks," *IEEE Communications Magazine,* pp. 97-103, February 2002.

[32] D. Xu, L. Guo, Y. Xiong and C. Qiao, "Novel Algorithms for Shared Segment Protection," *IEEE J. on Selected Areas in Commu.,* Vol. 21, No. 8, pp. 1320-1331, Oct. 2003.

[33] J. Tapolcai, P. Ho, D. Verchere, T. Cinkler and A. Haque, "A new shared segment protection method for survivable networks with guaranteed recovery time," *IEEE Trans. Reliability,* Vol. 57, No. 2, pp. 272-282, June 2008.

[34] O. Gerstel and R. Ramaswami, "Optical layer survivability - An implementation perspective," *IEEE J. Select. Areas Commun.*, vol. 18, pp. 1885-1899, Oct. 2000.

[35] Ramasubramanian, S. "On failure dependent protection in optical grooming networks," *IEEE International Conference on Dependable Systems and Networks (DSN),* pp. 475-484. Florence, Italy (June-July 2004).

[36] Michael T. Fredrick, Pallab Datta, Arun K. Somani, "Sub-graph routing: a generalized fault-tolerant strategy for link failures in WDM optical networks," *Computer Networks,* pp. 181-198, Volume 50, Issue 2, February 2006.

[37] Taiming Feng, Long Long, Ahmed Kamal and Lu Ruan, "Two-link Failure Protection in WDM Mesh Networks with p-Cycles," accepted for publication in *Elsevier Computer Networks*, 2010.

[38] Long Long and Ahmed Kamal, "P^2-Cycles: P-Cycles with Parasitic Protection Links," *in the proceedings of IEEE International Conference on Communications (ICC)*, South Africa, 2010.

[39] W.D Grover and D. Stamatelakis, "Cycle-oriented distributed preconfiguration: Ring-like speed with mesh-like capacity for self-planning network restoration," in *Proc. IEEE ICC' 98,* 1998, pp. 537-543.

[40] W.D. Grover and D. Stamatelakis, "Bridging the ring-mesh dichotomy with p-cycles," in *Proc. of DRCN Workshop,* 2000.

[41] D.A Schupke, C.G. Gruber, and A. Autenrieth, "Optimal configuration of p-cycles in WDM networks," in *Proc. of IEEE ICC,* 2002.

[42] Bin Wu, Kwan L. yeung, and Shizhong Xu, "ILP Formulations for p-Cycle Design without Candidate Cycle Enumeration," accepted in *IEEE/ACM Transactions on Networking,* 2008. (`http://www.eee.hku.hk/research/doc/tr/TR2008001_IFDCC.pdf`)

[43] W.D. Grover and J.E. Doucette, "Advances in optical network design with p-cycles: Joint optimization and preselection of candidate p-cycles," in *Proc. of IEE LEOS Summer Topical Meeting*, 2002.

[44] G. X. Shen and W. D. Grover, "Extending the p-cycle concept to path segment protection for span and node failure recovery," *IEEE Journal on Selected Areas in Communications,* vol. 21, no. 8, pp. 1306-1319, Oct. 2003.

[45] Adil Kodian and W. D. Grover, "Failure-Independent Path-Protecting p-Cycles: Efficient and Simple Fully Preconnected Optical-Path Protection," in *IEEE Journal of Lightwave Technology,* Vol. 23, 2005, pp. 3241-3259.

[46] Christian Mauz, "Unified ILP Formulation of Protection in Mesh Networks," *7th International Conference on Telecommunication (CONTEL)*, 2003.

[47] Raghav Yadav, Rama Shankar Yadav, and Hari Mohan Singh, "A review of survivable transport networks based on p-cycles," in *International Journal of Computer Sciences and Engineering Systems*, 2007, vol. 1.

[48] Bin Wu, Kwan L. yeung, and Shizhong Xu, "Ilp formulation for p-cycle construction based on flow conservation," in *proceedings of the IEEE GLOBECOM*, 2007.

[49] Markopoulou A., Iannaccone G., Bhattacharyya S., Chen-Nee Chuah, and Diot C., "Characterization of failures in an ip backbone," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2004, vol. 4, pp. 2307–2317.

[50] W. He, M. Sridharan and A. K. Somani, "Capacity Optimization for Surviving Double-Link Failures in Mesh-Restorable Optical Networks," Photonic Network Communications, 9:1, 99-111, 2005.

[51] Hongsik Choi, Suresh Subramaniam, and Hyeong-Ah Choi, "Double-link failure recovery in wdm optical networks," in *proccedings of IEEE INFOCOM*, 2002.

[52] D.A Schupke, "Multiple failure survivability in wdm networks with p-cycles," in *Proceedings of the International Symposium on Circuits and Systems*, 2003.

[53] H. Choi, S. Subramaniam, H.-A. Choi, "Loopback recovery from double-link failures in optical mesh networks," in *IEEE/ACM TRANSACTIONS ON NETWORKING,* Vol. 12, 2004, pp. 1119–1130.

[54] Hongxia Wang and H. T. Mouftah, "P-cycles in multi-failure network survivability," in *Proceedings of International Conference on Transparent Optical Networks*, 2005.

[55] A. Kodian, W. D. Grover, "Multiple-quality of protection classes including dual-failure survivable services in p-cycle networks," in *proceedings of the Broadnets,* 2005, pp. 231–240.

[56] A. E. Kamal, "1+n protection against multiple faults in mesh networks," in *proceedings of the IEEE International Conference on Communications (ICC)*, 2007.

[57] Ramesh Bhandari, "Survivable networks, algorithms for diverse routing," 1999.

[58] Taiming Feng, Lu Ruan, and Wensheng Zhang, "Intelligent p-cycle protection for multicast sessions in wdm networks," in *proceedings of the IEEE International Conference on Communications (ICC)*, 2008.

[59] Asthana R. and Singh Y.N., "Second phase reconfiguration of restored path for removal of loop back in p-cycle protection," in *Communications Letters, IEEE*, 2007, vol. 11, pp. 201–203.

[60] P. Arijs, B. V. Caenegem, P. Demeester, and P. Lagasse, "Design of ring and mesh based wdm transport networks," *Optical Networks Magazine,* vol. 1, no. 3, pp. 27-41, 2000.

[61] Dahai Xu, Y. Xiong and C. Qiao, "Novel algorithms for shared-segment protection," *IEEE Journal of Selected Areas on Communications,* v21. p1320-1331, 2003.

[62] Janos Tapolcai and et al. "A New Shared Segment Protection Method for Survivable Networks with Guaranteed Recovery Time," *IEEE Transactions on Reliability,* Vol. 57, pp. 272-282, 2008.

[63] D. Stamatelakis and W. D. Grover, "Theoretical underpinnings for the efficiency of restorable networks using pre-configured cycles *p*-cycles," *IEEE Trans. Commu.,* vol. 48, no.8, pp.1262-1265, 2000.

[64] Wayne D. Grover, "Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking," *Prentice Hall PTR,* Upper Saddle River, New Jersey, 2004.

[65] D. A. Schupke, "An ILP for optimal p-cycle selection without cycle enumeration," *8th Conference on Optical Network Design and Modelling, ONDM,* 2004.

[66] Kamal, A. E., "1+N Network Protection for Mesh Networks: Network Coding-Based Protection using p-Cycles", *IEEE/ACM Transactions on Networking,* Vol. 18, No. 1, Feb. 2010, pp. 67–80.

[67] T. Chow, F. Chudak, and A. Ffrench, "Fast optical layer mesh protection using pre-cross-connected trails," *IEEE/ACM Transactions on Networking*, vol. 12, no.3, pp.539-548, 2004

[68] D. Stamatelakis and W. D. Grover, "Network restorability design using pre-configured trees, cycles, and mixture of pattern types," TRLabs Technical Report, Tech. Rep. TR-1999-05, 1999.

[69] Samir Sebbah and Brigitte Jaumard, "A Resilient Transparent Optical Network Design with a Pre-Configured Extended-Tree Scheme," *in the Proceedings of ICC',* 2009.

[70] Wensheng He and Arun K. Somani, "Comparison of Protection Mechanisms: Capacity Efficiency and Recovery Time," *in the Proceedings of IEEE ICC',* 2007.

[71] D. A. Schupke, M. Jger, R. Hlsermann, "Comparison of resilience mechanisms for dynamic services in intelligent optical networks," *Proc. of DRCN workshop,* pp. 106-113, Oct. 2003.

[72] Wensheng He, Jing Fang, and Arun K. Somani, "A p-cycle based survivable design for dynamic traffic in wdm networks," *in Proceeding of IEEE Globecom,* pp. 1869-1873, 2005.

[73] Gangxiang Shen and Wayne D. Grover, "Design and performance of protected working capacity envelopes based on p-cycles for dynamic provisioning of survivable services," *Journal of Optical Networking,* Vol. 4, Issue 7, pp. 361-390, 2005.

[74] Bhandari R. "Optimal diverse routing in telecommunication fiber networks," *Proceedings of IEEE INFOCOM,* Toronto, Ontario, Canada, vol.3, June, 1994, pp. 1498-1508.

[75] Long Long and Ahmed Kamal, "Tree-based Protection of Multicast Services in WDM Mesh Networks," *in the proceedings of IEEE Globecom*, Nov. 2009.

[76] J. He, Gary Chan and Danny H.K. Tsang, "Multicasting in WDM networks," *IEEE Commun. Surveys and tutorials*, Dec. 2002.

[77] M. Medard, S. G. Finn, R. A. Barry, and R. G. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge redundant graphs," *IEEE/ACM Trans. Netw.*, vol. 7, no. 5, pp. 641-652, Oct. 1999.

[78] L. H. Sahasrabuddhe and B. Mukherjee, "Light trees: Optical multicasting for improved performance in wavelength routed networks," *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 67-73, Feb. 1999.

[79] N. K. Singhal, L. H. Sahasrabuddhe, and B. Mukherjee, "Provisioning of survivable multicast sessions against single link failures in optical WDM mesh networks," *J. Lightwave Technol*, vol. 21, pp. 2587-2594, 2003.

[80] N. Singhal and B. Mukherjee, "Protecting multicast sessions in WDM optical mesh network," *IEEE/OSA J. Lightwave Technol,*, vol. 21, no. 4, pp. 884-892, April 2003.

[81] N. K. Singhal, C. Ou, and B. Mukherjee, "Cross-sharing vs. self-sharing trees for protecting multicast sessions in mesh networks," *Computer Networks,*, vol. 50, no. 2, pp. 200-206, 2006.

[82] H. Luo, L. Li, H. Yu and S. Wang, "Achieving shared protection for dynamic multicast sessions in survivable mesh WDM networks," *IEEE J. on Sel. Area in Commun.*, Vol. 25, No. 9, Dec. 2007.

[83] F. Zhang, W. Zhong, and Y. Jin, "Optimizations of p-Cycle-based protection of optical multicast sessions," *J. Lightwave Technol*, vol. 26, No. 19, Oct. 2008.

[84] R. Koetter, M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Trans. on Netw.*, Vol. 11, No. 5, Oct. 2003.

[85] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory,*, vol. 46, pp. 1204-1216, July 2000.

[86] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Info. Theory,* vol. 49, No. 2, pp.371-381, 2003.

[87] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inf. Theory,* vol. 51, no. 6, pp. 1973-1982, 2005.

[88] T. Ho, M. Medard, R. Koetter, D.R. Karger, M. Effros, J. Shi and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inform. Theory,* vol. 52, no.10, pp. 4413-4430, 2006.

[89] T.Ho, B. Leong, Y. Chang, Y. Wen and R. Koetter, "Network monitoring in multicast networks using network coding," *in International Symposium on Information Theory (ISIT)* 2005.

[90] Al-Kofahi, O. and A. E. Kamal, "Network Coding-Based Protection of Many-to-One Wireless Flows," *IEEE J. Select. Areas Commun. special issue on Network Coding in Wireless Commun.,* Vol. 27, No. 5, June 2009, pp. 797-813.

[91] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution," *Proceedings of IEEE Infocom,* 2005.

[92] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard and J. Crowcroft, "XORs in the Air: Practical Wireless Network Coding," *in Proceedings of SIGCOMM,* 2006.

[93] M. Zhang, L. Wang and P. Ye, "All optical XOR logic gates: technologies and experiment demonstrations," *IEEE Communications magazine,* vol. 43, no.5, pp.s19-s24, May 2005.

[94] Eric D. Manley, Jitender S. Deogun and Lisong Xu, "Network coding for optical layer multicast," *Proc. Broadnets,* 2008.

[95] E.G. Coffman, M.R. Garey, D.S. Johnson, "Approximation algorithms for bin packing - A survey. In: Approximation Algorithms for NP-Hard Problems," *PWS Publishing Company*, Boston, pp 46-93, 1997.

[96] J.M.V de Carvalho, "Exact solution of bin packing problems using column generation and branch and bound," *Annals of Operations Research*, Springer, Volume 86, No. 0, January, 1999.

[97] S. L. Hakimi, "Steiners problem in graphs and its implications," *Networks*, vol. 1, no. 2, pp. 113-133, 1971.

[98] H. Takahashi and A. Matsuyama, "An Approximate Solution for the Steiner Problem in Graphs," *Mathematica Japonica 6* (1980), 573-577.

[99] L. Kou, G. Markowsky and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica 15* (1981), pp. 141-145.

[100] T. H. Corman, C. E. Leiserson, and R. L. Rivest, "Introduction to Algorithms," 2nd ed. Cambridge, MA: MIT Press, 2001.

[101] S. Thorpe, G. Edwards, and D. Stevenson, "Using Just-in-Time to Enable Optical Networking for Grids", *Proc. IEEE Broadnets Workshop*, Oct. 2004.

[102] S. Baroni, "Routing and wavelength allocation in WDM optical networks," PhD Thesis, University College London, May 1998, pp. 118.