University of Alberta

DESIGN AND PERFORMANCE OF PROTECTED WORKING CAPACITY ENVELOPES BASED ON *P*-CYCLES: AN ALTERNATIVE FRAMEWORK FOR DYNAMIC SURVIVABLE NETWORK SERVICE PROVISIONING

by

Gangxiang Shen 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirement for degree of Doctor of Philosophy

Department of Electrical and Computer Engineering

Edmonton, Alberta

Spring 2006

Your file Votre référence
ISBN: 0-494-14040-2
Our file Notre référence
ISBN: 0-494-14040-2

To

My parents, my wife and son, and my sister

ABSTRACT

Network survivability is becoming a more and more critical issue for transport networks. On the other hand, automation of network service provisioning is beginning to be another key issue for future transport network design and operation. As an integrated paradigm, Shared Backup Path Protection (SBPP) dynamic survivable service provisioning, which aims to support the above two critical aspects, i.e., service survivability and provisioning automation, is currently the *de facto* dynamic survivable service provisioning approach. However, its nature of determining protection relationship online for each service and complicated spare capacity sharing makes the provisioning too complicated to strictly limit its scalability. More effort is therefore required to further improve its scalability. Alternatively, other approaches should be developed.

Protected Working Capacity Envelope (PWCE) is a novel dynamic survivable service provisioning approach, which is promising enough to show the advantage over the counterpart approach SBPP in the aspects of control simplicity, restoration speed, and blocking performance. This thesis focuses on the detailed implementation and performance evaluation of this concept, of which the most important technical aspects include how to design an envelope, how to realize the concept based on a practical control technique, and how to tackle the issue of traffic load uncertainty.

Specifically, we realize the PWCE concept in the context of span-protecting p-cycle networks. We develop various volume-maximization PWCE models, which ensure the best spare capacity efficiency. We also use the GMPLS technique to realize the control system for PWCE service provisioning. Also, the performance of the approach is

evaluated in terms of control overhead and blocking probability in comparison with the counterpart approach SBPP. Moreover, to enable PWCE to adapt traffic load variation, a concept termed Adaptive PWCE (APWCE) is proposed and investigated in a network under uncertain traffic loads. In summary, the thesis demonstrates an attractive dynamic survivable service-provisioning approach that shows advantages over the *de facto* approach SBPP in aspects such as operational simplicity, blocking performance, and restoration speed.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to many individuals and organizations.

First, I would like to thank Prof. Grover for his considerable guidance and support throughout my entire PhD program. Besides this exciting PhD research topic, Dr. Grover taught me a lot on how to conduct a professional and high-quality research. All these would be life-long beneficial to my future career growth. Also, his high passion and dedication to research always impress me. He has set a great example for my future research career. Prof. Grover, I want to tell you that wherever I go, I am always proud of being one of your students.

I am also grateful to a number of current and past members of Network Systems Research Group, Dr. John Doucette, Dr. Matthieu Clouqueur, Dr. Dion Leung, Anthony Sack, Govindkrishna Kaigala, Peter Giese, Adil Kodian, Grace Shi and Dimitri Baloucov, for creating a friendly and simulating research environment in TRLabs. Also, I am indebted to Linda Richens, Luke Chong and Rhoda Hayes for your kindness and personal support throughout my entire research program.

I am also deeply thankful to the members of my thesis supervisory committee, Prof. Ray DeCorby, Prof. Ivan J. Fair, Prof. Jose Nelson Amaral and my external examiner, Prof. Jaumard, for giving me many valuable comments, corrections and clarifications to my thesis. I especially appreciate the effort made by Prof. Jaumard to fly from Montreal to Edmonton for my defence although she encountered very bad weather and day-long flight delays.

The financial support provided by Izaak Walton Killam Memorial Scholarship, iCore Graduate Student Scholarship and TRLabs Graduate Student Scholarship is highly appreciated.

Finally, I must thank my parents, my wife and son, and my sister for your constant encouragement and support throughout my entire research program. Especially, my mother, I want to say you are the greatest in my world.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

APS	Automatic Protection Switching
APWCE	Adaptive Protected Working Capacity Envelope
ATM	Asynchronous Transfer Mode
AWG	Array Wavelength Grating
BLSR	Bidirectional Line-Switched Ring
CBR	Constraint-Based Routing
CSPF	Constrained Shortest Path First
CWDM	Coarse Wavelength Division Multiplexing
DWDM	Dense Wavelength Division Multiplexing
FDL	Fiber Delay Line
FDM	Frequency Division Multiplexing
FF	First Fit
FIR	Full Information Restoration
FSC	Fiber Switching Capable
GMPLS	Generalized Multi-Protocol Label Switching
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
JCA	Joint Capacity Assignment
KSP	K-Shortest Path
L2SC	Layer2 Switching Capable
LDP	Label Distribution Protocol
LER	Label Egress Router
LIR	Label Ingress Router
LL	Lease Load

LMP	Link Management Protocol
LP	Lightpath
LSA	Link State Advertisement
LSP	Label Switching Paths
LSR	Label Switching Router
MPLS	Multi-Protocol Label Switching
OBS	Optical Burst Switching
OEO	Optical-Electronic-Optical
ONS	Optical Network Simulator
OPS	Optical Packet Switching
OSPF-TE	Open Shortest Path First-Traffic Engineering
OTDM	Optical Time Division Multiplexing
OXC	Optical Cross-Connect
PIR	Partial Information Restoration
PNNI	Private Network-to-Network Interface
PR	Path Restoration
PSC	Packet Switching Capable
PSR	Path Segment Restoration
PWCE	Protected Working Capacity Envelope
PWCT	Protected Working Capacity Tunnel
QoP	Quality of Protection
QoS	Quality of Service
RSVP-TE	Resource Reservation Protocol - Traffic Engineering
SBPP	Shared Backup Path Protection
SBSP	Shared Backup Segment Protection

SCA	Spare Capacity Assignment
SDH	Synchronous Digital Hierarchy
SLA	Service Level Agreement
SR	Span Restoration
SRLG	Shared Risk Link Group
SONET	Synchronous Optical Network
TDR	Time-Dependent Routing
TE	Traffic Engineering
TNM	Telecommunication Network Manage
TS	Time Slot
UNI	User-to-Network Interface
UPSR	Unidirectional Path-Switched Ring
VC	Virtual Circuit
VP	Virtual Path
WSC	Wavelength Switching Capable

CHAPTER 1
INTRODUCTION

1.1. BACKGROUND AND MOTIVATION

The unprecedented increase of data traffic has overtaken voice traffic to become the primary application in transport networks. Besides the traditional IP-based applications such as emails, web browsing, FTP, bandwidth-intensive services ranging from Voice over IP (VoIP) to Video on Demand (VoD), IPTV, Internet games, Peer-to-Peer (P2P) services, and reliability-sensitive services such as Internet banking, online stock systems, online credit card payment applications, and so on, are becoming pervasive. Meanwhile, the recent advances of the Dense Wavelength Division Multiplexing (DWDM) technique have created opportunities for transmitting multiple terabits per second of data traffic in a single fiber. Moreover, the rapid development of optically switching techniques such as MEMS switches has enabled the new generation of wavelength cross-connects to switch thousands of wavelengths simultaneously. With such a tremendous growth of data traffic and intense usage of fiber-optics telecommunications, any fiber cut or switching node failure will yield a huge loss of data traffic, which in turn may affect millions of telephone calls or Internet sessions, unless network survivability has been incorporated into network design and operation [WuTh92] [Gro04b].

In parallel with the IP-based traffic growth and the upgrading of fiber-optic transport network capacity, network service provisioning is turning from traditional static provisioning mode to future dynamic provisioning mode. Arbitrarily rapid provisioning and provisioning on demand are becoming two of the most important features for future network service provisioning, which will impose challenges to network service providers because they need to respond to each network service request in real time and thus will have far less time for demand or traffic-load forecasting and networks pre-planning than previously. They will have to rely more on the new generation of network control systems, which are expected to be more intelligent, flexible, adaptable, and automotic, so as to effectively compensate for the slow human operations in the traditional forecasting and pre-planning procedure.

As an integrated paradigm, Shared Backup Path Protection (SBPP) dynamic survivable service provisioning, which aims to support the above two critical aspects, i.e., service survivability and provisioning automation, is currently the most promising and *de facto* dynamic survivable service provisioning approach [KiKo00] [KaSa94]. However, its online nature to determine protection-working relationship in real time and feature of complicated spare capacity sharing are fatal issues strictly limiting its scalability of service provisioning. The current SBPP provisioning approach must be further improved. Alternatively, other approaches should be developed.

As an alternative, Protected Working Capacity Envelope (PWCE) is a novel and promising dynamic survivable service provisioning technique, which shows the advantage over the conventional SBPP technique in the aspects of simpler control, faster restoration speed, and comparable or even better blocking performance. The concept of PWCE was first proposed in [Grov04a] [Grov04b], in which the fundamental operational principles and potential strength of PWCE over SBPP were conceptually highlighted. However, the details on how to implement the technique have not yet be fully explored, the most important of which include how to design an envelope, how to realize the concept based on a practical control technique such as Generalized Multiple Protocol Label Switching (GMPLS), and how to tackle traffic load uncertainty issue. Also, the quantitative performance evaluation has not yet been conducted for PWCE in comparison with the conventional SBPP provisioning approach.

1.2. OBJECTIVE

The objective of this research is to implement the PWCE concept in the context of span-protecting *p*-cycle networks. Although the implementation is general enough to support other types of survivability techniques that are eligible to the PWCE concept, the span-protecting *p*-cycle technique is chosen owing to its features of ring-like switching speed and mesh-like spare capacity efficiency. To assure the best spare capacity sharing efficiency, we will develop various PWCE volume-maximization models under different capacity budgets. We will also use the GMPLS technique to realize the control system for PWCE service provisioning. The aspects of routing component, signaling component, and the concrete structure of network state database, will be specifically examined. Also,

the performance of the paradigm in terms of control overhead and blocking probability will be evaluated in comparison with the *de facto* SBPP-based provisioning approach. Based on the performance comparisons, the advantages of PWCE will be verified. Finally, to have a thorough understanding of the behaviors and characteristics of PWCE, other issues that are closely related will also be explored, involving the impacts of capacity modularity and network capacity uniformness, and the adaptiveness of PWCE to traffic load variation. In particular, for the last issue, a concept termed Adaptive PWCE (APWCE) will be investigated in a network with the feature of demand or traffic load uncertainty. Based on this study, more advantages of PWCE are expected over SBPP such that PWCE can resist the influence of the traffic load uncertainty to maintain a sound network performance any time under a highly-fluctuating traffic load environment. In summary, this thesis will explore and demonstrate an attractive dynamic survivable service provisioning paradigm that shows advantages over the *de facto* approach SBPP in the aspects of simpler operation and faster restoration speed, but comparable or even better blocking performance.

1.3. THESIS ORGANIZATION

Chapter 2 gives a brief introduction to transport networks. The definition of a transport network is given in detail, and its functions and evolution are discussed. Three types of switching techniques—circuit, packet, and burst switching—are described and compared. Finally, various network protocols and transmission techniques are briefly described, including MPLS, wavelength-routed WDM network, Optical Burst Switching (OBS), and Optical Packet Switching (OPS).

Chapter 3 provides background information on network survivability, which covers concepts and terms relevant to network survivability, introductions to a wide range of survivability techniques, and multiple-layer network survivability. The concepts and terms include network failure types, concept of Shared Risk Link Group (SRLG), route diversity, restoration, working and spare capacities, spare capacity redundancy, restorability, etc. The survivability techniques involve the simplest 1+1, 1:1, and M:N techniques, conventional ring-based ones, and more advanced mesh-survivable networks, which include span restorable mesh networks, *p*-cycles, path restoration, path-segment

restoration, etc. Finally, for the multiple-layer survivable techniques, two possible strategies, i.e., top-down and bottom-up, and the concept of common pool survivability, are discussed.

Chapter 4 gives a detailed introduction to *p*-cycle techniques. Two types of *p*-cycles, namely span-protecting *p*-cycles and flow *p*-cycles, are introduced. A literature survey on the *p*-cycle technique reviews the current state-of-art of the technique. Also, optimal design models for span-protecting *p*-cycles are presented.

Chapter 5 introduces the SBPP-based survivable service provisioning. The fundamental of the GMPLS technique is first described. Next, the relevant terms and concepts of GMPLS ranging from traffic engineering to bidirectional LSP, IP addressing, unnumbered link, link bundling and others, are introduced. Based on the above fundamental knowledge, general architecture and operation of GMPLS are further reviewed and its routing component, signaling component, and Link Management Protocol (LMP), are discussed in detail. Following this, the concept of SBPP is reviewed and the literature on SBPP is surveyed. Finally, the GMPLS technique is applied to control the SBBP-based service-provisioning network. The detailed operation steps and the related network state database and connection database are presented. Also, the routing algorithm on how to select working and protection routes for each survivable service is given.

Chapter 6 introduces the forcer concept that is key to the volume-maximization models when designing PWCE envelopes. In the context of span-restorable mesh network, forcer examples are first given to illustrate the forcer concept, which is then extended for span-protecting *p*-cycle networks. To identify all the forcers in a network, two conventional forcer identification methods are reviewed, followed by two novel identification methods with better efficiency. Forcer analyses are conducted on some sample networks to discover forcers and how many extra working capacity units can be raised before non-forcers become forcers as well. Finally, taking advantage of the forcer phenomenon, two potential applications are discussed, which involve "forcer clipping" and "non-forcer filling." Particularly, the non-forcer filling application is the theoretical foundation of our following volume-maximization envelope designs.

Chapter 7 studies p-cycle-based PWCE. Specifically, the concept, design, and basic performance evaluation are examined. The GMPLS technique is applied to control and operate the PWCE-based network. The network state database, Link State Advertisement (LSA) flooding process, and signaling process are described. In addition, to fully take advantage of the spare capacity, a wide range of volume-maximization models are proposed to design PWCE envelopes. The forcer theory discussed in the previous chapter is used and various spare and total capacity budgets are considered when developing these models. Based on the volume-maximized envelopes, the performance of PWCE is then evaluated in comparison with the SBPP-based survivable service provisioning method. The terms of control overhead, network state memory, and blocking probability are studied respectively. The results show that PWCE needs a much lower control overhead and less network state memory, but achieves a comparable or even better blocking performance than SBPP.

Chapter 8 investigates other concerns that are related to PWCE so as to thoroughly understand the behaviors and properties of the technique. The issues include the impacts of capacity uniformness and modularity. Capacity uniformness means there is an equal total installed capacity on each span within the network, and capacity modularity means the deployed network capacity is modularized to be some discrete capacities such as OC-16, OC-48, etc.

Chapter 9 studies the performance of PWCE under the circumstance of demand or traffic load uncertainty. A new concept termed Adaptive PWCE (APWCE) is proposed and explored. A variety of relevant aspects are specifically considered, including the strategies that trigger PWCE reconfigurations, the types of capacity patterns for PWCE reconfiguration, and the types of total span capacities. The performance of APWCE is evaluated and compared with those of fixed PWCE and SBPP under several artificial load-uncertainty experimental environments.

Chapter 10 concludes the thesis and specifies future research directions.

CHAPTER 2
BACKGROUND ON TRANSPORT NETWORKS

2.1. DEFINITION OF TRANSPORT NETWORKS

A transport network is a system that transmits information from one location to another via intermediate switching devices such as routers and switches. In terms of hardware, a transport network comprises a group of communication entities interconnected by transmission media or even without any media (e.g., wireless and free space optical communications). These communication entities can be as simple as an optical transceiver, or as complicated as a switching node like a multiple protocol label switch. In the wired transport network, the transmission media can be either metal cables or fibers, while in the case of free-space optical and RF wireless communication systems, essentially no transmission media are needed.

Transport networks offer two major functions. First, they offer the functions of *multiplexing* or *traffic grooming*, which involve synchronizing frames, detecting data errors, encapsulating data packets from the application layer into link-layer frames, and hierarchically multiplexing lower-bit-rate data flows into higher-bit-rate data tributaries. For example, in the traditional IP/ATM/SONET networks, IP packets are first segmented and encapsulated into ATM cells, and the latter are then encapsulated into low bit-rate SONET frames such as OC-1. These low bit-rate tributaries are then further multiplexed into higher levels of tributaries, such as OC-3/4, then OC-12, and ultimately OC-48/192. Second, transport networks possess the flexibility of building various "virtual" topologies in different network layers. In appearance, these virtual topologies can be very different from the bottom physical topology. Each virtual topology is made up of different types of virtual links, which are specifically determined by the layer of which the virtual topology is constructed. For example, a virtual topology in the wavelength-routed WDM layer consists of a set of end-to-end lightpaths, a virtual topology in the SONET layer is made up of a set of end-to-end SONET virtual containers, and the virtual topologies in the MPLS and IP tunnel layers are made up of MPLS Label Switching Paths (LSP) and IP Tunnels, respectively. The ability to create various virtual topologies brings the transport

networks two important benefits. First, it enables the transport networks to improve network resource utilization by balancing traffic loads. Specifically, by pre-planning and reconfiguring virtual topologies, the transport networks can control the traffic flow distribution and achieve an optimal network throughput. The second benefit is that it helps the transport networks to enhance their reliability by applying a wide range of survivability techniques. The virtual topology construction allows the networks to establish an alternate connection to bypass a failure or to be failure-disjoint from the primary connection, thus enabling the transport networks to survive any network failure or to be designed with a guaranteed reliability.

2.2. LAYERED TRANSPORT NETWORK AND ITS EVOLUTION

Transport networks have a layered architecture, made up of a wide range of protocol stacks, as shown in Figure 2.1. There are four typical architectures. Figure 2.1(a) shows the most common but also the most complex protocol stack, which subsumes almost all the major transport protocols in the telecommunications area. From the top to the bottom, IP layer is a client layer interfacing with MPLS/ATM layer, and the latter is in turn a client layer interfacing with SONET/SDH layer, and then WDM and fiber layers. From the bottom to the top, the WDM layer functions to provide lightpath-based virtual topologies for the SONET/SDH layer, and the latter offers communication tunnels for the MPLS/ATM layer, and then for the IP layer. Figure 2.1(b) shows a more concise network-layered architecture after the intermediate ATM layer is eliminated [MaAn98], while the architecture in Figure 2.1(c) is even "thinner" to have the SONET/SDH layer eliminated as well to transport IP traffic directly over WDM networks [RaLu04]. Ultimately, the architecture in Figure 2.1(d) is envisioned as the future transport network-layered architecture to carry IP traffic directly over super-fast OPS/OTDM techniques. OPS and OTDM stand for Optical Packet Switching and Optical Time Division Multiplexing respectively, which are anticipated to be the next generation super-fast all-optical transmission and switching techniques. We will provide more detail on these techniques later.

From the most complex layered architecture to the simplest, Figure 2.1 also illustrates a history and/or roadmap for transport networks, which aims to save costs, simplify systems, and increase capacity.

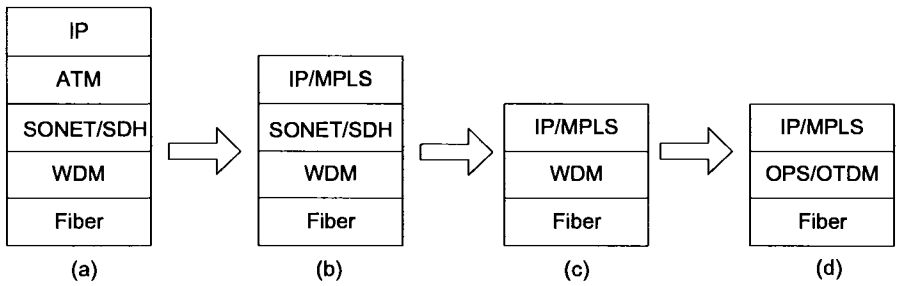


Figure 2.1. The Evolution of Transport Network Layered Architecture.

The aspects of cost-saving and system-simplifying are mainly inferred from the reduced number of network layers. An N-layer network contains N sets of hardware facilities and control software with one for each layer. The hardware normally includes switching or routing equipments such as IP routers, optical switches, etc., and the control software involves the software packages functioning to control the hardware equipments. Figure 2.1(a) represents the most complex network architecture, in which it subsumes the hardware facilities of IP routers, ATM/MPLS switches, SONET/SDH switches, optical switches, etc., and correspondingly the required software packages include IP routing package, the package for the Label Distribution Protocol (LDP) and OSPF-TE protocols (MPLS network) [DaRe00], the package for the User-to-Network Interface (UNI) and Private Network-to-Network Interface (PNNI) protocols (ATM network), the control package based on the Telecommunication Network Manage (TNM) protocol (SDH/SONET network), and probably some vendor-specific private control packages specifically for certain network layers such as the wavelength-routed WDM layer. In contrast, a network with a *compressed* layered architecture requires much fewer hardware devices and software packages. In the simplest architecture as shown in Figure 2.1(d), only IP routers and optical switches and two sets of corresponding control software are required, while all the other hardware and software in the ATM and SONET/SDH layers are saved. A simple layered network architecture is undoubtedly far cheaper than a complex one.

The compression of network-layered architecture is also beneficial to upgrading network capacity. This is so because the elimination of redundant network layers can remove the layers that may potentially bottleneck the network capacity. Each transport network layer has its own capacity up-limit, which is constrained by factors such as switch/router interface and exchange capacities. Under the multi-layer circumstance as shown in Figure 2.1, the overall network capacity is limited by the capacity of each contained layer. The ATM layer is a good example that had ever been a capacity bottleneck for the traditional transport networks. Due to the extremely short length of an ATM cell and the lack of super-fast data-processing chips, the interface speed of ATM switches had ever been constrained at 622Mb/s (i.e., OC-12) for a long time. Even though more advanced ATM switches were developed later to support 2.5Gb/s interfaces, it is likely that there are no ATM switches supporting 10Gb/s interfaces. In contrast, even in the middle of the 1990's, SONET/SDH switches that interface with the ATM layer could support 2.5Gb/s and 10Gb/s interface speeds. Thus, between the ATM and SONET/SDH layers, there had been a capacity mismatch. The ATM layer as an intermediate layer formed a capacity bottleneck between the IP and SONET/SDH layers. To streamline the network architecture, a more concise architecture, IP directly over SONET/SDH[1], was therefore developed later as shown in Figure 2.1(c). Now IP core routers commonly support 10Gb/s or even 40Gb/s interface speed [Rear02], which perfectly match the SONET/SDH interface speeds. In future, another critical capacity bottleneck for the transport networks will be the up-limit of electronic processing speed. This bottleneck can severely constrain the electronic TDM transmission capacity from being beyond 40Gb/s. Fortunately, new switching techniques such as Optical Burst Switching (OBS), Optical Packet Switching (OPS) [GaRe98], Optical Time Division Multiplexing (OTDM) [Omah95]—which switch network traffic in all optical domain and thus is immune to the up-limit of electronic processing speed—have been investigated and are coming into being. Thus, the future transport networks are foreseen to evolve from IP over SDH/SONET over WDM to IP directly over OPS/OTDM.

[1] Of course, another important reason that people favor IP over SONET/SDH is that the ATM layer needs a 10% luxury "cell-header tax," which "wastes" a large volume of network capacity.

2.3. THREE TYPES OF SWITCHING TECHNOLOGIES: CIRCUIT, PACKET AND BURST

Based on the methods of establishing communication connections, the switching technologies for communication networks can be broadly divided into three categories: *circuit switching*, *packet switching*, and *burst switching*.

2.3.1. CIRCUIT SWITCHING

Circuit switching establishes a confirmed connection before a formal communication begins. The traditional telephony network is a typical example of this technology. To make a phone call, one person needs to ring another person first, and only after the second person answers the call, which means a confirmed connection has been formed, can they begin to talk. In addition, besides the traditional telephony networks, the circuit-based switching technology also exists in communication networks such as ATM, MPLS, SONET/SDH, and wavelength-routed WDM networks. In the ATM network, Virtual Circuit (VC) and Virtual Path (VP) are two typical examples of circuit. In the MPLS network, Label Switching Path (LSP) with traffic engineering support is another example [DaRe00]. Ultimately, Time Slot (TS) and Lightpath (LP) [ChGa92] are two types of existing circuits in the SONET/SDH and WDM networks respectively. In particular, the circuits in the packet or cell-oriented networks such as the ATM and MPLS networks are often called *virtual circuits*. Here "virtual" means "not independently existing." In these networks, despite the virtual circuits established in some confirmed ways, they do not independently exist like telephony circuits, times slots, or lightpaths. Rather, they are multiplexed in a common capacity pipe and provided with only guaranteed bandwidth and fixed routes for cells or packets to forward in sequence. Thus, they would probably interfere with one another. For instance, in an MPLS network the overload of an LSP may degrade the Quality of Service (QoS) performances of other collateral LSPs (although a certain level of QoS may still be guaranteed for each of the LSPs). However, in a SONET/SDH or wavelength-route optical network, this type of situations will not occur, in that time slots or lightpaths are independently carrying services, and cannot interfere with each other.

2.3.2. PACKET SWITCHING

As opposed to circuit switching, packet switching does not need to establish any confirmed connection before communication. It forwards packets hop-by-hop based on packet header information and packet forwarding tables at each intermediate router or switch node. This forwarding process is analogous to the process in virtual circuit switching, yet they differ in the aspects of (1) packet transmission route and (2) forwarding sequence. Virtual circuit switching ensures all the packets that belong to the same communication session traverse an identical route in order, whereas packet switching allows the packets to travel via diverse routes, and probably the arriving sequence of the packets at the destination can differ from the departure sequence at the source. Furthermore, virtual circuit switching is capable of offering services with guaranteed QoS. In contrast, it is extremely difficult for packet switching to make assured QoS. Specifically, in a packet-switching network, each application session utilizes network resources in a best-effort fashion; they are not assured with any QoS concerns such as precise time delay, sufficient bandwidth, fixed packet loss ratio and others.

The traditional pure IP network is a typical example of packet switching. IP routers forward packets based on IP addresses and IP routing tables. The packets that belong to the same session are allowed for traversing diverse routes; thus, it is possible that an earlier departure IP packet may reach the destination later than those leaving subsequently. Also, the traditional IP network offers services in the best-effort fashion; no promises are made to applications on how much bandwidth is reserved and how long it will take for a packet to reach the destination.

2.3.3. BURST SWITCHING

Finally, burst switching can be seen as a compromise between packet and circuit switching [YoJe97][2]. When burst traffic arrives, the source node first sends a control packet to establish a connection. However, in contrast to circuit switching, the network can send the burst traffic immediately after the control packet and via the same route, not requiring a confirmation message back from the destination indicating that the connection has been set up successfully. Depending on the operational policy, the burst traffic can be

[2] Note that here the definition of burst switching is not identical to the one provided in [Amst89].

a single packet or a large batch of data. The volume of burst that triggers a new session of burst transmission is predefined by the switching system. Whenever the volume of accumulated data reaches the predefined burst size, a new burst session will be initiated to transmit the burst. The major functions of the control packet are to reserve network resources, e.g., bandwidth, and set up intermediate switch status before the burst traffic arrives. Between the control packet and the burst traffic, an appropriate time offset, which is normally much shorter than the time that is taken by the circuit switching network to establish a confirmed connection, is preserved such that the control packet can properly reserve network resources and set up switch status before the burst arrives. The process of establishing a burst-switching connection can result in two consequences. The control packet may succeed in establishing the connection and the burst traffic is transmitted successfully as well, or the effort of establishing a connection fails due to lack of network resources. In the latter case, the burst traffic must be dropped and retransmitted later. Two possible retransmission strategies can be applied: one to develop a special retransmission mechanism within the burst-switching layer, and the other to take advantage of the retransmission mechanism within the client layer, e.g., the retransmission function in the TCP layer [SiGo03].

As a compromise, burst switching is expected to perform between packet switching and circuit switching. Compared to packet switching, it has the advantage of stronger control over network resource reservation, time-delay and delay jitter guarantees, and so on, whereas compared to circuit switching, it is not yet a switching mechanism able to reliably guarantee service's QoS. Burst switching is a mechanism more suitable to transmit the services with burst feature such as IP traffic. Currently, *Optical Burst Switching* (OBS), which switches burst traffic all in the optical domain, is one of the most popular applications of this type of switching technique [YoJe97]. We will give more detail on OBS in the following section.

2.4. Protocols and Transmission Technologies in Layered Transport Networks

In this section, we will provide a brief overview of a wide range of protocols existing in the layered transport networks. As shown in Figure 2.1, IP is a protocol within the

network layer. MPLS, which lies between the IP and SONET layers, spans both the network layer and the data-link layer, and is therefore often called a 2.5-layer protocol. An MPLS network *without* traffic engineering support forwards packets based on label headers and label-forwarding tables, whose fundamental principle is analogous to the pure IP network. However, with the support of traffic engineering, MPLS is more like a data-link layer protocol. Some data-link layer functions, such as explicit-route and QoS-guaranteed LSP establishment, virtual topology construction, and so on, can also be supported. Likewise, SONET/SDH supports frame synchronization, error detection, and others. It thus belongs to the category of data-link layer protocol. Below SDH/SONET, the wavelength-routed WDM network is often perceived as a 1.5-layer technique because it not only interfaces the fiber transmission medium, but also supports the functions of lightpath establishment and virtual topology construction. Finally, Optical Burst Switching (OBS) [YoJe97] and Optical Packet Switching (OPS) [GaRe98] can be seen as multi-layer techniques, for they support not only all the functions of the data-link layer, but also the functions of the network and physical layers. Specifically, the function of packet forwarding in the optical domain, which is similar to that in the IP layer, is a function belonging to the network layer, while the capabilities of wavelength conversion and using Fiber Delay Lines (FDLs) as optical buffers are the functions in the physical transmission layer. Consequently, OBS and OPS are usually classified as 1.5-2.5 layer techniques.

2.4.1. Multiple Protocol Label Switching (MPLS)

MPLS is a new switching technique developed to improve the IP packet forwarding efficiency and eliminate the high *cell-header tax* in the ATM layer [DaRe00] [RoVi01]. In the IP networks, it is well known that looking up a routing table is time-consuming, which constrains the packet-forwarding efficiency. To improve the forwarding efficiency, MPLS employs a label-switching mechanism to avoid the complicated routing table searching process. This mechanism is very similar to the cell switching in the ATM networks. Each MPLS packet contains a short label and user data, and the label is used to forward the packet in the same way as the cell switching in the ATM networks. However, in contrast to the ATM networks, the MPLS networks are more efficient in network switching overhead. Specifically, in each 53-byte ATM cell, there is a five-byte header,

which equivalently consumes around 10% network bandwidth, whereas in each MPLS packet—often several hundred bytes long—there is only a four-byte header, which obviously requires much less network bandwidth.

Figure 2.2 illustrates a typical example of an MPLS network, which contains three types of MPLS switches, namely Label Ingress Router (LIR), Label Switching Router (LSR), and Label Egress Router (LER). LIR and LER are two types of edge switches. LIR functions to encapsulate an IP packet into an MPLS packet by adding an MPLS label before each IP packet, and LER offers an inverse function to recover an IP packet from an MPLS packet by stripping the label header from the MPLS packet. LSR is a type of central switch that is deployed in the middle of the MPLS network and functions to forward/relay MPLS packets.

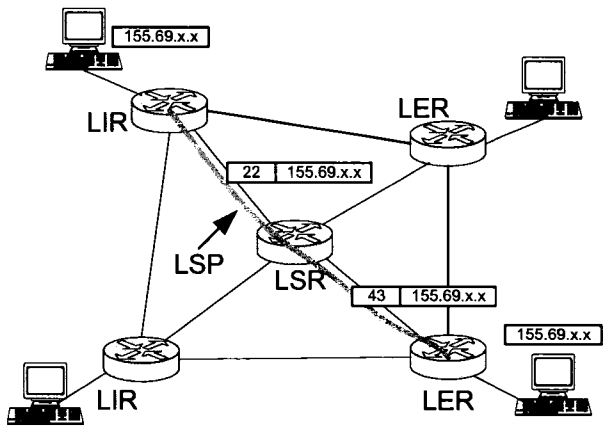


Figure 2.2. An Example of a MPLS Network.

MPLS forwards packets based on a label-switching mechanism, which is different from the traditional IP forwarding. In the example, an LSP is established between a pair of LIR and LER switching nodes. To forward packets over the LSP, IP packets are first encapsulated into MPLS packets at the LIR node; an IP packet addressed 155.69.x.x is encapsulated into an MPLS packet by assigning a label 22. This MPLS packet is then sent to the intermediate LSR, where the packet is switched and its label is swapped. The packet label is swapped from 22 to 43, and then the packet is switched to the output interface that leads to the LER node. Finally, when the MPLS packet reaches the LER node, the original IP packet is recovered by stripping the label header from the MPLS

packet; label 43 is stripped from the MPLS packet, and the original IP packet is recovered as shown in the figure.

As a general switching mechanism, MPLS can also be used to support various types of others networks. It can support Frame Relay and ATM networks by regarding the virtual circuit ID of Frame Relay and path ID of ATM as special label cases [RoVi01]. It can also support circuit-based switching networks such as SONET/SDH and wavelength-routed WDM networks by regarding time-slot ID and wavelength ID as special label cases. These extensions have been standardized by IETF as Generalized Multi-Protocol Label Switching (GMPLS) [Mann03]. We will provide more detail on GMPLS in Chapter 5.

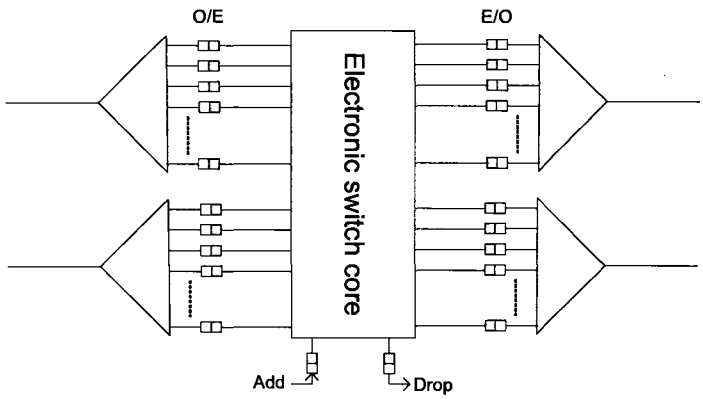
2.4.2. Wavelength-Routed Optical Networks

Wavelength-routed WDM networks, also termed WDM networks, are a type of high-speed transport network, in which Wavelength Division Multiplexing (WDM) centered in two low-loss low-dispersion spectrum windows (i.e., 1300nm and 1550nm) is applied to simultaneously transmit multiple distinct wavelengths in a single fiber. Depending on the spacing between two neighboring wavelengths, the WDM transmission systems can be broadly divided into Dense WDM (DWDM) and Coarse WDM (CWDM). As standardized, DWDM normally has a 0.8nm wavelength spacing and CWDM is coarser to have a 20nm wavelength spacing. DWDM generally transmits more wavelengths than CWDM.

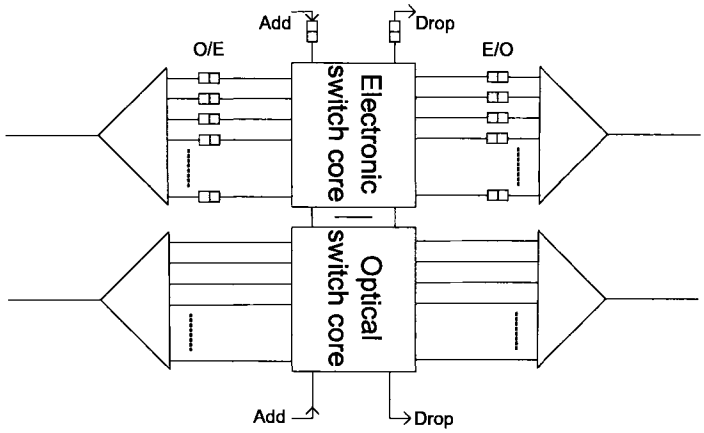
Technically, WDM is a multiplexing technique analogous to Frequency Division Multiplexing (FDM) in wireless communications. A WDM transport network is usually made up of following hardware components: (1) optical lasers and receivers, (2) wavelength multiplexers and demultiplexers, (3) optical amplifiers, and (4) optical switches. Optical lasers and receivers function to convert electronic signals into optical signals at transmitters and optical signals into electronic signals at receivers, respectively. Wavelength multiplexers and demultiplexers function to combine and split wavelengths in a fiber. The technologies such as Array Wavelength Grating (AWG) [Uets04] and Fiber Grating [Gile97] are widely used to fabricate this type of component. Optical amplifiers function to compensate optical signal loss after the signal traverses a long

distance. Currently EDFA is one of the most popular amplification techniques [MeRe87], although other advanced techniques such as Raman Amplifiers [Isla02] can be used if a wider amplification spectrum is required. Finally, optical switches function to switch lightpaths. There are two techniques available to fulfill the function, which involve Optical-Electronic-Optical (OEO) switching [KoAc96] and all-optical switching [ChGa92] respectively.

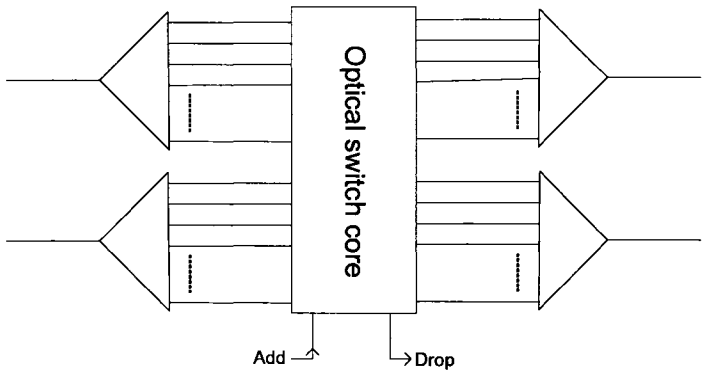
The OEO technique switches lightpaths all in the electronic domain, whose switching procedure contains three steps. It first converts all the incoming lightpaths from optical signals to electronic signals. Next it switches the converted electronic signals to output interfaces accordingly. Finally, it converts the switched electronic signals back to optical signals by modulating lasers at the output interfaces. The OEO technique shows the strengths of maturity, ease of channel monitoring, signal regeneration, wavelength conversion, and capability of traffic grooming. However, it suffers from low scalability, high electricity power consumption, low switching capacity, and electronic processing bottleneck. In contrast, the all-optical technique switches lightpaths all in the optical domain, with advantages over the OEO counterpart in the aspects of bit-rate transparency, protocol transparency, no speed bottleneck of electronic processing, and low electricity power consumption. However, the penalties are the difficulty in channel performance monitoring (due to the signal transparency) and low flexibility in traffic grooming. Particularly, to overcome the difficulty in channel monitoring, techniques such as optical power tapping [GaBl98] can be used. Nonetheless, the immaturity and other challenges such as accuracy prevent the techniques from being widely used.



(a) Opaque OXC Architecture



(b) Translucent OXC Architecture



(c) Transparent OXC Architecture

Figure 2.3. Three Types of OXC Node Architectures: (a) Opaque OXC Architecture, (b) Translucent OXC Architecture, and (c) Transparent OXC Architecture.

Based on the above two types of switching techniques, three types of switch nodes—also termed Optical Cross-Connects (OXC)—can be constructed as shown in Figure 2.3. In each of the architectures, there are pairs of "triangles" that represent wavelength demultiplexers and multiplexers. As indicated, they function to split and combine wavelengths from input interface(s) to output interface(s). Figure 2.3(a)

illustrates the architecture of an OEO OXC, in which optical signals are first converted into electronic signals, then switched by the electronic switch core, and finally converted back to optical signals. Because all the switching processes occur in the electronic domain, this type of OXC is also termed *opaque* OXC. In contrast, Figure 2.3(c) depicts an all-optical switch node. Similarly, because the switching process occurs all in the optical domain without any OEO conversion process, this type of OXC is also termed *transparent* OXC. Finally, Figure 2.3(b) displays a compromised architecture of the previous two extremes, termed *translucent* OXC [RaDa99a]. The architecture consists of an all-optical switch core and an electronic switch core. Each of the cores is generally smaller than it would be in a corresponding all-optical or all-electronic switch node. In the node, some lightpaths are switched in the optical domain and others in the electronic domain. Also, the electronic switch core can provide the functions of signal regeneration and (implicitly) wavelength conversion to lightpaths that need it.

For the transparent OXC, a variety of techniques can be used to fabricate all-optical switches, which range from MEMS to bubble [HeRe00] as well as solutions based on thermal, electrical, mechanical effects, and others. Among them, MEMS optical switches are one of the most scalable and promising solutions. Depending on the mechanisms applied, MEMS switches can be further subdivided into 1D, 2D, and 3D MEMS switches [YeLa01] [DoFa02]. These switches show their individual advantages and disadvantages in the aspects of cost, flexibility, scalability, and performance. Interested readers are referred to [MaKu03], [YeLa01] and [DoFa02] for more detail.

In addition, depending on wavelength conversion capability, OXC can be divided into three categories, namely *without wavelength conversion*, *partial wavelength conversion* [LeLi93] [YaLa96] [SuAz98], and *full wavelength conversion*. An OXC node without wavelength conversion capacity requires that the wavelength *continuity* be retained whenever a lightpath transits the node. In contrast, an OXC node with full wavelength conversion capability allows for changing the wavelength when a lightpath transits the node. As a compromise, partial wavelength conversion means that a limited number of wavelength converters are equipped at each node [LeLi93], or each converter is allowed to convert only within a limited range of spectrum [YaLa96]. A lightpath traversing an OXC node can lead to either consequence: retaining the wavelength continuity or shifting

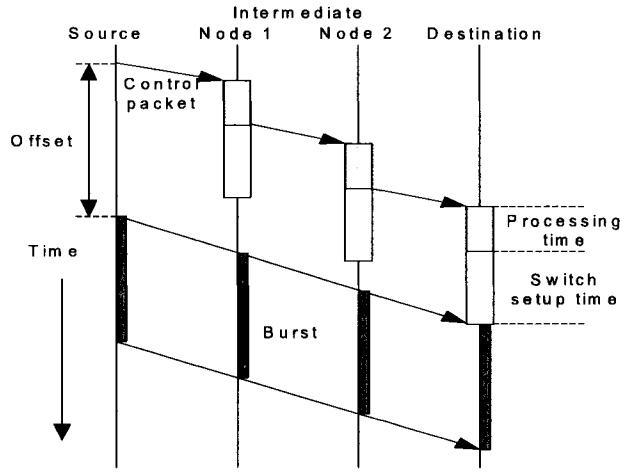
to another wavelength. The impacts of wavelength conversion on the optical network design and performance have been extensively investigated [LeLi93] [YaLa96] [SuAz98]. It has been concluded that wavelength conversion indeed shows impacts on network service provisioning performance, but rather than full wavelength conversion, partial wavelength conversion is sufficient to improve the performance to be as good as that of full wavelength conversion. In our study, we will assume that each OXC node is equipped with the full wavelength conversion capability in that the focus of the study is on the survivable network design and dynamic service provisioning. Nonetheless, we believe that this research is also applicable to the network with partial wavelength conversion.

Finally, the WDM optical networks need to consider signal regeneration. Due to the non-ideality of a fiber transmission system, an optical signal cannot travel an infinitely long distance. Factors such as fiber dispersions, non-linear effects, optical amplifier noise, and optical switch crosstalk can degrade the quality of the optical signal and make it unrecognizable after traversing a certain distance [RaDa99b]. To assure the optical signal is correctly demodulated at the receiver, signal regeneration is therefore necessary. Although it is possible to regenerate optical signals all in the optical domain [Sart01], currently OEO regeneration is the most popular and fledged technique used for the optical transmission systems. Viewing the high cost of OEO regeneration, we always want to control the number of OEO regenerations in the networks as small as possible. Translucent optical networks are an efficient method of taking advantage of OEO switch nodes sparsely deployed in a backbone transport network, which have advantage over the other alternative types of networks in cost-saving and performance improvement. For more detail on translucent optical networks, interested readers may refer to references [RaDa99a], [ShGr02] and [ShGr04a].

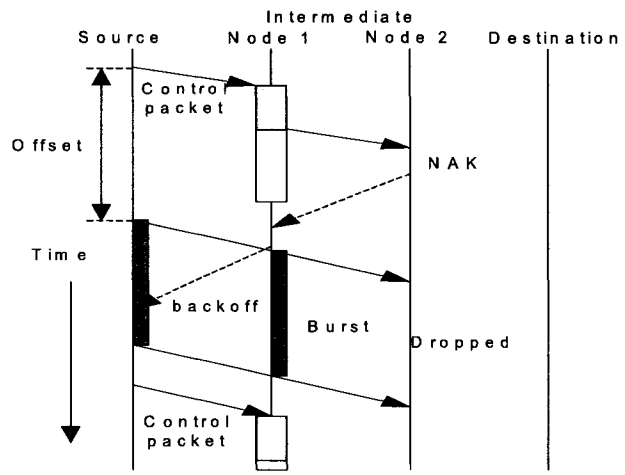
2.4.3. Optical Burst Switching (OBS)

Optical burst switching belongs to the category of burst switching [YoJe97]. The basic assumption of the technique is that data traffic is bursty. That is, the traffic arrives in a burst manner; between two neighboring bursts, there is little or no traffic. As a special application, optical burst switching possesses all features of burst switching. Specifically,

its signaling mechanism belongs to the category of one-session reservation. A control packet is first sent to make a network resource reservation, and then without a confirmation back from the destination indicating that the connection has been set up, the burst traffic is directly forwarded over the same route traversed by the control packet. Figure 2.4 illustrates the one-session reservation process showing (a) a successful reservation and (b) a failed reservation.



(a) Success



(b) Failure

Figure 2.4. OBS One-Session Reservation Process.

Figure 2.4(a) shows an example that the control packet succeeds in establishing a connection, and the burst traffic is successfully forwarded as well. Specifically, a control packet is first sent to the destination node, which makes resource reservations and sets up the switch status at the intermediate nodes. After a certain time offset, the burst traffic is

sent following the same route traversed by the control packet. Between the control packet and the burst traffic, an appropriate time offset is preserved, of which the duration is to ensure that the required resources have been reserved and the switch status has been configured when the burst traffic reaches. In the example, the offset time has typically been predefined to be the time it takes for the control packet to reach the destination node. Nevertheless, it is also legitimate to apply more efficient strategies such as JET, JIT, etc. [YoJe97] [BaRo02].

The control packet may fail in the reservations due to lack of resources, policy issues, etc. In this case, the burst traffic must be dropped at some intermediate node on the route (albeit it might have passed several hops). As shown in Figure 2.4 (b), the burst traffic is dropped at node 2 because the control packet fails to make a reservation at the node. We see that an NAK message is sent back to the source node to notify this. The NAK message travels in the reverse direction along the route, which was previously traversed by the control packet. Once it reaches the source node, the latter will apply some strategy: either to retransmit the burst traffic following same process as described before, or transfer the retransmission responsibility to the upper layer such as the TCP layer [SiGo03].

OBS can be perceived as a transitional technique from the wavelength-routed optical network to the future pure Optical Packet Switching (OPS) network. To date, this technique still suffers from the difficulties in the aspects of QoS guarantee and hardware components. First, the OBS switching mechanism cannot completely avoid the loss of burst traffic. Thus, the technique cannot make a reliable and accurate QoS promise on the performances for upper-layer application services in terms of time delay, delay jitter, packet loss ratio, etc. Therefore, it cannot function as a common data transport platform to simultaneously transmit various heterogeneous application services ranging from best-effort services to QoS-sensitive services. Second, there are still no mature and inexpensive techniques to support super-fast optical switches and optical buffers. Super-fast optical switches function to switch OBS burst traffic all in the optical domain. Among all the existing techniques, apparently only the semiconductor-based integrated optical switch is a viable solution. However, its prohibitive cost and immaturity prevents it from being widely used. Likewise, optical buffers function to temporarily store burst

traffic if a control packet fails to reserve network resources, in order to minimize the loss of burst traffic [YoJe97]. Currently Fiber Delay Lines (FDLs) are the unique solution for optical buffers. However, their poor scalability and prohibitive cost makes one doubt whether it is an efficient and practical way to use them for optical signal storage.

2.4.4. Optical Packet Switching (OPS)

Optical Packet Switching (OPS) is the most advanced switching technique for future wired transport networks [GaRe98]. Although it has the same fundamental switching mechanism as the electronic packet switching, the advantage of OPS concerns its super-high switching capacity. OPS switches signals all in the optical domain, which eliminates the bottleneck of electronic processing. Consequently, it is possible for this technique to switch up to terabit per second data traffic.

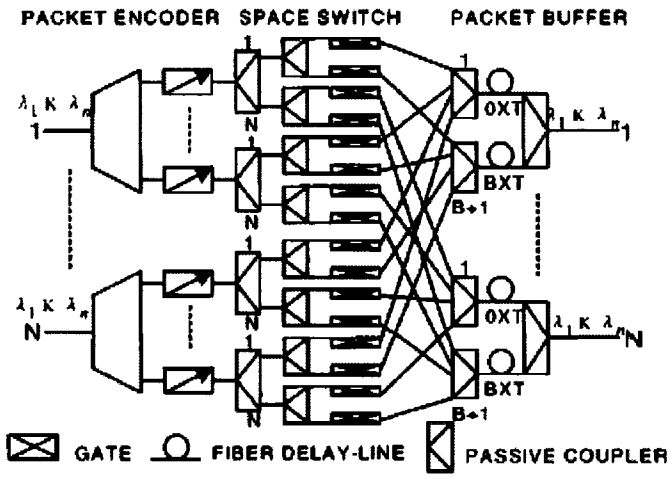


Figure 2.5. Architecture of Optical Packet Switch (OPS) (Adapted from [ShBo01a]).

Figure 2.5 illustrates a typical OPS architecture [DaHa98] [ShBo01a], comprised of three stages: (1) a demultiplexer per inlet fiber, followed by wavelength converters (WC) that converts signals from one wavelength to another, (2) an optical switch constructed by passive couplers and gate switches, and (3) output buffers realized by Fiber Delay Lines (FDLs). The function of demultiplexers is to split wavelengths in the input fiber. Following this, each split wavelength is allowed to be converted to any wavelength between λ_1 and λ_n. The intermediate switch stage is a wide-sense nonblocking one that can switch any input packet to any output buffer. The final output buffer stage functions

to store packets in order to avoid packet collisions. FDLs delay packets in a discrete fashion with a maximum of B time slots. A total of n wavelengths are allowed to transmit in each FDL. Therefore, in sum the output buffer corresponding to each output interface has up to nB units of storage space. Packet loss will occur when the whole output buffer of an interface is used up, if a new packet arrives and should be switched to the interface.

Depending on whether or not the switched packets are slotted, OPS can be broadly divided into two categories [ShMu00], namely slotted OPS and unslotted OPS. Architecturally, the slotted OPS is more complicated in nodal structure than the unslotted OPS to require an extra slot-alignment stage, which consists of a set of FDLs. The slot-alignment stages function to synchronize all the incoming packet slots by delaying earlier-arriving ones to be in line with those that arrive later. Although costly, the slot-alignment stage can greatly reduce the chances of slot collision, thereby helping the slotted OPS achieve a better throughput than the unslotted OPS.

Similar to OBS, the most challenging issue for OPS is the immaturity of hardware components. Particularly, all-optical wavelength conversion is unfledged. Also, using FDLs as output buffers is only a temporary makeshift. Finally the scalability and reliability of the OPS nodal architecture is still a big concern.

CHAPTER 3
BACKGROUND ON NETWORK SURVIVABILITY

As discussed in Chapter 1, network failures such as fiber cut and cross-connect fault can be catastrophic unless rapid restoration of services is an integrated part of network design and operation. This chapter provides general background information on network survivability. We will first introduce the concepts and terms related to network survivability. That will be followed by discussion of various survivability techniques. Finally, the strategies on multi-layer protection and restoration are addressed.

3.1. RELEVANT CONCEPTS AND TERMS

3.1.1. DEFINING NETWORK FAILURES

Natural disasters such as fire, power outages, earthquake, etc., and man-made operational mistakes, can give rise to network failures. Network failure displays in the fashion of span failure, node failure, Shared Risk Link Group (SRLG) failure [RaLu04], or even the failure of the whole network control system [LiYa02]. Among them, span failure is considered the most common and frequent one, and is often caused by fiber cable cut due to unintended or malicious excavations, shark biting, etc., or by inline component malfunctions due to various reasons such as the failure of an optical amplifier. In contrast, node failure is likely less frequent than span failure, because nodes are comprised of facilities housed in robust buildings and have high internal redundancy at the equipment-design level. Thus, the probability of a node failure is probably 100 to 1000 times lower than that of a span failure. However, the impact of node failure on the transport networks is often far more severe. Node failure can completely isolate a region from communicating with others and cause all the data that is relayed by the node to be totally lost. IP router failure is a typical example of node failure. Because a great many software manipulations and configurations are vulnerable to bugs and operational mistakes, IP router may incur node failures from time to time [ChSt03]. Finally, although rare, network failure can also occur in the fashion of network control system failure. Like IP routers, a network control system contains many software packages and requires many human manipulations, and is thus vulnerable to software bugs and operation errors as

well. The impact of the network control system failure is usually more disastrous, and it can partially or completely disable the whole transport network.

3.1.2. Shared Risk Link Group (SRLG)

SRLG is a concept specifically used to define those common failure risks in the transport networks [RaLu04]. For example, a bridge that is traversed by fiber cables can be perceived as a SRLG, as the collapse of the bridge may sever all the cables. Like the transport networks, SRLG is a layered concept. A span or link in a server layer can be perceived as a SRLG of its client layer. For example, a fiber link in the fiber layer can be seen as a SRLG of the wavelength-routed layer, because in each fiber there are generally multiple wavelengths being multiplexed. Similarly, a lightpath (i.e., wavelength) in the wavelength-routed layer can be considered a SRLG of the MPLS layer, as on a single lightpath there can be multiple LSPs multiplexed and transmitted. A SRLG can also span multiple network layers. A fiber link can also be seen as a SRLG of the MPLS layer, because in each fiber there are often thousands of LSPs multiplexed and transmitted. Compared to the conventional span failure protection, designs for SRLG failure protection are often challenging and costly. This is due to the difficulty of obtaining an accurate SRLG distribution map [SeYa02]. Even given a complete and accurate SRLG distribution map, the designs for SRLG-failure protection are far more expensive than the designs simply for span-failure protection [DoGr02]. However, in this thesis, in order to dedicate our effort to concept, design, and operation of *p*-cycle-based PWCE, we have considered a simple single-span failure only, but this does not affect the generality of the approaches herein to study SRLG failure.

3.1.3. Route Diversity: Node, SRLG, and Span Diversity

To recover network failures, a pair of working and protection paths that keep certain diversity on their routes must be simultaneously established between a node pair. Depending on restoration requirements, route diversities are divided into span disjointness, node disjointness, and generally SRLG disjointness. Span disjointness means the working and protection paths do not share a common span. Node disjointness requires the two paths not traversing any common node. Similarly, SRLG disjointness means that the two paths must not transverse a common SRLG. When discussing SBBP-

based service provisioning in Chapter 5, we will provide more detailed illustrative examples for these three types of diversities.

3.1.4. Restoration Time

Restoration time (or restoration speed) is defined as the time elapsed between the moment at which a failure occurs and the moment at which traffic is restored. It consists of the times taken for failure detection, failure notification, switching-over, and protection-path establishment. Restoration time often functions as an important criterion to evaluate a survivability technique; a short restoration time is often preferred for the network restoration.

3.1.5. Working and Spare Capacities, and Capacity Costs

Working capacity is defined as the total capacity required for the working paths to serve all the demands. Protection capacity is defined as the total capacity required for the protection paths to offer protection for the working capacity. To plan working capacity, a wide range of routing strategies, such as single shortest path routing, K-Shortest Path (KSP) routing [DuGr94], etc., can be applied. To plan protection capacity, depending on the level of network reliability, one of the three types of route diversities can be ensured, and the detailed planning strategies can be as advanced as Integer Linear Programming (ILP) optimizations or simply based on some heuristic algorithms. Mathematically, the working and protection capacities are defined as follows:

Total working capacity: $\sum_{j \in \boldsymbol{S}} w_j$

Total spare capacity: $\sum_{j \in \boldsymbol{S}} s_j$

where $\boldsymbol{S}$ denotes the set of spans in the network. w_j and s_j represent the number of working and protection capacity units on span j, respectively.

If the unit capacity cost on each span is different, then other two definitions, termed *working capacity cost* and *spare capacity cost*, can be made as well, expressed as follows:

Total working capacity cost: $\sum_{j \in \boldsymbol{S}} C_j \cdot w_j$

Total spare capacity cost: $\sum_{j \in S} C_j \cdot s_j$

where C_j denotes the unit capacity cost on span j. Depending on network types, the span unit capacity cost can be computed in various ways. For a long-haul transport network, in which fibers cost dominates the overall network cost, the unit capacity cost of each span is considered approximately proportional to the distance of the span. In contrast, for a metro-area network, with a much shorter distance between two neighboring nodes, it is necessary to also take the node cost into account when determining C_j. Specifically, the total cost of each node can be evenly distributed onto each capacity unit that is incident to the node. The total unit capacity cost on each span is thus measured as the sum of fiber cost plus the evenly distributed portions of node cost and inline optical amplifier cost.

3.1.6. Spare Capacity Redundancy

Network spare capacity redundancy, which represents network capacity efficiency, is defined as the ratio of the total spare capacity to the total working capacity, or if the network capacity cost is considered instead, the ratio of the total spare capacity cost to the total working capacity cost. For survivable network designs, spare capacity redundancy is often considered an important criterion to compare the performances of different survivability techniques. An efficient survivability technique is often expected to have a low network spare capacity redundancy.

3.1.7. Restorability

Restorability can be defined in two ways: (1) the types of failure that a survivability technique can handle—span, node, or SRLG failure, and (2) the percentage of demands that can be restored after a failure occurs. The second definition is also often called *restoration*, e.g., 100% restoration. Mathematically, the percentage of restoration can be expressed as one of the following formats:

Non-weighted restorability: $\sum_{f \in F_i} d_f^{\mathrm{r}} \Big/ \sum_{f \in F_i} d_f^{\mathrm{a}}$, or

Hop-length-weighted restorability: $\sum_{f \in F_i} d_f^{\mathrm{r}} \cdot h_f \Big/ \sum_{f \in F_i} d_f^{\mathrm{a}} \cdot h_f$

where f denotes a flow ID used to index the flows in the affected flow set F_i upon span failure i; d_f^{a} and d_f^{r} denote the numbers of affected demand units and restored demand units of flow f respectively; h_f defines the hop length of flow f. The non-weighted restorability purely defines the percentage of restored demands, while the hop-length-weighted restorability takes the factor of hop length into account when computing the restorability.

3.2. SURVIVABILITY TECHNIQUES

There are various ways to carry out network protection, which can be broadly divided into *physical protection* and *topological protection*. The examples of physical protection include cable encapsulation, duct construction, cable burial, etc., which prevent fibers from being exposed to water and also improve their pliability. In contrast, topological protection takes advantage of network connectivity redundancy and uses a backup channel to substitute for a disrupted working channel when a failure occurs. However, due to the straightforward nature of physical protection, henceforth our discussion will be focused on topological protection only.

Any network layer capable of reconfiguring network topologies, which involves the fiber, wavelength-routed WDM, SDH/SONET, MPLS, and even IP tunnel layers, can apply network protection. Specifically, to ensure protection, a working path and a backup path, which are failure disjoint from one another, are established simultaneously between a pair of switching nodes. Upon a failure, a restoration process is initiated to use the backup path to substitute for the failed working path to recover the failure. Here the term of "switching nodes" is a layer-specific concept. Different layers have different types of switching nodes. These nodes however share a common feature—they must be active and possess switching capability. These switching nodes can be SDH/SONET switches, OXCs, etc., but they cannot be dull ones such as switch panels, manholes, etc.

Three important criteria—spare capacity efficiency, operational complexity, and restoration speed—are often considered when evaluating a survivability approach. Depending on the network layer in which network protection is applied, these three aspects vary a great deal. To facilitate our discussion, we broadly classify the network layers close to the physical layer as *system layers*, and those close to the network layer, i.e., IP layer, as *logical layers*.

The protection in a system layer is usually *span-oriented*. Automatic Protection Switching (APS), Unidirectional Path-Switched Ring (UPSR), Bidirectional Line-Switched Ring (BLSR), ring covers [MoGr01], generalized loopback networks [MéBa02], and span-protecting *p*-cycles [GrSt98a] all belong to this category. These techniques show two important features: (1) backup path pre-configuration, and (2) local node switching-over. When a network is in the normal state, the backup path for a certain network failure is always pre-determined and pre-configured. When the failure occurs, only the nodes adjacent to the failure carry out the restoration with a simple switching-over operation, but all the intermediate nodes on the backup path need not take any action, such as setting up switch status or making bandwidth reservation. Thus, these techniques are advantageous in having simple operation and fast restoration. They can often recover a failure within 50ms. However, these techniques usually suffer from a disadvantage of inferior spare capacity efficiency. Specifically, all the above survivability techniques except span-protecting *p*-cycles [GrSt98a] each require over 100% spare capacity redundancy. Such a high spare capacity redundancy is attributed to the strategy of spare capacity dedication adopted by most of these survivability techniques except span-protecting *p*-cycles. Span-protecting *p*-cycles allow for spare capacity sharing around the network, thereby enabling it to achieve a mesh-like spare capacity redundancy much lower than 100%.

In contrast, the protection in a logical layer is often *path-oriented*. Techniques such as Path Restoration (PR) [IrMa98], Shared Backup Path Protection (SBPP) [KiKo00], Path Segment Restoration (PSR) [HoHu02] [ShGr04a], Shared Backup Segment Protection (SBSP) [ShGr04a], and flow *p*-cycles [ShGr03a] belong to this category. Under these logical layer techniques, rather than the end-nodes of a failed span as in the system layer

techniques, the nodes that are responsible for switching-over are the source and destination nodes (under PR and SBPP), or upstream and downstream nodes (under PSR, SBSP, and flow *p*-cycles) of an affected working flow. To protect a working path, a backup path or segment, which is different from the working path, is established between the switching-over node pair. Because these survivability approaches allow for wider ranges of spare capacity sharing, they can often achieve better spare capacity efficiency than those in the system layer. On the other hand, they suffer from the penalties of longer restoration time and more complex operation. In these techniques, the backup path or segment is usually not pre-configured, which is either pre-computed only or not even pre-planned until a failure occurs. Thus, when recovering a failure, a signaling process is necessary to set up switch status and reserve bandwidth for the backup path (segment), which prolongs the restoration time and complicates the operation compared to the system layer techniques.

3.2.1. POINT-TO-POINT APS SYSTEMS

Point-to-Point Automatic Switch Protection (APS) include 1+1, 1:1, 1:N, and M:N strategies [WuTh92]. These approaches belong to the category of system layer survivability. 1+1 is the simplest and fastest protection technique. It simultaneously transmits signals on both working and dedicated backup channels. The receiver monitors the two received signals and chooses the one with better quality. 1:1 is similar to 1+1, but in contrast to the latter, it does not simultaneously send a copy of signal on the backup channel. Rather, the backup channel standbys or is used to transmit other lower priority traffic (for efficiency) when the system is normal. Only when a failure occurs would the backup channel be used to carry the affected working traffic. Compared to 1+1, 1:1 shows the advantage of better capacity utilization, but at the cost of a comparatively longer restoration time.

To achieve even better efficiency, 1:N, which is extended from 1:1, allows multiple working channels to share a single common backup channel. It assumes that there is only a single working channel failed each time. Consequently, a single backup channel suffices for the restoration of any channel failure. In addition, in order to ensure that the backup channel is always ready to recover any failed working channel, 1:N also needs to

take a step called "reversion" to return the traffic on the backup channel back to a working channel after the failed working channel is repaired.

Finally, it is also possible to extend 1:N to M:N, where M is the number of backup channels and N is the number of working channels. In this technique, any time a working channel fails, one of the protection channels will be used to recover the failure. To have better capacity utilization, in a similar way, the M protection channels are allowed to carry low priority traffic when the system is normal. Compared to 1:N, M:N is relatively more complex in control and operation, as M:N needs to choose one of M backup channels to recover a failure. However, it has the advantage of multi-failure protection to allow up to M channels to simultaneously fail but get them fully restored.

3.2.2. RING-BASED TECHNIQUES

Unidirectional Path-Switched Ring (UPSR) and Bidirectional Line-Switched Ring (BLSR) [WuTh92] [Grov04b] are the two most popular ring-based techniques.

A) Unidirectional Path-Switched Rings (UPSR)

UPSR can be perceived as packaging up of several logical 1+1 APS systems to share a common high-speed transmission system, but it is more flexible and economic than 1+1 APS. Figure 3.1 illustrates an example of UPSR protection switching operation. The system consists of a pair of fibers, with one carrying working channels and the other carrying protection channels. For a bidirectional working demand pair, each of the working paths is unidirectional, which makes the two working paths between a node pair form a unidirectional cycle on the ring. For example, for the bidirectional working demand pair between nodes A and D, a working path is established via route (A-E-D), and the other via route (D-C-B-A). These two working paths complete a unidirectional cycle on the ring as shown. The protection paths for the demand pair are established each in the reverse direction of the working paths. They also form a unidirectional circle on the ring, whose direction is opposite to that of the working paths. As in the case of the 1+1 APS system, in UPSR one working signal and one backup signal are transmitted via the working path and protection path simultaneously, and the receiver monitors the two signals to accept the better one. Figure 3.1(a) illustrates a UPSR under the normal operation (i.e., before failure). Figure 3.1(b) shows the recovering operation of UPSR

after a failure occurs. For example, if a failure occurs on span (C-D), the receivers of the two end nodes A and D will detect the loss of working signals. They will switch to receive the signals on the backup channels.

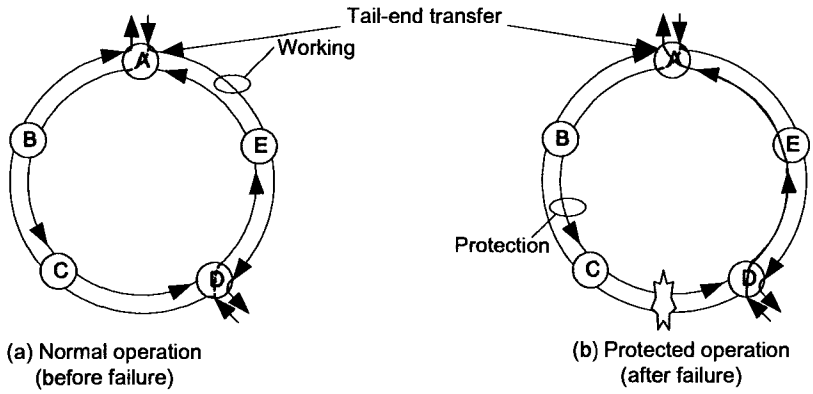


Figure 3.1. UPSR Protection Switching Operation.

B) Bidirectional Line-Switched Ring (BLSR)

Rather than switching-over at the source and destination nodes as in UPSR, BLSR carries out the restoration by local switching-over adjacent to a failure. There are two types of BLSR. One is 2-fiber BLSR, and the other is 4-fiber BLSR. In the former, half of the capacity in each fiber carries working channels and the other half is reserved for protection channels. Similar to UPSR, the signals transmitted in the fiber pair are in opposite directions. When a failure (say a span failure) occurs, a loopback process will be carried out at each local node of the failed span. By merging the remaining parts of the ring, a new unidirectional ring will be formed by using the spare on the ring to carry the affected working demands. The mechanism and operation of 4-fiber BLSR is similar to that of 2-fiber BLSR except that the former has four fibers with two fibers bidirectionally transferring working signals and the other two fibers bidirectionally reserved for protection channels. Figure 3.2 illustrates an example of 4-fiber BLSR.

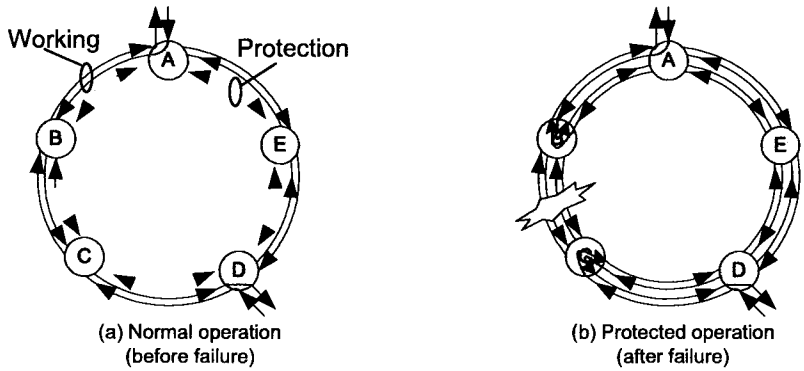


Figure 3.2. 4-Fiber BLSR Protection Switching Operation.

3.2.3. MESH SURVIVABLE NETWORKS

A) Span-Based Restorable Networks

Span-based restorable networks include the techniques of Span Restoration (SR) [HeBy95] [Grov97] and span-protecting *p*-cycles [GrSt98a].

Depending on whether or not the protection routes are pre-determined, SR can be implemented in two ways. In the first, all the protection routes and the assigned spare capacity for each protected span have been pre-determined and stored at the two end nodes of the span. When a span fails, the two end nodes of the span will initiate signaling processes to set up the pre-determined protection paths to complete the restoration. In the second way, all the protection routes and assigned spare capacity are not pre-determined, but in real time searched by the network whenever a failure occurs. This method is often referred to as a distributive *self-organizing* span restorable network [Grov97]. The advantage of this method over the previous one is that all the end nodes need not store any information on the working and spare capacities related to the restoration of span failure. Figure 3.3(a) illustrates an example for span restoration. When a span failure occurs on span (3-4), the method will use the protection paths (3-0-4) and (3-7-9-8-4) to restore the affected traffic on the span. Multiple protection paths are allowed to restore a failed span under the span restoration technique.

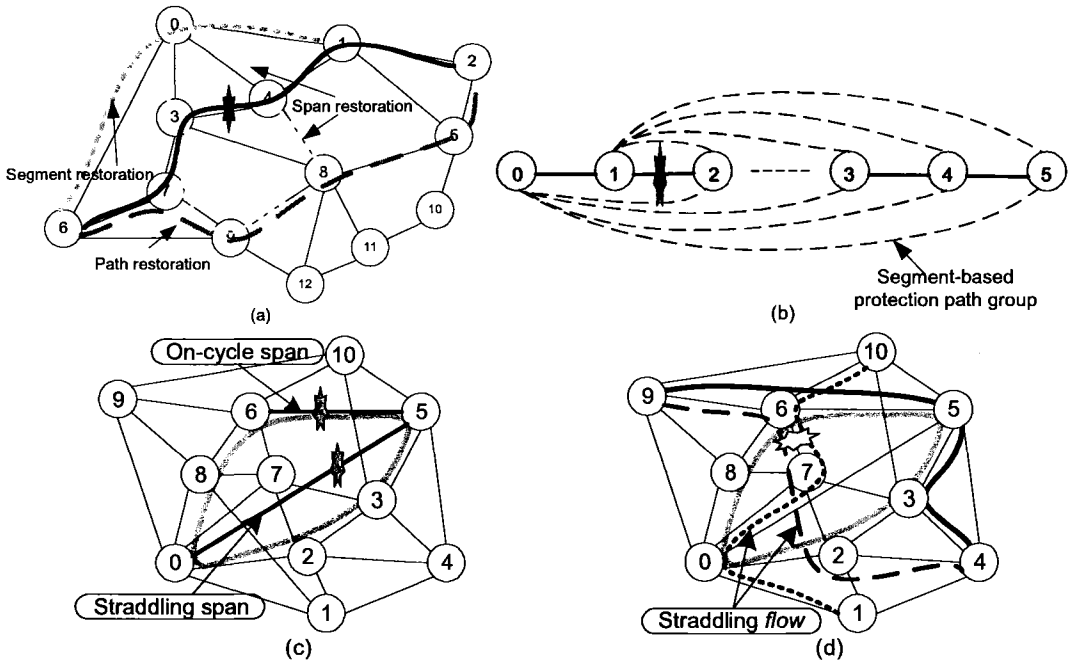


Figure 3.3. Examples of (a) Span and Path-Based Restoration, (b) Segment-Based Restoration, (c) Span-Protecting p-Cycles, and (d) Flow p-Cycles.

Span-protecting p-cycle is another span-oriented survivability technique that is extended from the conventional ring techniques. However, better than rings, p-cycles can protect not only on-cycle spans but straddling spans as well. Here "straddling span" is referred to as a span whose two end nodes are on the cycle, but itself is not on the cycle, whereas "on-cycle span" is referred to as a span that is itself on the cycle, which is essentially the same as a span on a ring under the ring survivability techniques. It is the straddling span protection that releases the constraint that working routes must strictly follow the trail of a ring as in the ring techniques, thereby allowing the establishment of working paths in a more efficient way. Compared to the ring techniques, p-cycles enjoy much better spare capacity efficiency, which was found to be close to that of the previous span restoration technique [GrSt98a]. Also, because p-cycles have the ring-like pre-configuration feature, they can recover a failure at a fantastic speed, which is close to that of the ring but much faster than the span restoration technique. Thus, the span-protecting p-cycles are often characterized as "ring-like switching speed and mesh-like spare capacity efficiency." Figure 3.3(c) illustrates an example of span-protecting p-cycles, in which there is a p-cycle traversing nodes (0, 2, 3, 5, 6, 8). All the spans on the cycle are on-cycles spans, which include spans (0-2), (2-3), (3-5), (5-6), (6-8), and (8-0). Span (0-5)

is a straddling span in that its two end nodes are on the cycle, but the span itself is off the cycle. When a failure occurs on an on-cycle span (5-6), the cycle will use the protection path (6-8-0-2-3-5) to restore the affected traffic on the span. Likewise, when a failure occurs on straddling span (0-5), the cycle will offer two protection paths (0-2-3-5) and (0-8-6-5) to recover the affected traffic on the span. We will give more detail on *p*-cycles in Chapter 4.

B) Path-Based Restorable Networks

Path-based survivability techniques realize restoration at the two end nodes of each affected working flow. Examples include Path Restoration (PR) [IrMa98] and Shared Backup Path Protection (SBPP) [KiKo00]. Path restoration allows multiple protection paths to restore an affected working flow, while SBPP allows only a single protection path to restore all the demands on an affected working flow. SBPP also requires that the protection path must be span or node-disjoint from the working path.

Figure 3.3(a) illustrates an example for path restoration in which there is a working path between node pair (2, 6). If span (3-4) fails, the working path (6-7-3-4-1-2) is disrupted. To restore the failure, protection path (6-9-8-5-2) will be used to carry the affected traffic. Specifically, path restoration has two sub-cases. One is path restoration *without stub release*, and the other is *with stub release* [IrMa98]. In this context, "stub" is defined as the remaining parts of an affected working path after a failure occurs. For example, in Figure 3.3(a), parts (6-7-3) and (4-1-2) are the stubs of the affected working path. When a working path fails, path restoration without stub release does not release the capacity on the two stubs, which was previously used by the working path, while path restoration with stub release always releases the capacity on the two stubs for use by the following restoration process. Theoretically, path restoration with stub release can achieve the best spare capacity efficiency (i.e., lowest spare capacity redundancy) among all the network survivability techniques, though it requires the longest restoration time and the most complex operation.

SBPP can be perceived as a special case of path restoration without stub release which requires that the working and protection paths must be span, node, or even SRLG-disjoint from one another and allows only a single protection path to restore all the traffic on an

interrupted working path. In Figure 3.3(a), the restoration path of the previous path restoration example cannot be used as a backup path for SBPP, as it shares some common spans [i.e., spans (6-7) and (7-9)] with the working path. Rather, we need to choose another backup path such as path (6-9-8-5-2) to function as the protection path for SBPP, which assures span and node disjointness[3] from the working path. Because SBPP allows only a single backup path to restore the affected working path, which has less flexibility in restoration path selection, it normally shows a spare capacity efficiency inferior to that of path restoration, though the difference is often marginal. Meanwhile, because SBPP is a failure-independent survivability scheme, no time is needed for failure location nor is a stub release required, thereby enabling SBPP to recover a failure faster than path restoration. The constraint of a single restoration path for each affected working flow is another advantage of SBPP, which helps to sustain the integrality for each restored working flow and can be useful to services requiring this kind of feature.

C) Segment-Based Restorable Networks

Segment-based restorable networks can be considered a compromise of span-based and path-based restorable networks that involve techniques such as Path-Segment Restoration (PSR) [ShGr04a], Shared Backup Segment Protection (SBSP) [ShGr04a], and flow *p*-cycles [ShGr03a]. PSR was extended from path restoration without stub release. Under the path restoration, the action of restoration is only taken at the two end nodes of the working path, whereas under the path-segment restoration, any segment is allowed to restore the affected working flow, as long as one of its end node is in the upstream of the affected flow, and the other node is in the downstream of the affected flow. Essentially, path segment is the most generalized concept. Besides segments of a complete path, path segment can also be used to model the complete path itself and a single span. That is, a working path in path restoration and a span in span restoration can both be seen as special path segments. Figures 3.3(a) and 3.3(b) illustrate the concept of path segments. In (a), for the failure of span (3-4), the restoration is carried out between segment end nodes 1 and 6, where (1-4-3-7-6) is a path segment whose segment end-nodes 1 and 6 are on the upstream and downstream of the affected working flow,

[3] Note that node disjointness does not require the source and destination nodes of the working and protection paths to be disjoint as well.

respectively. Also, for each working flow there can exist multiple segments able to restore a certain span or node failure. Figure 3.3(b) simply enumerates all possible segment restoration paths. These paths have been divided into several groups, and each group consists of all possible eligible paths between the two segment end nodes.

The relation of PSR to SBSP is similar to that of PR to SBPP. SBSP can be seen as a special case of PSR, but it can also be regarded as an extension of SBPP. Similar to SBPP, SBSP allows only one segment-based backup path to restore an affected working flow, which thereby is able to sustain the integrality of the restored flow as well.

In addition, as addressed, path and span can be considered as special cases of segments. Thus PSR and SBSP can be used to model Path Restoration (PR), SBPP, and Span Restoration (SR) as well by restricting the paths and spans as the unique special eligible segments. Compared to path restoration and SBPP, despite the complexity in network planning and operations, the segment-based methods have the advantages of faster restoration due to their shorter restoration segments, and better spare capacity efficiency because of a larger set of eligible backup routes for selection.

Finally, flow *p*-cycles were extended from span-protecting *p*-cycles to implement the *p*-cycle concept on path segments [ShGr03a]. Flow *p*-cycles keep the *p*-cycle pre-configured attribute, thereby giving the technique a restoration speed faster than other segment-based survivability techniques such as PSR and SBSP. Meanwhile, compared to span-protecting *p*-cycles, the feature of segment-based restoration also enables flow *p*-cycles to enjoy a better spare capacity efficiency. Nevertheless, the key constraint of flow *p*-cycles is its much more complicated operation and planning processes than span-protecting *p*-cycles. Thus, to take advantage of the merits of both span and flow *p*-cycles, a combined paradigm can be proposed to apply span-protecting *p*-cycles and flow *p*-cycles in a hierarchical fashion, wherein span-protecting *p*-cycles are used to protect the traffic flows within sub-networks (or local domains) for speed and simplicity, while flow *p*-cycles are used to protect the inter-domain and crossing-domain traffic flows for capacity efficiency [ShGr03a].

3.3. MULTIPLE-LAYER SURVIVABILITY

The survivability techniques discussed thus far always assume a single network layer. However, a transport network generally consists of multiple network layers, each of which may apply an independent survivability technique. Consequently, a well-known question on network survivability escalation arises: "For a network failure, which layer(s) should be responsible for the restoration?" [NeSt95] [DeGr99] To answer the question, we must consider two important issues, namely multi-layer restoration strategy and spare capacity assignment.

For a multiple-layer network, two types of network restoration strategies can be applied. One is termed *parallel strategy*, and the other is termed *sequential strategy* [NeSt95] [DeGr99]. The parallel strategy is one wherein restoration efforts are initiated at the same time in different layers upon a network failure. Whenever any of them succeeds in the restoration, all the other efforts should be stopped. Despite faster restoration, the cost of coordinating the restorations in different layers to avoid the contention for the common network resources is often prohibitive. In contrast, the sequential strategy is much easier in control and operation, though it probably incurs a longer restoration time. For this strategy, we may further subdivide it into two sub-cases, i.e., top-down strategy and bottom-up strategy [NeSt95] [DeGr99].

Under the top-down strategy, the network restoration is always carried out in the top layer first. Only if the failure cannot be fully restored, would the restoration responsibility be transferred down to the lower layers until all the traffic gets fully restored. This strategy is usually recommended when (1) multiple reliability grades need to be provided with fine granularity, (2) multi-layer parallel recovery is impossible due to system limitation, and (3) the survivability schemes in the upper layers are more fledged than the lower layers. In contrast, the bottom-up strategy requires that the network restoration always be undertaken in the lowest layer first. The restoration effort in the upper layers will never be initiated unless the lower layers cannot fully restore the failure. The bottom-up strategy is recommended when (1) the number of entities to recover must be limited or reduced, (2) the lower layers are supporting multi-types of services and it is appropriate to provide a homogeneous survivability for the services, and (3) the survivability

schemes in the lower layers are more mature than those in the upper layers. In summary, the bottom-up strategy is generally associated with a smaller number of restoration entities and thus shows a faster restoration speed, while the top-down strategy has a better spare capacity efficiency owing to its finer restoration granularities, but may take a longer time for restoration.

For the multi-layer network survivability, we also need to consider the issue on the spare capacity assignment. In the traditional way, each network layer is independently planned for spare capacity, and the total capacity that sums the working and spare capacities in an upper layer (i.e., client layer) is regarded as the working capacity and then transferred to a lower layer (i.e., server layer), which in turn assigns spare capacity to protect this working capacity. Obviously, it is inefficient to assign spare capacity in a lower layer to protect spare capacity in an upper layer. Even worse, such inefficiency would be transferred down layer by layer until the lowest layer under the multi-layer circumstance. As a result, the lowest layer would reserve a huge amount of spare capacity purely to protect the spare capacities in upper layers. To avoid such an inefficient situation, an economical scheme called *common pool survivability* was proposed in [DeGr99], which allows the spare capacity sharing among different layers. For example, we may reserve the spare capacity in wavelength-routed WDM layer as a *common spare capacity pool* to simultaneously offer protections for both wavelength-routed WDM and SDH/SONET layers. By doing this, the redundancy of the spare capacities in upper layers can be largely compressed, and thus the overall capacity utilization can be significantly improved.

CHAPTER 4
P-CYCLE TECHNIQUES

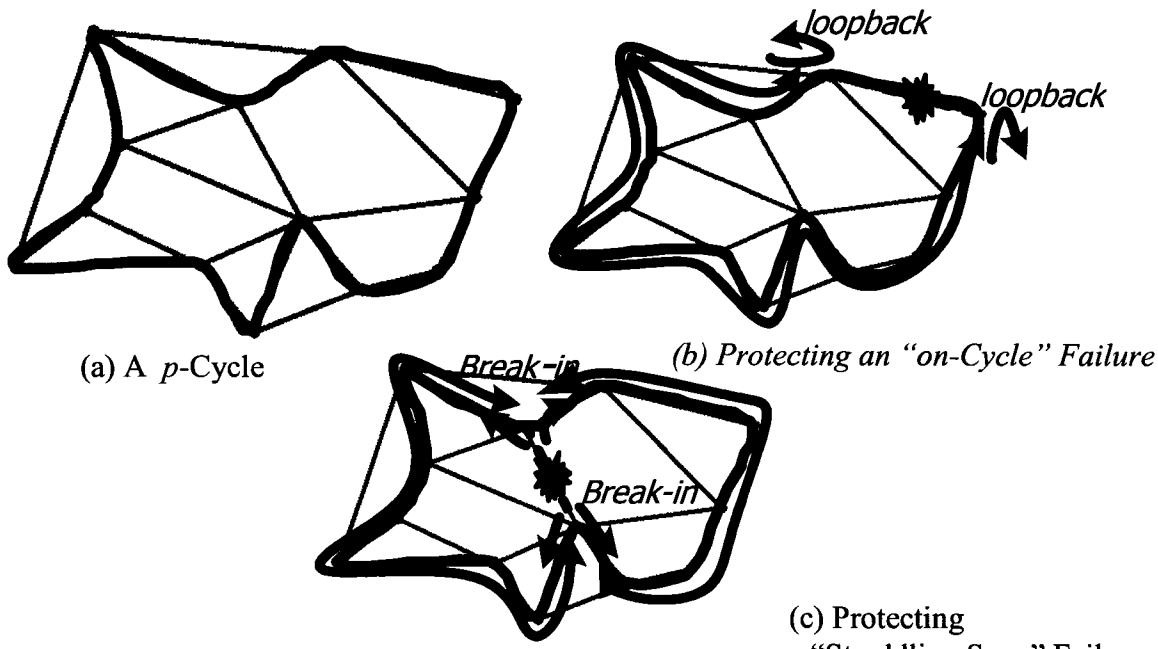
This chapter will provide more detail on the *p*-cycle techniques introduced in Chapter 3. *p*-Cycles can be subdivided into two categories: span-protecting *p*-cycles [GrSt98a] and segment-protecting *p*-cycles [ShGr03a]. The segment-protecting *p*-cycles are also termed flow *p*-cycles, which were extended from span-protecting *p*-cycles. The concept of flow *p*-cycles is general enough to model the span-protecting *p*-cycles as well.

4.1. SPAN-PROTECTING *P*-CYCLES

One of the most interesting recent developments in survivable network architecture is the method of *p*-cycles. *p*-Cycles, introduced in 1998 [GrSt98a], are in a sense like BLSR rings, but with support for the protection of *straddling* span failures as well as the usual protection of spans on the ring itself. An important property of *p*-cycles is that the cycles are fully pre-configured with pre-planned spare capacity. When a span failure happens, only the two end nodes of the span do any real-time switching, but no switching actions are required at any intermediate nodes of the cycles. This property improves the *p*-cycles restoration speed to be essentially the same as BLSR rings. This is an important advantage over restoration schemes and over other "protection" schemes such as Shared Backup Path Protection (SBPP), where the protection routes are pre-planned but all the switches on the intermediate nodes of these routes need to seize and cross-connect spare capacity in real time upon a failure. The latter always implies a two-way signaling protocol (e.g., RSVP-TE protocol, which consists of a PATH message and a RESV message) to fulfill this process when a failure occurs. Thus *p*-cycles have an inherent speed advantage—they inherit BLSR ring-like speed.

Protecting straddling span failures is another unique property of *p*-cycles that enables networks to be designed with essentially the same capacity-efficiency as a span-restorable mesh network. This means *p*-cycle based networks can be 3 to 6 times more capacity-efficient than ring-based networks, while still providing BLSR ring switching speeds. In addition, when it comes to efficiency, another important factor is that *p*-cycles are "spare-capacity only" structures that—unlike rings—do not constrain the routing of

working paths to coincide with the layout of the cycles themselves. Working paths are therefore free to take the shortest mesh-like routes between their end-points on the graph. In fact, because each unit of protection capacity on a p-cycle can protect two units of working capacity on a failed straddling span it is more advantageous for working capacity to "straddle" p-cycles than to underlie the p-cycles themselves. Figure 4.1 illustrates the protection of basic "span protecting" p-cycles.



(a) A p-Cycle

(b) Protecting an "on-Cycle" Failure

(c) Protecting
a "Straddling Span" Failure

Figure 4.1. The Basic Concept of Span-Protecting p-Cycles.

In Figure 4.1(a) a single p-cycle of one channel of protection capacity is shown in a small network. Note that this happens to be a Hamiltonian p-cycle, a special case in which a single p-cycle can offer protection to every span of the network. All spans are either protected in an "on-cycle" manner as in (b) or as "straddling" spans as in (c). For the failure of an on-cycle span, the p-cycle will offer one protection path as in ring, whereas for the failure of a straddling span, two proteciton paths as shown will be offerred to restore the span failure.

The number of opportunities for straddling span establishment on a given p-cycle depends on the relationship between the cycle and the graph topology. Straddling spans need not, however, only be visually "inside" the cycles (see [GrSt00] for an example). A single Hamiltonian cycle, if it exists on a graph, will establish an on-cycle or straddling

relationship to all network spans, but more generally we are considering *p*-cycle designs using multiple non-Hamiltonian unit-capacity cycles for reasons of availability, transmission performance, and capacity-efficiency.

As is now well understood, and validated by other research groups including [ScGr02], it is the admission of straddling span protection relationships that dramatically reduces the total design redundancy relative to ring-based networks, to the point of near-equivalence with a span-restorable mesh network. It is somewhat remarkable because the addition of straddling spans to a BLSR seems like such a minor technical difference [GrSt98b]. And yet its significance in terms of network efficiency is huge: one goes from ring-like redundancies of 200% or more, right to mesh-like redundancies of under 50% to 60% to support the same set of service demands. Put the other way around, within a given set of installed facilities, *p*-cycle based operation can typically support two to four-times growth in the entire demand served, *without adding new capacity and without giving up the protection speed of existing ring-based networks* [ClGr01].

4.1.1. Literature Survey

Since 1998, the basic theory of *p*-cycles as pre-configured structures in the spare capacity of a mesh network has been developed [Stam97] [StGr00a], and there have been studies on self-organization of the *p*-cycle sets [StGr98], application of *p*-cycles to the MPLS/IP layer [StGr00b], application to DWDM networking [ScGr02] and studies on joint optimization of working paths and spare capacity [GrDo02]. Notably in [ScGr02] it was found that full survivability against any span cut could be achieved with as little as 39% total redundancy. This greatly motivates further work and practical applications of the *p*-cycle concept. A more comprehensive review paper on *p*-cycles provides additional background [GrSt00].

Concurrent work on logical rings and cycle covers [ElHa00] under the (coincidentally similar) name of "protection cycles" should not be confused with *p*-cycles. The fundamental difference concerns straddling spans in *p*-cycles. Straddling spans on a *p*-cycle can each bear two units of working capacity per unit of *p*-cycle capacity, and require *no* associated protection capacity on the same spans. All forms of ring or cycle covers fundamentally involve equal (or greater) amounts of protection and working

capacity on every span and *at best* (which is what *oriented* cycle double covers in [ElHa00] accomplishes) reach a 1-to-1 ratio between these, for 100% redundancy. In contrast, *p*-cycles yield fully restorable architectures at well under 100% redundancy. This is due ultimately to the effectiveness of straddling span protection aspects. For more on rings, cycle covers, and generalized loopback networks and how they differ from *p*-cycles, see [Grov02].

4.1.2. Optimal Design for Span-Protecting *p*-Cycle Networks

Integer Linear Programing (ILP) apporaches can be used to design span-protecting *p*-cycle networks with the objective of minimizing total required spare capacity, or total network cost if the working and spare capacities are allocated in a joint fashion [GrSt98a] [GrDo02]. In the following, we present the optimization models for *p*-cycles Spare Capacity Assignment (SCA) and Joint Capacity Assignment (JCA). The common notations used in the models are listed below:

- $\boldsymbol{S}$ is the set of network spans.
- $\boldsymbol{P}$ is the set of all eligible cycles in the network.
- C_j denotes the unit capacity cost on span j.
- s_j and w_j denote the number of spare capacity units (channels) and working channels on span j, respectively.
- n_i is the number of copies of cycle i in the p-cycle design.
- X_i^j is the number of restoration paths that cycles j can provide for failed span i. If span i is an on-cycle span of cycle j, which means that the cycle can provide one restroation path for the span, then X_i^j takes the value of one. If span i is a straddling span of the cycle, which means that the cycle can offer two protection paths for the span, then X_i^j takes the value of two. Otherwise, X_i^j take the value of zero.
- P_k^j indicates the topological relationship between cycle j and span k. It takes the value of one if the cycle includes the span. Otherwise, the value is zero.

A) Model for Spare Capacity Assignment (SCA) [GrSt98a]

Given a forecasted demand matrix, the SCA model assumes that the working capacity has been pre-assigned based on some simple routing strategy such as the shortest path routing algorithm. The working capacity w_i is thus a given parameter. The model aims to minimize the total required spare capacity cost in the network, and ensures the full restoration against any single-span failure.

SCA: minimize $\sum_{k \in \boldsymbol{S}} C_k \cdot s_k$ (4-1)

Subject to:

$$s_k \geq \sum_{j \in \boldsymbol{P}} P_k^j \cdot n_j \quad \forall k \in \boldsymbol{S} \tag{4-2}$$

$$w_i \leq \sum_{j \in \boldsymbol{P}} X_i^j \cdot n_j \quad \forall i \in \boldsymbol{S} \tag{4-3}$$

$$n_j \geq 0 \quad j \in \boldsymbol{P} \tag{4-4}$$

Constraint (4-2) ensures that span i is assigned enough spare capacity to support the number of copies of each p-cycle that traverses the span. Constraint (4-3) shows that there are enough copies of p-cycles pre-configured in the network to guarantee the full restoration against any single-span failure.

B) Model for Joint Capacity Assignment (JCA) [GrDo02]

To have a better network capacity utilization, we can further design a p-cycle network in a joint optimization fashion, wherein only a forecasted demand matrix and a network topology are given. The objective is to minimize the total required network design cost and determine the working and spare capacity assignments. In the model, both of the working and spare capacity, s_j and w_j, are variables. For the joint model, new notations additional to those of SCA are listed below:

- $\boldsymbol{D}$ is the set of demands between node pairs in the network, which is indexed by r.
- $\boldsymbol{Q}^r$ is the set of eligible working routes between node pair r, which is indexed by q.
- d^r is the number of demand units between node pair r.

- $g^{r,q}$ denotes the demand flow assigned on the eligible route q to serve the demand between node pair r.
- $\zeta_k^{r,q}$ is a topological term to reflect the relationship between the eligible routes and spans. It takes the value of one if working route q between node pair r transits span k; zero, it is otherwise.

JCA: minimize $\sum_{k \in S} C_k \cdot (s_k + w_k)$ (4-5)

Subject to:

Constraints (4-2) and (4-3)

$$d^r \leq \sum_{q \in Q^r} g^{r,q} \quad \forall r \in D \tag{4-6}$$

$$w_j = \sum_{r \in D, q \in Q^r} g^{r,q} \cdot \zeta_j^{r,q} \quad \forall j \in S \tag{4-7}$$

In addition to the previous two constraints, constraint (4-6) ensures that the demand between node pair r gets fully served. Constraint (4-7) computes the required working capacity on each span to support routing of all demands.

4.2. SEGMENT-PROTECTING *P*-CYCLES

It is important to note, however, that all the previous studies on p-cycles have considered what we call "span-protecting" p-cycles, which operate as shown in Figure 4.1. Each such p-cycle protects only spans that are part of itself or that *directly* straddle the respective p-cycle. In fact it was natural, even at the time of the first work in [GrSt98a] and [Stam97], to ask whether there was a *path* protection equivalent to basic span-protecting p-cycles. As simple as basic p-cycles is the later question turned out to be difficult to address. Ultimately the difficulty is how to handle the aspect of "mutual capacity" contention that is intrinsic to any path oriented or multi-commodity flow type of recovery scheme in formulating the design model under a paradigm of cyclic spare capacity structures. Recently, these difficulties have been largely overcome by basing the

approach on *path segment-protecting p-cycles*, which are also termed *flow p-cycles*[4] [ShGr03a].

Figure 4.2(a) illustrates a span-protecting p-cycle that, unlike the example in Figure 4.1, is not a Hamiltonian cycle and thus does not by itself maintain an on-cycle or straddling relationship to every span in the network. Specifically, the spans (0-2), (2-3), (3-5), (5-6), (6-8), and (8-0) are *on-cycle* spans of the p-cycle and the span (0-5) is a *straddling* span, for its two end nodes are on the cycle, but the span itself is not on the cycle. Thus, if span (5-6) fails, the route (5-3-2-0-8-6) on the cycle would offer one protection path to restore the failure, in exactly the way BLSR operates. If span (0-5) fails, paths (0-2-3-5) and (0-8-6-5) are both (simultaneously) available to provide protection paths. Note, however, in Figure 4.2(a) that spans (6-7) and (7-2), and several others, of the basic p-cycle are *close* to being straddling spans but cannot actually be span-protected by the cycle. Meanwhile, a *path* that crosses both spans (6-7) and (7-2), as shown in Figure 4.2(b), *can* be considered to straddle the cycle when taking a path-level view of only the one demand that does flow all the way from side to side across the p-cycle. More specifically, the *path segment* (6-7-2) can be considered to straddle the cycle, between the nodes 6 and 7.

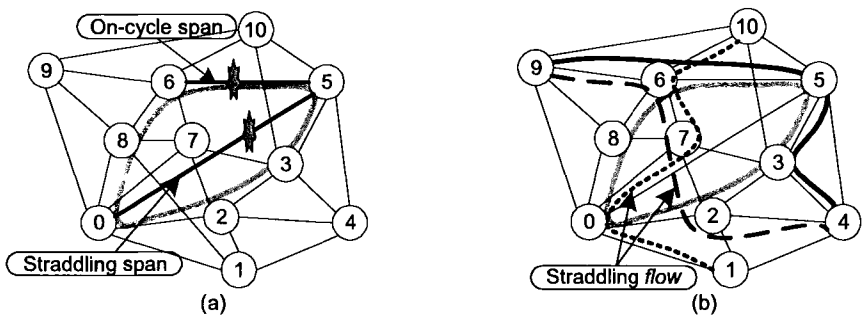


Figure 4.2. Examples of (a) a Span Protecting p-Cycle and (b) the Same Cycle Viewed as a Flow-Protecting p-Cycle.

These simple observations are the starting point for the fully generic concept of flow protecting p-cycles. To develop this, we must now take an explicit view of the service paths crossing the network and how they relate to the p-cycles we may use for protection.

[4] The flow p-cycle survivability technique is another important research contribution of my doctoral program. Nonetheless, to make the thesis consistent, we will only briefly mention the technique. Interested readers are referred to our JSAC paper [ShGr03a] for more detail on flow p-cycles.

Figure 4.2(b) depicts this more highly resolved viewpoint, wherein we can design a flow *p*-cycle protected network. For convenience, let us define a *flow* as any single contiguous segment of a working end-to-end path. Thus, the entire path, each span on the path, and any sequence of several spans along the path are flow segments. Flow *p*-cycles enable any flow segment that intersects a flow *p*-cycle, not simply spans directly on or straddling the cycle, to be protected. For example, if spans (2-7) and (6-7) in Figure 4.2(a) incur failures, they cannot be restored by a span-protecting *p*-cycle. However, under the flow *p*-cycle shown in Figure 4.2(b), the contiguous flow that traverses spans (2-7) and (6-7) can be restored by the cycle as well. Two alternate routes, namely (6-8-0-2) and (6-5-3-2), are available to recover the failure. Moreover, flow *p*-cycles have another advantage of node failure recovery. Flow *p*-cycles can recover transit traffic demands due to the loss of an intermediate node on a flow. As shown in Figure 4.2(6), a failure of intermediate node 7 can break the flows between node pairs (1-10) and (4-9). The span-protecting *p*-cycle in Figure 4.2(a) likewise cannot restore these affected transit flows. However, under the flow *p*-cycle as shown in Figure 4.2(b), the flows that transit node 7 can be recovered by the cycle. Note that any demands being added or dropped at node 7 cannot be handled by the same flow *p*-cycle. The flow must be unchanged in its composition between the nodes where it intersects the flow *p*-cycle.

Theoretically, flow *p*-cycles can be categorized as the *p*-cycle equivalent to path restoration with failure-specific stub-reuse. This is intermediate between completely general stub release (as in [IrMa98] [IrGr00]) and the complete prohibition against reuse of stub capacity that is built into the recent Shared Backup Path Protection (SBPP) scheme. In stub *reuse* the surviving path up to the protected segment is always part of the end-to-end path in the restored state. In other words, the surviving stub path segments are always reused in a failure-specific way by the same path that used that capacity before the failure. Unlike stub release, which can permit the same or any other simultaneously failed path to exploit released stub capacity of failed paths, the stub reuse that is implicit in flow *p*-cycles requires no explicit real-time actions to effect it.

4.3. CLOSING REMARKS

p-Cycles were invented as a span-protecting technique, which was then extended to segment-protecting *p*-cycles. As a span-based protecting technique, span-protecting *p*-cycles can achieve a ring-like switching speed and a mesh-like spare capacity efficiency, thereby placing itself at the best position in the tradeoff context of restoration speed versus spare capacity efficiency. Meanwhile, as a segment-based protecting technique, flow *p*-cycles not only perform at a good spare capacity efficiency close to that of path restoration, but also achieve a much faster restoration speed than other path-based or segment-based approaches.

Nevertheless, a common misunderstanding on the *p*-cycle techniques is that they are only planning rather than provisioning techniques, and consequently, they may not be applicable to real network operation, or in efficiency can be extremely poor if used to provision dynamic survivable services. In order to eliminate this confusion, further research is necessary to efficiently apply the *p*-cycle techniques in dynamic operational environment, which will be the focus of the following thesis research.

CHAPTER 5
GMPLS-BASED SBPP SURVIVABLE SERVICE PROVISIONING

Dynamic service provisioning and network survivability are two important aspects of transport networks. Dynamic service provisioning requires the networks to offer services in a dynamic fashion, but they do not have any *a priori* knowledge on future requests. Network survivability is essential to network service provisioning, which ensures that a provisioned network service will survive any network failure. Depending on the Service Level Agreement (SLA) on Quality of Protection (QoP), the provisioned services can be assured with various levels of protection, which involve span, node, and even SRLG failure protection. Currently, a host of techniques have been proposed to provision dynamic survivable services, among which the Shared Backup Path Protection (SBPP)-based technique is perceived the most promising [KiKo00].

In this chapter, we will give a detailed introduction to the SBPP-based dynamic survivable service provisioning. The concept of Generalized Multiple Protocol Label Switching (GMPLS) and its related terms are first introduced, followed by a detailed description of the SBPP technique. Next, issues on how to apply the GMPLS-based control infrastructure to provision SBPP-based survivable services are addressed, of which link state synchronization and signaling processes are specially highlighted. The structure of network state database is elaborated in detail.

5.1. GENERALIZED MULTIPLE PROTOCOL LABEL SWITCHING (GMPLS)

5.1.1. THE FUNDAMENTALS OF GMPLS

GMPLS is a new network control infrastructure extended from the conventional MPLS technique. GMPLS aims to employ a single set of unified IP-based control protocols to control almost all types of networks ranging from packet switching networks such as IP/MPLS and ATM networks, to circuit-oriented networks such as SONET/SDH and wavelength-routed WDM networks [Mann03]. Prior to GMPLS, each network is generally controlled by its own independent control protocol suite. For instance, ATM

uses the User-to-Network Interface (UNI) and Private Network-to-Network Interface (PNNI) protocols [FiWa94], while SONET/SDH uses the Telecommunication Network Management (TNM)-based control system [YaSa95]. As a new type of transport network, wavelength-routed optical networks are often managed by some vendor-specific private software packages such as a Simple Network Management Protocol (SNMP)-based centralized control system. However, with the emergence of GMPLS, a single GMPLS protocol suite can be applied to control almost all types of networks by simply regarding them as special cases under the GMPLS general umbrella. Particularly, besides the Packet Switching Capable (PSC) networks such as IP, ATM and MPLS, GMPLS is also able to support non-packet switching networks, including Layer2 Switching Capable (L2SC), Time-Division Multiplexing (TDM), Wavelength Switching Capable (WSC), and Fiber Switching Capable (FSC) networks [Mann03].

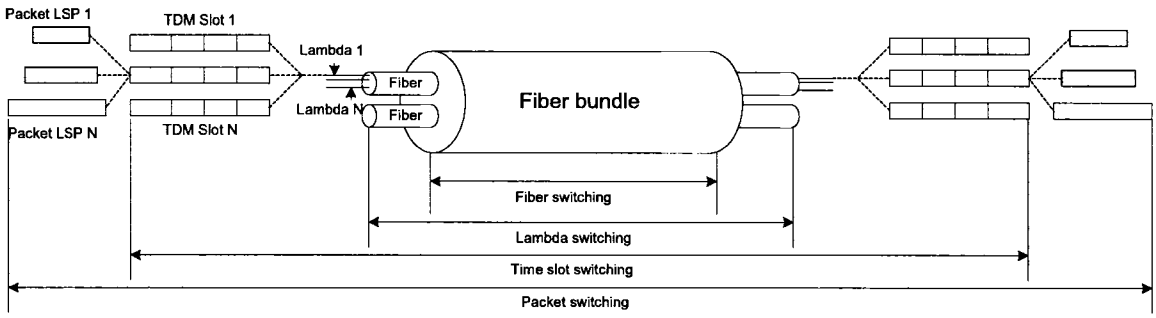


Figure 5.1. Hierarchical Switching Layers Supported by the GMPLS Technique.

Figure 5.1 illustrates all the hierarchical layers that can be supported by the GMPLS technique. MPLS functions as a switching mechanism for the *packet layer*. MPLS packets are encapsulated into TDM frames, and in the *TDM layer*, the TDM frames are switched by the GMPLS technique based on their time-slot indexes. The high bit-rate tributaries such as OC-48 and OC-192 are then modulated onto wavelengths. In the *wavelength layer*, tens or even hundreds of wavelengths are multiplexed in each fiber and also switched by the GMPLS technique based on their individual wavelength indexes. Finally in the *fiber layer*, there can be up to hundreds of fibers encapsulated in a single fiber cable and a host of fiber cables laid through a conduit. The GMPLS technique switches each fiber by considering the fiber indexes as labels.

Essentially, GMPLS has made three major extensions to the traditional MPLS technique: (1) generalizing the label space to any type of interface, (2) assigning IP addresses to "non-packet" terminating interfaces, and (3) supporting traffic engineering concepts and attributes to "non-packet" resources. With these extensions, GMPLS brings the network control system two significant benefits, as described below.

First, it improves the flexibility and reliability of a network control system. The flexibility improvement is inferred from the fact that GMPLS can support a wide range of networks, from a simple "thin" network, such as IP directly over optical network [RaLu04], to the traditional networks with the most complicated layer architecture such as an IP/ATM/SONET/WDM network. Thus, GMPLS possesses the freedom to insert a new network layer or to eliminate any redundant network layer in a network, without significantly changing the network control system. As for the reliability improvement, because GMPLS is characterized of a unified control infrastructure and applies a homogeneous set of control protocols for each of the network layers, the risk of such as network control system errors due to the mismatch between the control protocols of two neighboring layers, which frequently occurs in the traditional layered architecture, can be significantly reduced.

Second, GMPLS enables all the network layers to "speak" a *common control language*, which enhances communications between different network layers and thus provides opportunities to significantly improve the network resource utilization. Specifically, the common control language allows the network to carry out some cross-layer operations, which involve (1) *integrated routing* which takes into account network resources in multiple network layers when provisioning a new service, and (2) multi-layer spare capacity sharing, which is also termed *common pool survivability* [DeGr99], allowing for spare capacity sharing in multiple network layers in order to improve the spare capacity efficiency. For example, in the peer model of IP over optical networks, routers in the IP layer and OXCs in the optical layer are allowed to share their network resource information [RaLu04]. Thus, when a switch node computes a new connection for a node pair, it is possible to consider the network resources in both of the layers to obtain the best overall network resource utilization. In contrast, under the traditional control

infrastructure, it is usually of extreme complexity or even impossible to achieve functions similar to the above due to lack of a common control language.

The last but the most important advantage of GMPLS is that it can save network costs, which consists of control system developing costs and network operational costs. The powerful functionalities of GMPLS allows network vendors to concentrate their efforts on developing a single unified homogenous protocol suite only, rather than multiple impendent heterogeneous suites as in the traditional network infrastructure. Thus, this is helpful to the development of a cheaper but more efficient and reliable unified control system to support all the network layers. As for the operational cost saving, it is mainly inferred from two aspects, namely (1) better network resource utilization and (2) expense-cut for training engineers. The costs of transport network equipment are usually expensive, so it would be of value to efficiently utilize network resources. GMPLS can support cross-layer functions such as multi-layer traffic engineering, common pool survivability, etc., thereby efficiently improving the network resource utilization. Similarly, using GMPLS as a single network control architecture to administer and manage networks requires that engineers master only a single network protocol suite (i.e., GMPLS), rather than a large set of different control protocols as before, thereby saving a great much money for staff training.

5.1.2. Relevant Terms and Concepts

This section will provide brief overview of the terms and concepts that are relevant to GMPLS.

A) Traffic Engineering

Traffic engineering, which is important to transport networks, functions to improve the network resource utilization. Under traffic engineering, traffic flows in a network are routed based on two criteria, namely (1) the resources required by a traffic flow and (2) the resources available in the network. Constraint-based routing (CBR) algorithms are usually employed to search optimal routes for traffic flows based on the traffic engineering metrics such as available bandwidth, time delay, delay jitter, packet loss ratio, service priority, and others. Also, traffic engineering may be involved in the support for link or node failure recovery [Swal99].

B) Bidirectional LSP

Bidirectional LSP means that each service connection has two LSPs existing collaterally between the source and destination node pair, and the two LSPs are in the opposite directions. In the GMPLS networks, the assumption of bidirectional LSP is often made by default. Using a single set of signaling messages, GMPLS is able to simultaneously establish a pair of downstream and upstream paths, which traverse the same route but in the opposite directions. Details of the signaling process will be addressed later.

C) IP Addressing

GMPLS is a class of IP-based control protocol, in which each of the controlled entities ranging from switch interfaces to internal elements are assigned a unique IP-related address in order to discern them. There are two effective ways to meet the requirement. One is to assign a unique IP address for each interface or element. This method is advantageous in that it uses a single IP address to globally identify each of the controlled entities. However, it may suffer from the potential crisis of "not enough IP addresses," especially for the future optical transport networks, in which thousands of wavelengths are anticipated to be simultaneously transmitted in each fiber. The second method is more scalable. It assigns a unique IP address for each switch node, but for the interfaces or internal elements of the switch, only a local index is allocated to distinguish each of them. This method globally identifies each interface or element by a pair of an IP address and a local index. The IP address is the node address and the local index is unique at the node. The advantages and disadvantages of the second method is that it can save IP addresses, but at the cost of a longer information pair to discern each network interface or element [Mann03].

D) Unnumbered Links

The concept of unnumbered links is closely related to the IP addressing. To save IP addresses, the interfaces and elements of each node are only locally assigned unique indexes. Unnumbered links are referred to as the links not globally identified by unique IP addresses, but locally by unique indexes allocated by the end-nodes of each (bidirectional) link [Mann03]. To match these indexes for each link, an index-matching table is needed at each node. For this, a protocol called Link Management Protocol (LMP)

is specially developed to be responsible for exchanging the information on these link indexes between the two end-nodes of a link [Lang03]. We will provide more detail on LMP later.

E) Link Bundling

In Link Bundling, all the links that share the same Traffic Engineering (TE) properties are integrated and perceived as a single TE link in the network routing table [Mann03]. Each of the links in the TE link is termed a *component link*. The TE link provides a property summarization for all the contained component links. Routing protocols such as OSPF/IS-IS disseminate only this summarized information and use it to compute constraint-based routes. This differs from the traditional routing processes, where the traffic metrics of *each* component link should be maintained by the routing protocols. With up to thousands of component links in each TE link, it is an extreme challenge (in the aspect of scalability) for a routing protocol to synchronize all the link state information and, when computing a route, consider all the component links. However, by performing information aggregation/abstraction, the concept of link bundling can significantly compress the amount of TE-related information and thereby improve the scalability of routing protocols.

5.1.3. GMPLS Infrastructure and Operations

To further understand the GMPLS switching technique, Figure 5.2 provides an overview of a general GMPLS control infrastructure, which contains three major components, namely *routing component*, *signaling component*, and *Link Management Protocol (LMP) component*.

A) Routing Component

The routing protocol of GMPLS was extended from the conventional OSPF/IS-IS protocols, whose major functions include (1) maintaining aliveness of neighboring routing adjacencies, (2) synchronizing network state database, and (3) computing routes for connections [KoRe03a] [KoRe03b].

The function of maintaining aliveness of neighboring routing adjacencies is realized by exchanging "hello" messages between the neighboring nodes, which is the same as the process in the conventional routing protocols.

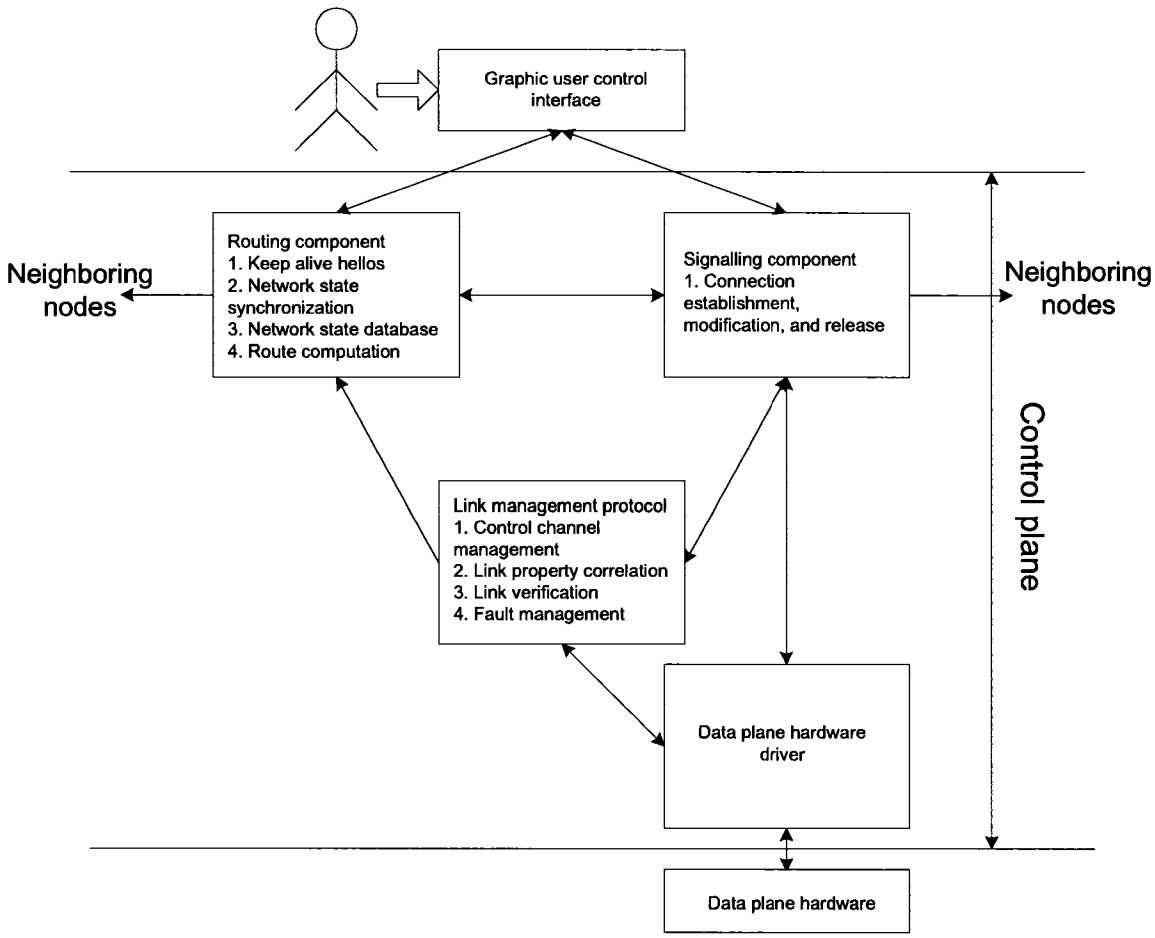


Figure 5.2. A Schematic Overview of GMPLS Control Infrastructure.

The process of network state database synchronization is also the same as that in the conventional protocols to comprise the basic steps such as database description exchanges and link state update/request/acknowledgement.

Network state database records the state information of each link, which is synchronized by the process of Link State Advertisement (LSA). An LSA message carries the information on newly-consumed or available bandwidth on each link. It can also carry some QoS or policy-related information. Additionally, if survivable services are considered, it can carry information related to spare capacity sharing. Network state database is critical to the routing component and closely related to all the routing processes. Specifically, the process of network state synchronization enables the network state database to be always synchronized around the network and the process of route computation needs to query the network state database when computing a route.

Finally, the route computation process of GMPLS can support Traffic Engineering (TE), which is more advanced than the counterpart in the conventional routing protocols. Based on the link state database, the computation process searches eligible explicit routes for node pairs, which takes into consideration constraints such as QoS and policy issues. The QoS metrics can include guaranteed bandwidth, constrained network delay, etc., and the policy issues specifically depend on different network service providers. The Constrained Shortest Path First (CSPF) routing algorithm is usually used to compute the routes. Moreover, if survivable service provisioning such as SBPP-based provisioning is considered, the route computation will search for a protection route, which is link-disjoint from the working route, and maximally shares spare capacity in the network.

B) Signaling Component

The signaling component functions to establish, modify, query, and release service connections [Berg03]. The current GMPLS-based signaling protocols fall into RSVP-TE [AwBe01] and CR-LDP [Jamo02], which realize equivalent signaling functions. Without losing generality, henceforth we assume that RSVP-TE is the default signaling protocol.

Given an explicit route found by the routing component, the signaling process to establish a new connection is illustrated in Figure 5.3. The process consists of two messages, namely PATH and RESV. The PATH message functions to collect network resource information on the route and examine whether the route can meet the related constraints and policies. The message is forwarded from a source node to a destination node. Upon receiving the message, the destination node (node 3) determines whether it is possible to establish a new connection on the route. Depending on various situations, node 3 may indicate that the route cannot offer sufficient resources or cannot meet the constraints. Consequently, a message will be sent back to the source node (node 1) to decline the request. Alternatively, it triggers a new message, i.e., RESV, back to node 1 to acknowledge the request and set up the connection. The RESV message reserves network resources and configures the switching status of the intermediate nodes enroute hop-by-hop back to the source node. After the message reaches node 1, which implies that the connection has been established successfully, the application data can be transmitted on the new connection. As indicated, because the connections of GMPLS are

usually bidirectional, the signaling process above can establish two LSPs simultaneously, with one in each direction.

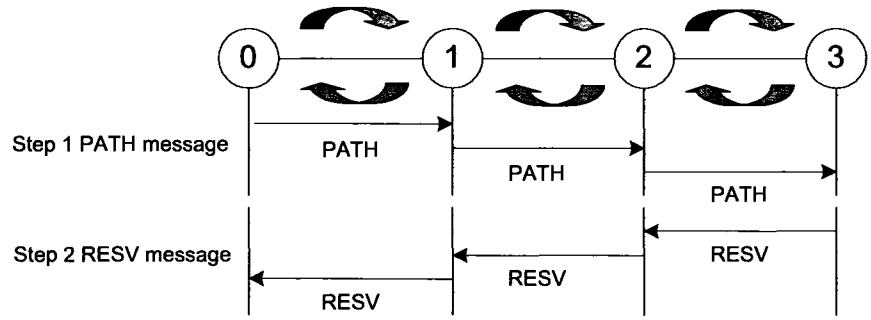


Figure 5.3. The RSVP-TE Signaling Process.

Examples of connection modification involve situations like bandwidth increase or decrease for an existing connection, connection route change, etc. The process of connection modification is similar to that of connection establishment, which also involves a pair of PATH and RESV messages to determine whether the state of the connection can be modified based on the current network resources. Additionally, it is often preferable that the modification should not influence the ongoing application services on the connection. To assure this, an effective approach is to establish a new connection first, which satisfies all the connection requirements, and then switch the services from the old connection to the new connection, and finally remove the old connection. This approach always ensures the service continuity.

Finally, the function of connection query is mainly to query the status or some parameters of the connection. Similarly, the function of connection release is to tear down a connection, which is simple enough to merely release the consumed network resources by sending a release message from the source node and responding with a confirmation message from the destination node.

C) Link Management Protocol (LMP)

Link Management Protocol (LMP) is a new protocol specifically developed for the GMPLS control infrastructure [Lang03]. In traditional networks, such as the IP network or the SONET/SDH network, the data plane and the control plane always coexist in a common physical network and the control overhead is often piggybacked or encapsulated in a packet or a frame, or multiplexed with user data traffic. For example, in the IP

network, control packets such as "hello" messages are multiplexed with the user data in the same data channel. In the SONET/SDH network, the control overhead piggybacks the user data encapsulated in the header of a frame. The benefit of such coexistence is that in addition to realizing all the control functions, the control system can monitor the health of a data channel. In the IP network, the "hello" message is used to monitor the healthy status of incident spans and neighboring nodes. Similarly, in the SONET/SDH network, K1/K2 bits in each frame overhead are used to detect failure of the span. Due to the low transmission capacity in traditional networks (typically, each fiber transmits only a single or a few wavelengths), no scalability issues need to address for the network control system. However, in the new generation of transport networks such as the WDM networks, which have up to thousands of wavelengths multiplexed in each fiber, the scalability of control system will become a critical issue if the control traffic were transmitted in the same way as in the traditional networks. It would be uncontrollable, as there could be up to thousands of control channels on each network span if each data channel corresponds to one control channel. For a larger-scaled network, it will be a heavy burden for the control system to maintain such a large number of control channels. On the other hand, for a network such as the WDM networks, the data channels transmitted in each span usually share some common features and properties, which triggered the idea to aggregate these features and properties by using a common set of parameters and to send this aggregated information only via a single control channel. This is simply a *link bundling* process as discussed previously.

To overcome the difficulty of scalability and take advantage of the common features of data channels, the GMPLS control architecture proposes to divide a network into a control plane and a data plane. The two planes can be independent enough to exist in two different physical networks. The data plane is responsible for user data transmission, while the control plane is responsible for control data exchange. Between the two planes, there are some interfaces that allow the control plane to send control messages to the data planes and the data plane to send enquiries to the control plane. With such a separation, a single control channel is sufficient to carry control messages for multiple data channels, which significantly offloads the burden of the control system and consequently improves its scalability. Despite the above benefits, the new architecture suffers from the

concomitant difficulty that, since they are separated, the control channels cannot monitor the healthy status for the data channels as before. For this, alternative failure detection mechanisms are required. LMP was specifically developed for this purpose to support the following four major functions: (1) control channel management, (2) link property correlation, (3) link verification, and (4) fault management [Lang03].

The function of control channel management is to establish, configure, and maintain connectivity between adjacent GMPLS controllers in the control plane. Link property correlation summarizes or abstracts multiple data component links into a single TE link based on their common properties. For example, a wavelength on a fiber can be abstracted as one unit of network capacity by the control system, which is the same for every wavelength on the span even though each wavelength is physically different in the fiber-optic transmission system. Thus, a span with N such wavelengths can be summarized as N units of capacity. This is just a *link bundling* process. Based on the link property correlation function, only a single control channel is needed to exchange the aggregated information for all the data channels. Likewise, the function of link verification is to verify the physical connectivity of the data links and to exchange each data link ID. With this function, a unique IP address is not mandatory for each interface or element. Rather, a combination of an IP address and a data link ID can globally identify each interface or element, in which the IP address is assigned to a switched node, and the data link ID is locally assigned for an interface or element of the switch. The component link index-matching table under the concept of *unnumbered links* is maintained by this function. Finally, in collaboration with the link verification function, the fault management function can suppress alarms and localize failures of networks. These two functions can compensate for the loss of fault detection capability after separating the control plane from the data plane.

D) Service Connection Database

Once a connection is established, we need to record information about the connection. A connection database at each node stores all the information related to the service connections. Detailed information on each connection can include the hop information of the connection (which can be a sequence of link IDs), protection priority of the connection (which can be best effort, spare capacity sharing protection, or dedicated

protection), capacity or bandwidth of the connection, etc. This connection information is required when a connection is queried or released.

E) Operational Overview

After describing the control components and related items, in the following we briefly review the overall operation of a GMPLS-controlled network, which mainly involves the processes of connection establishment and release.

Connection establishment: If a new connection request arrives at a node, which consists of the information on source and destination nodes, required network bandwidth, QoP level, and some policy-based constraints, the routing component of the node first looks up the network state database to compute an eligible route that meets the constraints. A CSPF routing algorithm is used to search an optimal route. If the route is found, the signaling component will then set up the connection by sending a PATH message along the route to the destination node, which collects and verifies the information on network resources during the forwarding. If the destination node receives the PATH message and finds that there are sufficient resources on the route and all the constraints are met, then it sends a RESV message back to the source node. When the RESV message retraces back to the source node, it makes reservations for network resources and sets up the switch status for intermediate switching nodes to prepare for the connection establishment. After the RESV message reaches the source node, which means that the connection has been established successfully, the source node can transmit application data on the connection.

After the connection is established, all the information on the connection should be stored in the service connection databases of the source and destination nodes. Meanwhile, LSA messages should be flooded around the network to update the network state database at each node, for some network capacity has been used after the connection was established.

Alternatively, the network may fail to establish a new connection due to lack of network resources or because some constraints cannot be satisfied. In this case, the connection request must be blocked or delayed for a while before making another attempt.

Connection release: The process of connection release is much simpler than establishment. When a connection completes its service, the source node will initiate a release signaling process by sending a release message to the destination node. Upon receiving the release message, the destination node responds with a confirmation message to confirm the success of the release. All the network resources used by the released connection are returned to the network for future use. Meanwhile, a new round of LSA message should be flooded to update the changes of network link state. Also, all the information related to the released connection should be deleted from the connection databases.

F) Impact of Unsynchronized Network State Information

Under the GMPLS routing protocols, when a local node computes routes for service connections, it assumes that it has a fully-synchronized network state database. Consequently, any successful finding of a route by the computation process assumes that the network has sufficient resources to accommodate the new service connection. Nonetheless, we know that in the OSPF/IS-IS routing protocols, the network state database cannot ensure 100% synchronization every time because the LSA flooding process always takes time to update the network state databases whenever there is any change in the network. Thus, it is possible to encounter situations as follows: before the database is fully synchronized, a local node may find a route that has sufficient network resources falsely based on *stale* network state information. As a result, when the signaling process attempts to establish the connection, it will be declined for the simple reason that there are insufficient real network resources on some spans, and hence the connection request has to be blocked. Despite some impact, unsynchronized network state database is not a primary aspect to cause network service blocking. Thus, in the following study, we assume that when the routing component searches for a new route, it always has an updated network state database.

G) Resource Contention between Connections

Besides the impact of unsynchronized network state database, resource contention may also contribute to service blocking. It is possible that there are multiple connection requests arriving at different nodes concurrently. Upon receiving the requests, each of the nodes computes the route for its own connection independently. If the found routes are

not assigned the same network resources, then the signaling processes can establish each of them successfully. However, if the routes are unintentionally computed to use the same network resources, a resource contention is formed, for only one of them can actually use the resources. To resolve the contention, several mechanisms have been proposed. For example, one of them is to compare the source node IP addresses of all the involved connections and then to select the one with the highest or lowest IP address to win the contention and use the resources, leaving all the other connections that lose the contention blocked. Besides IP address, it is also possible to use other metrics ranging from service priorities to required bandwidth of connections to resolve the contention. However, for simplicity without losing the effectiveness of the study, we assume that the impact due to the resource contention between connections is ignored when evaluating performance for dynamic service provisioning.

5.2. Shared Backup Path Protection (SBPP)

In Chapter 3, we briefly introduced the concept of SBPP. SBPP is a path-oriented survivable technique that can be perceived as a special case of the path restoration. Nevertheless, in contrast to the latter, SBPP allows only a single protection path to recover the traffic on the affected working path, which means that no flow splitting is allowed when carrying out the restoration. For better spare capacity efficiency, SBPP also allows protection paths to share spare capacity on the common spans so long as their corresponding working paths maintain some disjointness. Depending on the levels of QoP, the disjointness can be span, node, or generally SRLG disjointness.

5.2.1. Various Types of Disjointness

The span-based disjointness is the lowest level of disjointness that requires the two paths not share any common span. With this disjointness, SBPP is capable of recovering any single-span failure, but does not ensure recovery of any service loss due to a node failure. Similarly, the node disjointness requires the two paths not share any common node except the source and destination nodes of the paths. The node disjointness generally shows a higher-level constraint of disjointness than the span disjointness, for the node disjointness can implicitly ensure the two paths to be span disjoint as well. That is, the node disjointness can assure the recovery of a single span or node failure. Finally,

the SRLG disjointness is the highest and also the most general level of disjointness that guarantees services to survive any single-SRLG failure, and is general enough to model both span and node disjointness plus some more complicated disjointness scenarios. For example, a fiber cable conduit containing a bundle of fiber cables can be regarded as an SRLG, for the damage of the conduit may sever all the cables. Also, a network central office, which contains many switching nodes, can be regarded as an SRLG as well, for a power-off due to a fire or some other reason can turn off all the switches in the office. Figure 5.4 illustrates examples for the above three types of disjointness under the SBPP context.

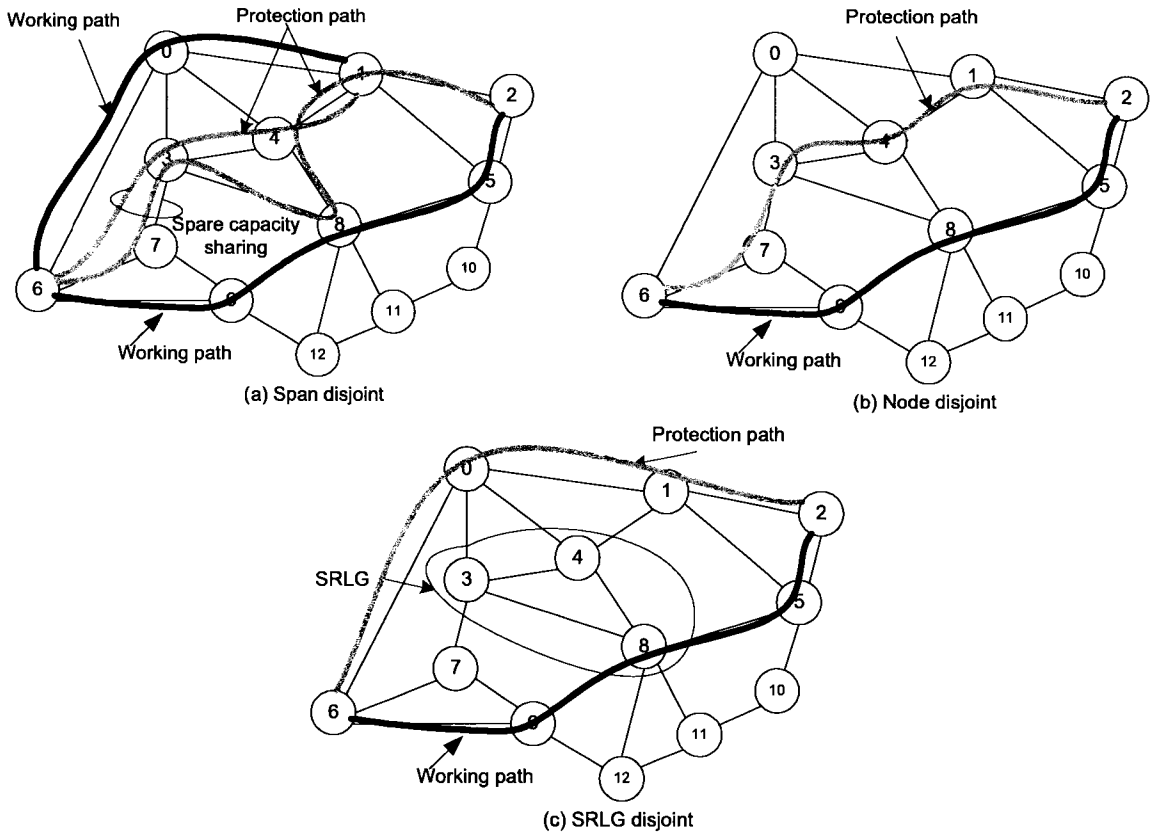


Figure 5.4. Examples of SBPP Disjointness.

Figure 5.4(a) shows examples of span disjointness, where a pair of working and protection paths is set up between node pairs (2, 6) and (1, 6) respectively. The working and protection paths between each node pair do not share any common span on the routes, but they are allowed to share common intermediate nodes under the definition of span disjointness. For example, the working and protection paths between node pair (2, 6)

share a common node, namely node 8. Moreover, the protection paths are allowed to share spare capacity on common spans as long as their corresponding working paths do not traverse any common span. The protection paths and working paths between node pairs (2, 6) and (1, 6) meet the above condition. Thus, their protection paths can share the spare capacity on their common spans (6-7), (3-7), and (1-4). The next example is on the node disjointness, which is shown in Figure 5.4(b). There is also a pair of working and protection paths established between node pair (2, 6), but the two paths do not share any common spans and nodes except the two sink nodes 2 and 6. Finally, the example on the SRLG disjointness is illustrated in Figure 5.4(c), where an SRLG is made up of three neighboring nodes (i.e., nodes 3, 4, and 8) and their interconnected spans. The SRLG is highlighted in the figure by a circle. The working path crosses the SRLG to transit one of its nodes (node 8). To ensure the SRLG disjointness, the protection path is required not to traverse any component of the SRLG. As such, route (6-0-1-2) is selected as the protection path in the example. The spare capacity sharing of the last two types of disjointness is very similar to that of the span disjointness. Consequently, examples are not provided for them. In this thesis, we have considered the span-disjointness for SBPP, but this does not affect the generality of the approaches herein to study the other types of disjointness.

5.2.2. Literature Survey on SBPP

SBPP has been widely studied due to efficient spare capacity sharing and flexibility in service provisioning. The SBPP idea was first proposed for the MPLS network in [KiKo00], although the previously-proposed scheme of backup Virtual Path (VP) protection in Asynchronous Transmission Mode (ATM) was logically the same scheme [KaSa94]. Its performance was evaluated in [KoLa00] to consider various cases including no spare capacity sharing (i.e., 1+1), spare capacity sharing with partial routing information, and spare capacity sharing with full routing information. The challenge of how to distribute spare capacity sharing information around the network was investigated as well. Besides spare capacity sharing between the protection lightpaths, a new scheme allowing working lightpaths and protection lightpaths to share capacity was proposed in [MoMu01]. Such sharing is somewhat similar to the concept of "stub release" or "stub reuse" proposed in [IrMa98] [Grov04b], where the protection lightpath is allowed to use

stub capacity released from the working lightpaths that incur a failure. Based on the work in [KoLa00], the network performance given some partial spare capacity sharing information was further investigated and a new routing algorithm, which could outperform the heuristic in [KoLa00], was proposed in [QiXu02]. In [BoLa02] a statistical approach was applied to compute shared mesh restored lightpaths, and its performance was compared to that of the deterministic approach. It was found that although the statistical approach needed much less network state information it could nonetheless perform close to the deterministic approach, where a complete list of capacity-sharing information was presented. Also based on the work in [KoLa00], [LiWa02] re-evaluated the network performance by arguing that it was not difficult for each node to obtain the full information on the spare capacity sharing with mature routing and signaling protocols. The results demonstrated that the method with full information restoration (FIR) could achieve a much better performance in terms of capacity efficiency than the method with partial information restoration (PIR) proposed in [KoLa00]. In addition, to improve the restoration speed of the shared backup path protection method, a fast restoration method was proposed in [HaKo02] by pre-configuring the switching states of the protection lightpaths so as to rapidly switchover the traffic on the working lightpaths upon a failure. The idea is similar in that regard to *p*-cycles [GrSt98a], where *p*-cycles are pre-configured to offer one protection path for an on-cycle span failure and two protection paths for a straddling span failure, but where there is no need to re-configure the switch states on the protection routes after a failure occurs. In terms of spare capacity efficiency, the method in [HaKo02] however may not be as efficient as *p*-cycles. Also by assuming that each node in the network has the full information on spare capacity sharing, to avoid trap topologies within a network, a simple but efficient heuristic was proposed in [OuZh03] to route SBPP lightpaths. It was found that the proposed heuristic could significantly outperform the FIR proposed in [LiWa02] under a sparse network topology. A comprehensive study to compare performances in terms of lightpath blocking probability was carried out in [YaZe03] to evaluate the various survivability schemes including 1+1, SBPP, link protection, link restoration, and other schemes. The results showed that overall the path-based survivability schemes performed better than the link-based schemes, and the spare capacity sharing schemes always

outperformed the dedicated schemes. More recently, a comprehensive survey specifically for SBPP-based survivable service provisioning was carried out in [ShGr05d] to compare various SBPP provisioning approaches in terms of blocking performance and operational complexity. The tradeoff between routing information maintenance and blocking performance was identified.

5.3. GMPLS-BASED SBPP SERVICE PROVISIONING

In this section we will apply GMPLS to provision SBPP-based survivable services. We first present the basic operations of service provisioning under the SBPP survivable technique. Following this, we elaborate the detailed network state database for SBPP service provisioning. Finally, we describe the routing algorithm applied to search working and protection routes, in which the issue on spare capacity sharing is considered.

5.3.1. BASIC OPERATIONS

Service establishment: As before, the service establishment process begins with a service request at a local node, which bears information on the two end nodes of the connection, required network bandwidth, and some policy-based constraints. The local node will look up the network state database to compute a working route and a protection route. According to SBPP, the working route and the protection route are required to maintain some disjointness, typically link-disjoint. Moreover, it is desirable that the protection path should maximally share spare capacity in the networks. Specifically, the spare capacity on the route can always be shared so long as all the other working paths protected by the spare capacity do not share any common span with the current protected working path. Obviously, selection of a pair of working and protection routes that meet the above constraints is complex. The details on the routing algorithm applied in our study will be provided in Section 5.3.3.

If the working and protection routes are found, signaling processes will be triggered to establish the working and protection paths. A combination of PATH and RESV messages will be sent in each signaling process. Based on the working and protection path establishment sequence, there are two possible strategies to apply. One is *concurrent strategy*, and the other *sequential strategy*. The concurrent strategy establishes the working and protection paths in parallel. Two signaling sessions are initiated at the same

time, with one for the working path and the other for the protection path. Only if both signaling sessions succeed can the survivable service be provisioned; otherwise, the provisioning process fails. In contrast, the sequential strategy establishes the working and protection paths one by one. The working path is usually established first, followed by the protection path, and only when the working path is established successfully, can the protection path begin to be set up. If the working path fails to be established, the establishment process for the protection path would not be initiated. Only when both paths are established can the survivable service be provisioned.

Upon a successful service provisioning, two network-state updating processes are required. One is to update the connection database, and the other to update the network state database, which is mainly concerned with the changes of network link state and spare capacity sharing. In particular, because the protection route searching process needs to consider the relationship of spare capacity sharing among all connections, the connection database should be *globally* updated around the network. That is, the connection database at each node needs to record the connection information of each new provisioned service.

Service release: The service release process frees the network resources used by the working path and those solely used by the protection path, with a pair of release and confirmation messages. Here "solely" means that the protection path being released is the unique one using the spare capacity unit, so the release of this spare capacity will not affect other protection paths in the network. Upon release, two updates are necessary for the network. One is to clear the record of the released connection from all the connection databases; the other is to update the network state database to notify that more network resources are available for future use and some spare capacity sharing relationship has been changed.

5.3.2. Link State Database and Connection Database

Figure 5.5 displays the possible structures of the link state database and the connection database for the SBPP-based service provisioning. The link state database is used to record all the network state information on the links: (1) deployed link capacity, (2) used link capacity, (3) working capacity and spare capacity on the link, and (4) the relationship

of spare capacity sharing. The connection database is used to store the information on all the established service connections, which includes (1) node addresses of the two end nodes, (2) required capacity, and (3) the routes and used capacity of the working and protection paths. In summary, these two databases record the link state information ranging from *how many capacity units are available on each span* (i.e., the field of "free capacity in channel" in the link state database) to the information related to protection, which includes the information on *which working path is protected by which protection path* (i.e., the fields of "working path" and "protection path" in the connection database) and *which spare capacity unit is being shared by which group of protection paths* (i.e., the table of "protection channel sharing relationship" in the link state database).

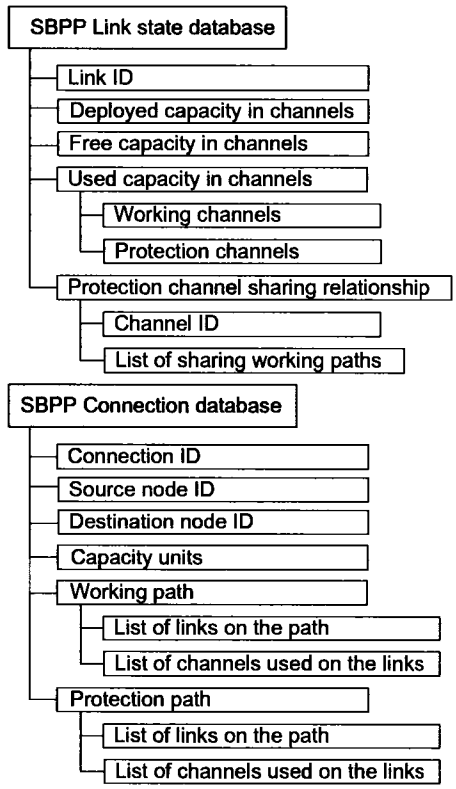


Figure 5.5. Link State Database and Connection Database of GMPLS-Based SBPP Service Provisioning.

5.3.3. Routing Algorithm of SBPP Service Provisioning

SBPP service provisioning establishes a working path and a protection path for each survivable service connection. Thus, the routing algorithm of service provisioning needs to search a pair of working and protection routes for each service request. Typically the

protection route must be link-disjoint from the working route. Also, in order to maximize spare capacity sharing, the protection route should be selected from the candidates, which can maximally share the reserved spare capacity. For this, the algorithm must consider path protection relationship (i.e., which working path is protected by which protection path), and sharing information of each protection capacity unit (i.e., which working paths are protected by which protection capacity unit, or which protection paths are sharing which protection capacity unit). These are challenging issues when developing the routing algorithm.

In our study, we have employed an all-distinct route searching algorithm to find the eligible routes in an ascending order of hop-length between a node pair, and then from these routes, search a working and protection route pair for each SBPP service. Specifically, in a first-fit fashion, from all possible combinations of pairs of routes, we find the first pair that satisfy the following two conditions: (1) the two routes in the pair are link disjoint from each other, (2) the current available network resources are sufficient to establish both of them, with one functioning as a working path and the other as a protection path. The searching process is terminated when the first eligible pair of working and protection routes is obtained. This algorithm is similar to the *deterministic algorithm* in [BoLa02]. It is called *First Fit* (FF) algorithm, for the search process is terminated when the first eligible pair of working and protection routes is found. The details of algorithm pseudocode specifically developed for this study are presented as follows:

1. Use an all-distinct route searching algorithm to find a complete list of distinct routes $\{R_i, i = 1, \cdots, n\}$ between a node pair in an ascending order of hop-length.
2. Set a flag *f:=false*.
3. *For* (all the routes in the route list: $R_i, i = 1, \cdots, n$)
   {
   Select R_i as a candidate working route and based on the current network resources, examine if there is enough capacity to establish a working path via route R_i;
   If(*true*)
   {
   For (all the routes in the route list: $R_j, j = 1, \cdots, n$)
   {
   Examine if R_j is link-disjoint from the working route R_i;
   If(*true*)

{
Based on the current network resources (including free resources and shared resources), examine if there is enough capacity to establish a backup path via route R_j;
If(*true*)
{
Establish the working and backup paths on the routes R_i and R_j, respectively; the working path consumes one capacity unit on each hop. The backup path shares existing protection capacity on each hop en route wherever possible. If on any span, no protection capacity can be shared, a new unit of unused capacity should be reserved as protection capacity.
Update the network resource status;
Set the flag *f*:=*true*;
Break the internal *for* loop.
}
}
}
}
Check flag *f* to see if the working and backup paths can be established successfully when R_i is selected as the working route;
If(*f*=*true*)
Break the outer *for* loop.
}
4. *If*(*f*=*true*)
The survivable service request has been established successfully;
Else
The request has to be blocked.
End.

When selecting the protection route, the algorithm checks the protection resources on the route unit-by-unit to see whether any of them is sharable by the new protection path. If there is one, then the protection path shares it; otherwise, a new unit of spare capacity is assigned on the span specifically for the protection path. In addition, because the algorithm examines the routes from the shortest to the longest, the working route found is implicitly the shortest among the working routes of all the protection and working route pairs that satisfy conditions (1) and (2). The advantage of this is that the algorithm can consume as few network resources as possible for each survivable service provisioning. This is the case because in a network with spare capacity sharing, a working path normally requires more network resources than a protection path for the simple reason that the working path always consumes *real* capacity units, while the protection path can share as much existing spare capacity with other protection paths as possible. Thus, it is generally more economical to first keep the working path shortest, and on this basis to

establish the shortest protection path. Certainly, selecting the working and protection routes in the above fashion cannot guarantee that the sum of the hops of the two routes is the smallest. Actually it is possible to find a minimum cycle that traverses both end nodes, and the sum of hops of the two link-disjoint routes are the smallest by using the algorithm in [SuTa84]. However, the minimum cycle cannot guarantee that the working route is the shortest among all the eligible route-pair candidates. If the working route that the minimum cycle contains is the second or third shortest route, then more resources are actually needed compared to the method we proposed. Also, based on cycles it is more complicated to implement spare capacity sharing. As such, we do not employ the minimum-cycle search algorithm to find the working and protection route pair in our study. In addition, there may exist more advanced or efficient SBPP routing algorithms in the literature. However, because the FF algorithm is able to achieve a representative performance for SBPP, for simplicity and rapid simulation we have adopted this algorithm in our comparative study.

5.4. Closing Remarks

In this chapter, we have described the GMPLS control and operation architecture and its survivable service provisioning process. SBPP is currently a *de facto* approach to provisioning dynamic survivable services and demonstrates good flexibility and spare capacity efficiency. The approach is however complicated in the maintenance of network state and connection databases, and requires a complex route-searching algorithm. It is doubtful that the approach is scalable enough to provision services under a future frantically dynamic environment. Thus, further research is required to examine the current SBBP provisioning technique to improve its scalability. Alternatively, other more scalable provisioning approaches should be developed.

CHAPTER 6
FORCER CONCEPT AND ITS APPLICATIONS[5]

The forcer concept was first introduced for span-restorable mesh networks [GrLi99]. However, it is general enough to fit all types of span-based survivability techniques, which include span-protecting *p*-cycles [ShGr03b]. This chapter provides a detailed description of the forcer concept in the context of both span-restorable mesh and span-protecting *p*-cycle networks. A simple but efficient ILP-based forcer identification method is presented to identify forcers for various types of span-based survivability schemes. The potential applications of the forcer concept in survivable network designs and dynamic service provisioning are discussed.

6.1. FORCER CONCEPT IN SPAN-RESTORABLE MESH NETWORKS

The forcer concept was first observed and defined in span-restorable mesh networks. Figure 6.1 illustrates an example of forcers, in this figure there is a five-node network with working and protection capacity as indicated by each span. The working capacity was assigned based on the shortest path routing algorithm. The spare capacity was optimally determined based on the ILP optimization model for span restorable networks [HeBy95].

Now let us consider some span failure scenarios and see how the working capacities are restored by the spare capacities. For example, if span (0-1) fails, five units of working capacity on the span will be affected. To restore them, two protection paths, i.e., (0-3-1) and (0-3-4-2-1), will be used to recover three and two units of traffic flows respectively. Particularly, on span (0-3), the five units of spare capacity will be completely occupied. In another example, if span (1-3) fails, two units of working capacity on the span will be affected. There are two eligible restoration paths, i.e., (3-0-1) and (3-4-2-1), to restore the affected traffic. To be efficient, we may employ path (3-0-1) to restore the failure, which results in three units of spare capacity used on the path. Thus, three units of spare capacity will be consumed on span (0-3), but with two units of spare capacity remaining on the span. Based on the above two failure restoration examples, an important

[5] This chapter contains some material previously published in [ShGr03b].

observation is that it is span (0-1) rather than span (1-3) that "forces" span (0-3) to require five units of spare capacity. In fact, to restore the span failure (1-3), no spare capacity is required on span (0-3) if the longer restoration path (3-4-2-1) is used to restore the failure instead.

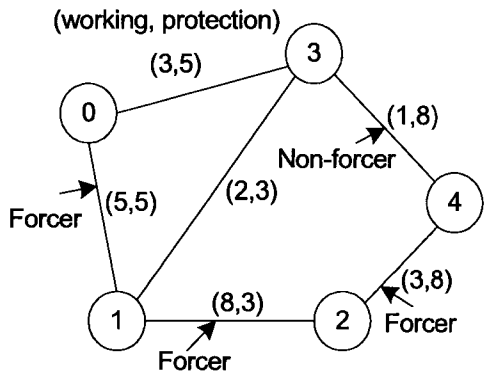


Figure 6.1. An Example of Forcer Concept in a Span Restorable Mesh Network.

The *forcer* concept was defined based on the above observation, which is always related to two spans. Span A is a forcer of span B if and only if among the whole network it is span A that forces span B to have certain amount of spare capacity in order to ensure the full restoration of span A if it fails. This means that if the assigned spare capacity on span B is a little less, then the full restoration of span A cannot be guaranteed. Equivalently, if there is any increase of working capacity on span A, the spare capacity on span B must be increased as well in order to fully restore span A. In the previous example, it is readily seen that span (0-1) is a forcer span of span (0-3), while span (1-3) is a *non-forcer* of span (0-3). The original definition for the forcer concept within a span-restorable mesh network given in [GrLi99] is re-stated as follows:

"A forcer span is any span for which an increase in network total sparing is required (to retain restorability) if the span's working capacity is increased."

Obviously, the term "forcer of a span" is interchangeable with the term "forcer of a network" since the increase of working capacity on the forcer span can cause both the span and correspondingly the network to increase spare capacity simultaneously for the full restoration.

There can be more than one forcer for the same span. These forcer spans are called *co-forcers*. In Figure 6.1, we can see that besides span (0-1), span (1-2) is also a forcer of span (0-3), because the failure of span (1-2) needs the restoration path (1-3-4-2) to recover three units of flow and the path (1-0-3-4-2) to recover five units of flow, which will use up all the protection capacity on span (0-3). Thus, spans (0-1) and (1-2) are *co-forcers* of span (0-3).

All the forcers in a network make up a *forcer skeleton*. Such a skeleton is sufficient to yield the entire spare capacity plan, which means that even if all the working capacity of those non-forcers are set to zero, the same network spare capacity would be planned in order to fully restore the failures of these forcer spans. In Figure 6.1, there are a total of three forcer spans, i.e., spans (0-1), (1-2), and (2-4). They have formed a forcer skeleton that contains sufficient information to determine the total required spare capacity on each span.

6.2. FORCER CONCEPT IN SPAN-PROTECTING *P*-CYCLE NETWORKS

The forcer concept is also applicable to *span-protecting* p-cycles. However, in contrast to the span restorable network, where the working capacity directly "forces" the spare capacity, the forcing relationship in the context of span-protecting p-cycles is indirect. Between the working capacity and the spare capacity, there is an intermediate p-cycle layer, which functions as a *bridge* to connect the forcing relationship between working and spare capacities. Using the same network example as shown in Figure 6.1, we design the span-protecting p-cycle network and discover the forcing relationship between the working and spare capacities as shown in Figure 6.2.

The required spare capacity of the span-protecting p-cycle network is almost the same as that of the span-restorable network, except that span (1-2) has eight instead of three units of spare capacity. This verifies the prior conclusion that the p-cycle technique can achieve a mesh-like spare capacity efficiency [GrSt98a]. In the network, there are two p-cycles that include the larger one, cycle (0-1-2-4-3-0), and the smaller one, cycle (1-2-4-3-1). The two cycles are assigned five and three units of spare capacity respectively. With these spare capacities the p-cycles can guarantee full restoration of any single-span failure in the network.

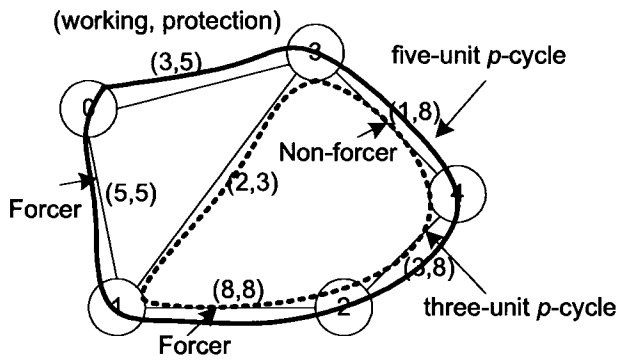


Figure 6.2. An Example of Forcer Concept in a Span-Protecting *p*-Cycle Network.

As in the previous span-restorable network, we again assume that spans (0-1) and (1-3) are the failed spans. If span (0-1) fails, it will be restored by the larger *p*-cycle in a fashion of on-cycle span failure restoration. To fully restore it, the spare capacity on the larger cycle will be totally used up. Thus, we can conclude that it is span (0-1) that forces the larger cycle to have this amount of spare capacity, which forms a forcing relationship between the span and the cycle. Namely, span (0-1) is a *forcer* of the larger cycle. Now let us look at the other span, span (1-3), which is an on-cycle span of the smaller cycle, and meanwhile a straddling span of the larger cycle. Since the affected working capacity on the span is only two units if it fails, and the two *p*-cycles are assigned five and three units of spare capacity respectively, the spare capacity of either *p*-cycle is over-provisioned for the restoration of the span. Consequently, span (1-3) does not form any forcing relationship with any of the *p*-cycles. Even if one of *p*-cycles were removed from the network, the failure of span (1-3) could still be fully restored by the remaining *p*-cycle. Thus, span (1-3) is a *non-forcer* for either *p*-cycle.

As indicated, *p*-cycles function as a bridge for the forcing relationship between the working and spare capacities. The forcing relationship between the spans and the *p*-cycles can be finally transformed to the relationship between the working and spare capacities. To make a transformation, the next step is to consider the relationship between the *p*-cycles and the spare capacity. Let us examine Figure 6.2 again. To set up the *p*-cycles, enough spare capacity should be assigned on each span. The relationship between the spare capacity on each span and the *p*-cycles is simply a topological mapping between the spare capacities on the spans and the *p*-cycles. For the larger *p*-cycle, all of its on-cycle spans should be assigned five units of spare capacity. Therefore, spans (0-1) and (0-

3) are assigned five units of spare capacities respectively. Because there is only a single *p*-cycle traversing the spans, the five units of spare capacity are the total required spare capacities on the spans. As for the other spans, including spans (1-2), (2-4), and (3-4), they are traversed by the smaller *p*-cycle as well. Consequently, the total spare capacities on these spans should be the sum of the spare capacities required by the two cycles, which are a total of eight units of spare capacity as shown. Similarly, because span (1-3) is only traversed by the smaller *p*-cycle, it needs only three units of spare capacity. Based on this mapping process and the previous forcing relationship between the spans and the *p*-cycles, we can discover the forcing relationships among the spans. The detailed rule is given as follows, which essentially is the forcer concept for the *p*-cycle networks:

Span A is a forcer of span B if and only if span A forms forcing relationships with all the *p*-cycles that traverse span B; otherwise, span A is a non-forcer of span B.

Based on the above definition, we can determine all the forcers in the network as shown in Figure 6.2. Specifically, span (0-1) forms a forcing relationship with the larger *p*-cycle, and the larger *p*-cycle is the unique cycle that traverses span (0-3). Therefore, span (0-1) is a forcer of span (0-3). Similarly, span (1-2) forms forcing relationships with both the larger and smaller *p*-cycles. Therefore, it is a forcer for all the other spans in the network. It should be noted that in contrast to the span-restorable network, in which there are three forcers, here in the *p*-cycle network, there are only two forcers, i.e., spans (0-1) and (1-2). This difference is ascribed to the fact that more spare capacity (i.e., eight instead of three units of spare capacity) has been assigned to span (1-2), which makes span (2-4) a non-forcer. Finally, in the *p*-cycle network, there also exist the concepts of co-forcer and force skeleton. Specifically, spans (0-1) and (1-2) are the co-forcers of span (0-3), and form the forcer skeleton of the network.

6.3. FORCER IDENTIFICATION METHODS

In the previous examples, we have manually identified the forcer, which works well due to the simple network topology. However, for a large-scaled network, with tens or even hundreds of eligible restoration routes between each node pair, it is impossible to identify the forcers simply based on manual examination. Rather, more efficient methods are desired.

In this section, we will present four forcer identification methods. The first two forcer-identification methods were proposed in [GrLi99] [GrMa00] for span-restorable mesh networks. Although the methods are also applicable to *p*-cycle networks, they are found to be complex and inefficient. This motivates us to find simpler and more efficient methods, which are the other two methods to be introduced in this section.

6.3.1. KSP-Based Identification Method

The *k*-shortest path routing-based forcer identification method was first developed to identify the forcers for the span-restorable network [GrLi99]. Under this method, a single-shortest path routing algorithm, e.g., Dijkstra's algorithm, is applied to find working routes and corresponding working capacity on each span. Next, each span is tested as a failed span one at a time, and the corresponding set of restoration paths for the failed span are found by the KSP route-finder. For each span on a restoration path, the required spare capacity for the restoration of the tested span is then identified. Let s_{ij} denote the number of spare capacity units required on span j for the restoration of span failure i. We then compare s_{ij} with the total assigned spare capacity on span j, s_j, which has been assigned in the stage of network planning. If $s_{ij} < s_j$, then span i is not a forcer of span j; otherwise it is. As long as there is at least one forcing relationship between span i and any other span in the network, span i is a forcer span of the network. Otherwise, it is a non-forcer. After all the spans in the network are tested, a forcer skeleton can be identified. Although the above method was initially proposed for the span-restorable network, it can be extended to identify the forcers for a span-protecting *p*-cycle network as well. Nonetheless, it can be foreseen to be much more complicated as the forcing relationship between spans is bridged by an intermediate *p*-cycle layer.

6.3.2. Multi-Round Retry Identification Method

The other forcer identification method employed a repeated process to run multiple rounds of ILP optimization for spare capacity allocation to identify forcers [GrMa00]. The details of the approach are as follows. Based on a given network topology and working capacity quantities on spans, an ILP optimization model is first run to find the minimal feasible total network spare capacity cost. Based on this, one span is then

selected from the network for a test run. A single extra working capacity unit is increased for the span, and the optimization process is rerun to determine whether any increase in the total spare capacity cost is required. If there is, then the span is considered as a forcer; otherwise it is not. Following a return of the first test condition to normal, another span is selected from the network for the same test as described. The process is terminated when all the spans have been tested. Based on this method, for a network with M spans, M such tests must be performed. For a large-scaled network, the above method can be time-consuming.

Moreover, if we want to know how many extra working channels can be raised before a non-forcer becomes a forcer, the task can be even more tedious [ShGr03b]. The simplest way is to add a single working capacity unit one by one to the span until the span becomes a forcer. This requires the test to have a complexity of $O(n)$. A more efficient approach is to apply the method of binary section by adding an amount of channels, which is half of that of the previous test. This can efficiently reduce the testing complexity to be $O(\log n)$. For both, it is very difficult to predict when a span will become a forcer, so the total testing rounds of the whole network are also unpredictable. However, we can at least foresee that it is much larger than M rounds for an M-span network.

The above method can be extended to identify forcers in a span-protecting p-cycle network. However, its inherent multiple-retry feature can be a fatal limitation to the method.

6.3.3. One-Run Identification Method

In view of the difficulties in the above two identification methods, a more effective optimization-based approach for forcer identification is now proposed. It requires only one run of the optimization model to detect and quantify all the forcers associated with a given spare capacity distribution [ShGr03b]. That is why we term it *one-run identification method*. The method can identify not only all the forcers within the network, but also all the extra working capacity units that can be raised onto the non-forcer spans before they become forcers as well.

The detailed steps of the method are as follows:

Step 1. Use the conventional span or p-cycle restorable ILP models to design the network and determine the minimum required spare capacity on each span to guarantee full restoration for any span failure.

Step 2. Based on the optimization results, list the required spare capacity on each span. Use the spare capacity as a budget to run an ILP optimization model (presented below) to finally identify all the forcer spans as well as how many units of extra working capacity can be added on the non-forcer spans before they become forcers as well.

In the context of the p-cycle network, the ILP model used to identify forcers is presented below. In addition to the sets, parameters and variables defined in Section 4.1.2, a new variable is required

- e_k denotes the number of extra working capacity units that can be served on span k without increasing the current spare capacity budget on any other spans.

Objective: maximize $\sum_{k \in S} e_k$ (6-1)

Subject to:

$$\sum_{j \in P} X_i^j \cdot n_j \geq w_i + e_i \quad \forall i \in S \tag{6-2}$$

$$s_k \geq \sum_{j \in P} P_k^j \cdot n_j \quad \forall k \in S \tag{6-3}$$

The objective is to maximize the total number of extra channels that it is possible to serve within the predefined spare capacity budget. In the context of this problem it is important to note that both w_k and s_k are given as *inputs*. In other words, they are properties of some *existing* design on which we are performing the forcer analysis. w_k is the number of working capacity units on span k, which is obtained based on some routing procedure such as shortest path routing. s_k is the number of predefined spare capacity units on span k, which is obtained from the conventional span-protecting p-cycle design model to protect full restoration of w_k within the network. n_j is still a variable in the new model. The set of p-cycles associated with the initial design are disregarded, however,

and new n_j values are solved for that correspond to the maximal extra working capacity e_k for the given spare capacity environment.

Constraint (6-2) says that all existing working capacity on span i upon a cut should be fully restored by p-cycles, in addition to any extra working channels that it would be possible to restore. Constraint (6-3) guarantees that the p-cycle set chosen for (6-2) does not require more spare capacity than there actually is in the existing design on span k.

The solution of the maximization problem can determine all the extra working capacity values e_k on the spans. According to these values, we can further discover forcers and non-forcers. All the spans with e_k equal to zero are forcers of the original network design. According to the forcer definition, any increase of working capacity on a forcer span leads to an increase of total spare capacity in the network. e_k=0 means that no extra working capacity is allowed to be added under the predefined spare capacity budget, which can therefore judge that the span is a forcer. The spans with non-zero e_k are non-forcers. e_k reflects the depth of the span below forcer status (we call this its non-forcer magnitude or forcing margin). In this case e_k is a direct measurement of the number of extra working channels that could be turned-up for service on span k before having any impact on the spare capacity requirement of the network as a whole (to retain 100% restorability). For better understanding on the proposed forcer identification method, Figure 6.3 visualizes the steps of the proposed forcer identification method.

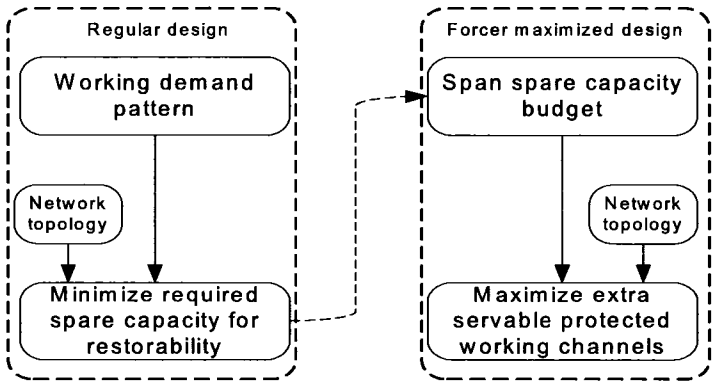


Figure 6.3. Steps of One-Run Forcer Identification Method.

Note that although we use p-cycle network as an example to present the proposed forcer identification method, the method is generic enough to identify forcers for any

span-based survivability schemes. Moreover, as a further extension, it is possible to use the method to identify forcers for the path-based survivability schemes ranging from path restoration to SBPP and others.

Compared to the previous two forcer-identification methods, the one-run method provides the benefits of simplicity, good efficiency, and generality. In the other two methods, tests are run for the spans one by one to determine whether or not they are forcers. Moreover, to quantify how much additional working capacity can be added on a non-forcer span before it becomes a forcer, the retesting process would be more complex and tedious. In contrast, the one-run method simply takes the test for the whole network only once, which can not only discover all the forcers within the network, but also find the quantities of extra working capacity that can be added before a non-forcer becomes a forcer. The comparison obviously allows us to conclude that the one-run method is much simpler and more efficient than the other two methods. As for the generality, thus far we are considering the forcer concept for span-based survivable networks, but it is possible to apply the proposed one-run method to identify the forcers for the path-based survivable networks as well. Conversely, it is almost impossible to use the other two methods to fulfill the same function in the context of the path-based survivability schemes. The details on how to extend the proposed one-run method to identify the forcers for the path-based survivable networks are beyond the scope of this study. For the details on the one-run method applied in the span-based survivable networks, interested readers are referred to our paper [ShGr03b].

6.3.4. Non-Forcer Filling Method

The non-forcer filling method is essentially the same as the previous one-run identification method. However, it provides more sensory information on how the forcers are identified and how the extra working capacity on the non-forcer spans is quantified. The procedure involves two steps. The first step focuses on the optimal survivable network designs based on the conventional survivable models such as span-restorable network and *p*-cycle network. The second step differs from the one-run identification method, but provides more sensory information helpful to understanding the forcer-identification process. It is called non-forcer filling step, which can be illustrated by the

same network example in Figure 6.2. The details of the identification procedure are shown in Figure 6.4.

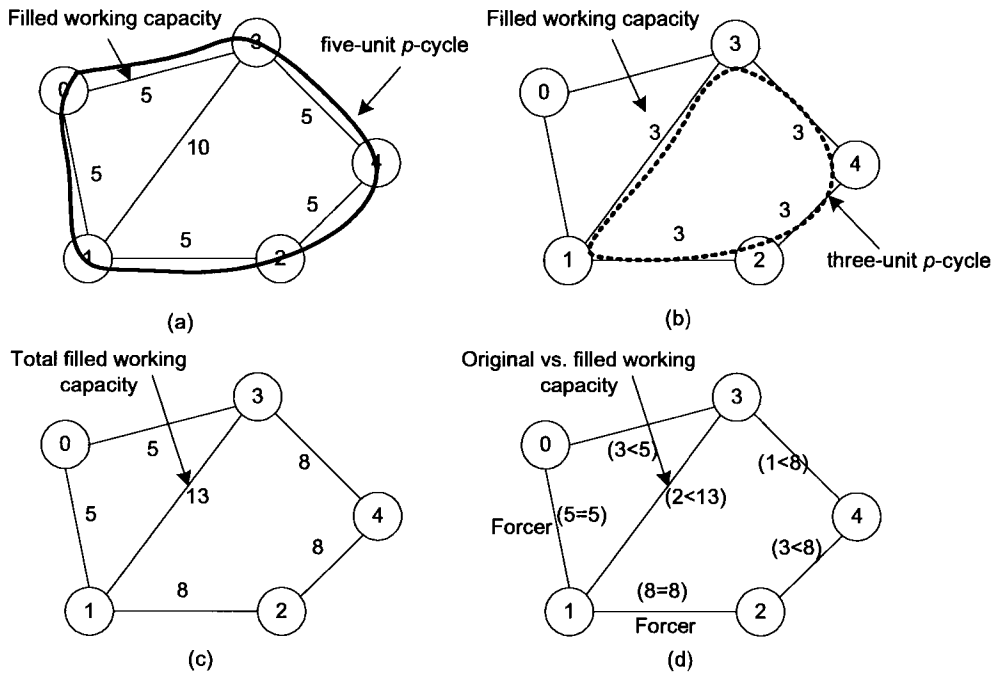


Figure 6.4. Non-Forcer Filling Procedure for a Span-Protecting *p*-Cycle Network.

Under the span-protecting *p*-cycle technique, Figures 6.4(a) and 6.4(b) try to fill the spans with the maximal protectable working capacity under the two pre-configure *p*-cycles. As indicated, the two *p*-cycles are assigned five and three units of spare capacity respectively. Therefore, for the larger cycle five units of working capacity can be filled onto each of its on-cycle spans, and ten units of working capacity can be filled onto each of the straddling spans. The filled working capacities are fully protected by the *p*-cycle. For example, span (0-1) is an on-cycle span, which can be filled with five units of working capacity. In contrast, span (1-3) is a straddling span, which can be filled with ten units of working capacity. These filled working capacities are fully protected by the cycle. A similar filling process can be carried out for the smaller *p*-cycle, as shown in Figure 6.4(b). We sum up all the filled protectable working capacities for each span, and obtain a network with total protectable working capacities as shown in Figure 6.4(c). The value adjacent to each span indicates the number of working capacity units that can be filled on the span, which is fully protected by the two cycles. Based on these total filled working capacities, we can determine the forcer spans and units of extra working capacities that

can be added before non-forcer spans become forcers spans, by comparing the current network design with the original working capacity distribution as shown in Figure 6.4(d). All the spans that have the same amounts of original and final filled working capacities are forcers, while the span whose original working capacity is smaller than the filled one is a non-forcer. The difference between the two working capacities is the exact number of extra working capacity that can be added before a non-forcer span becomes a forcer span. In the above example, we can see that spans (0-1) and (1-2) are forcers, while all the others are non-forcers. Specifically, up to eleven units of working capacity can be added on non-forcer span (1-3) before it becomes a forcer. Note that such a capacity addition does not require any spare capacity increase on the *p*-cycles or in the network.

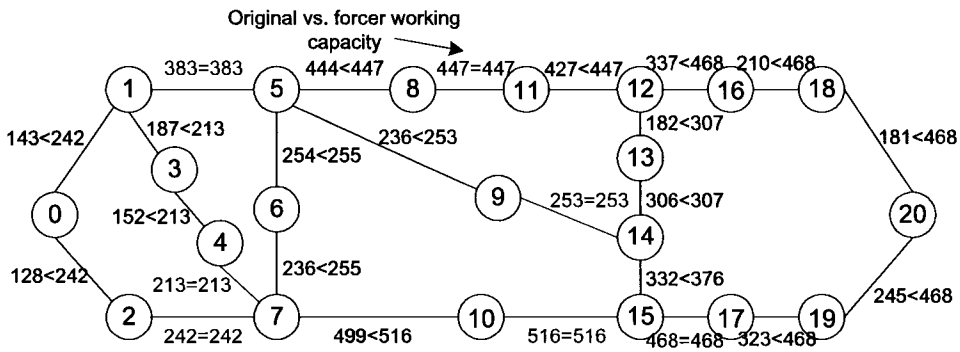
Like the previous one-run identification method, the non-forcer filling method is also applicable to other types of span-based survivable networks. In addition to span-protecting *p*-cycles, it can be used to identify forcers for the span-restorable network [HeBy95]. A brief description of how to use the non-forcer filling method to identify forcers for the span-restorable network is as follows.

In a span-restorable network with pre-planned working and spare capacities, we assume a certain span incurs a failure. Under the constraint of the pre-planned spare capacity budget, we can discover the max-flow between the two end nodes of the span. Note that when determining the max-flow, the span should not be included since it has been assumed to incur a failure. After the value of the max-flow is obtained, we can determine maximal protectable working capacity that can be filled on the span based on the current network design. By comparing the total filled working capacity and the pre-planned working capacity, we can then determine whether the span is a forcer or non-forcer. If they are equal, then the span is a forcer; otherwise, it is a non-forcer and the difference between the two capacities is the exact number of working capacity units that can be added before the span become a forcer as well.

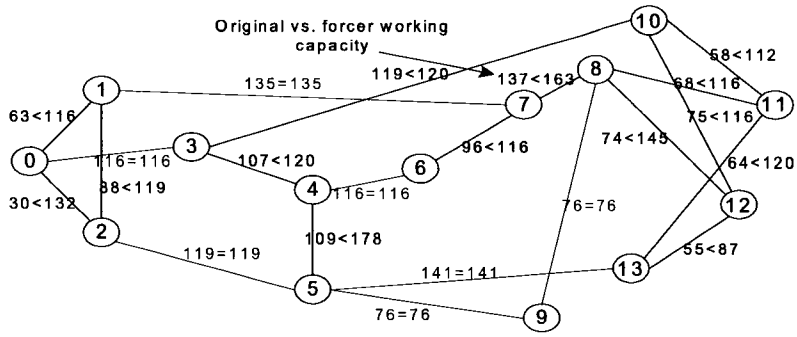
6.4. RESULTS OF FORCER ANALYSIS

Although all the methods developed for the forcer identification above are capable of finding the *same* set of forcers and discovering how many units of extra working capacity can be added before non-forcers become forcers, we have selected the simplest and the

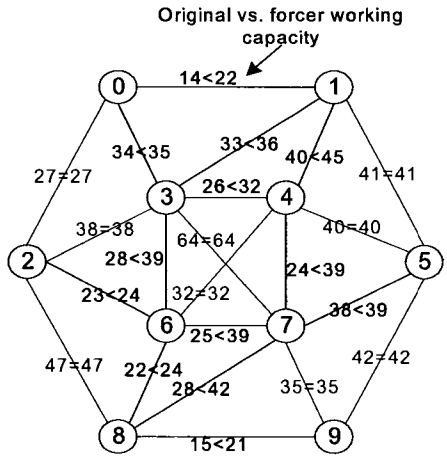
most efficient one, i.e., the one-run identification method, to analyze forcers in our experiments. We conducted experiments for four test networks as shown in Figure 6.5. They include the ARPA2, NSFNET, SmallNet, and COST239 networks. The numbers of nodes, spans, and elemental candidate cycles for each network are listed in Figure 6.5. Here the term "cycles" denotes eligible cycle candidates from which the design models may select actual p-cycles for the design solutions. These are not the number of p-cycles in the designs themselves. Network demands were generated following a uniform random distribution on the range [1...20] for each node pair. To route these working demands, we employed the demand-splitting shortest path algorithm. This means that if there is a single shortest path between a node pair, then the working demand between the node pair is completely carried by the path. Otherwise, if there are multiple shortest paths with the same length, then the demand between node pair is evenly distributed onto the shortest paths subject to retaining integer demand flow on each route. When searching the shortest paths, the number of hops functions as the unique metric of the selection, i.e., the routes with the smallest number of hops will be selected as the working paths. The design models consider all elemental cycles for the initial (conventional) spare capacity planning problems and later for the forcer identification method. The design problems were all solved to complete optimality within a short time by AMPL/CPLEX 7.1 on an UltraSparc Sun server at 450MHz with 4 GBytes of RAM.



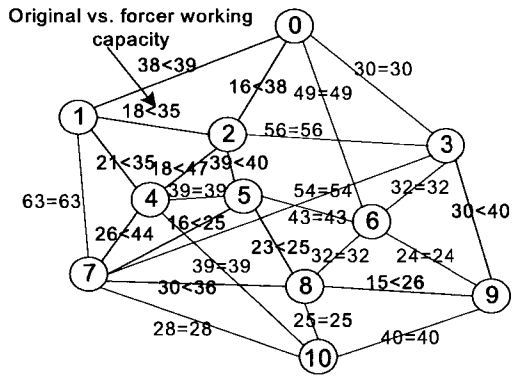
(a) ARPA-2



(b) NSFNET



(c) SmallNet



(d) COST239

Figure 6.5. The Identified Forcers and the Difference Between the Original Working Capacity and Forcer Working Capacity in the Four Test Networks: (a) ARPA2 with 21 Nodes, 25 Spans, and 18 Eligible Cycles, (b) NSFNET with 14 Nodes, 21 Spans, and 139 Eligible Cycles, (c) SmallNet with 10 Nodes, 22 Spans, and 833 Eligible Cycles, and (d) COST239 with 11 Nodes, 26 Spans, and 3531 Eligible Cycles.

Based on the above conditions, we obtained the experimental results of forcer analysis and discuss them from two perspectives. We first consider forcers and extra working capacity on the spans before non-forcers become forcers, and then the improvement of

spare capacity efficiency (i.e., redundancy) of the *p*-cycle network designs after exploiting the forcer structures.

In addition to the topological information of the test networks, Figure 6.5 provides the information on the forcer spans and how many working capacity units can be added to the spans so that they become forcers as well. The forcer spans have been highlighted in red, and a pair of numbers is provided adjacent to each span. The first number represents the original working capacity under the initial *p*-cycle network design, which was routed based on the demand-splitting shortest path routing algorithm. The second number indicates the working capacity when the span becomes a forcer of the network. We find that there are seven, seven, nine, and fourteen forcer spans respectively in the four test networks. All these spans have formed forcer skeletons in the respective networks.

Table 6.1. Experimental Results of Forcer Analysis in the Four Test Networks

Network	**ARPA2**	**NSFNET**	**SmallNet**	**COST239**
Initial working	7344	1922	716	844
Forcer working	8935	2539	803	984
% of extra working	21.7	32.1	32.6	16.6
Spare	8631	1371	267	296
Initial redundancy	1.175	0.713	0.373	0.351
Forcer redundancy	0.956	0.540	0.333	0.301
1/(d-1)	0.724	0.500	0.294	0.268

Table 6.1 presents more experimental results on the forcer analyses for the test networks. The data involve the working and spare capacities under initial *p*-cycle design, the forcer working capacity, and the spare capacity redundancies under the initial design and the forcer analysis. Note that the redundancy of forcer analysis is referred to as the network redundancy when all the spans become forcers. According to the table, we can see that a large amount of extra working capacity can be added before all the spans in the networks become forcers. In the four test networks, the minimum percentage of such extra working capacity is more than 16%. It should be noted that the addition does not require any increase in the network spare capacity, which stays the same as the initial design. In addition, it is important to note that under the forcer analysis, the network spare capacity redundancy is low enough to be very close to the lower bound of span-based survivability methods, i.e., *1/(d-1)* [GrDo01]. This implies that the forcer-based survivable network design is the most efficient way to design a survivable network.

6.5. POTENTIAL APPLICATIONS OF FORCER CONCEPT

There are two possible ways to apply the forcer concept based on the above forcer analyses. One is forcer-oriented, and the other nonforcer-oriented.

A) Forcer-Oriented Application

Because all the assigned spare capacity in a network is "forced" by the forcer skeleton, we may reduce the total required spare capacity by decreasing the working capacity on the forcers. One way to achieve this is to reroute the working capacity on the forcers onto some non-forcers in order to decrease the working capacity on the strongest forcers, which is equivalent to detouring the working paths from the shortest routes to some longer ones. Another approach is to design a hybrid network with ring and mesh-restorable techniques [GrMa00]. The approach was called *forcer clipping*, the basic idea of which is to remove or "clip" some working capacity from the strongest forcers and then use the ring technique to protect the clipped working capacity. There is a tradeoff in this approach. Specifically, the process of forcer clipping helps to decrease the total required spare capacity of the mesh-restorable network, but the clipped portion of the working capacity requires the ring technique to accommodate it, which is not as efficient as the mesh-restorable network in spare capacity efficiency. Hence, whether the consequent total network design cost is reduced or increased is determined by the difference between *saving* of forcer clipping and *wasting* of the ring accommodation. The total network cost will be reduced if the saved spare capacity in the mesh-restorable network after forcer clipping is more than the additional spare capacity required by the ring accommodation for the clipped working capacity. Otherwise, the network design cost will be increased. As such, the challenging issue for the above method is how to identify the optimal point at which the total network design cost reaches the minimum.

Figure 6.6 provides a conceptual illustration on forcer clipping. As shown, before forcer clipping, it is the strongest forcer that mandates the network to require that amount of spare capacity. For those non-forcers or hidden forcers, the spare capacity is actually over-provisioned. However, after forcer clipping, some part of the strongest forcer is clipped as shown, the required spare capacity to protect the remaining working capacity is thus reduced and the initial non-forcers or hidden forcers are now upgraded to be

forcers. With this forcer clipping process, the overall network cost can be reduced if the clipping of the strongest forcer can bring more spare capacity saving than the spare capacity required to accommodate the clipped part by rings. For more detail on forcer clipping, interested readers are referred to [GrMa00].

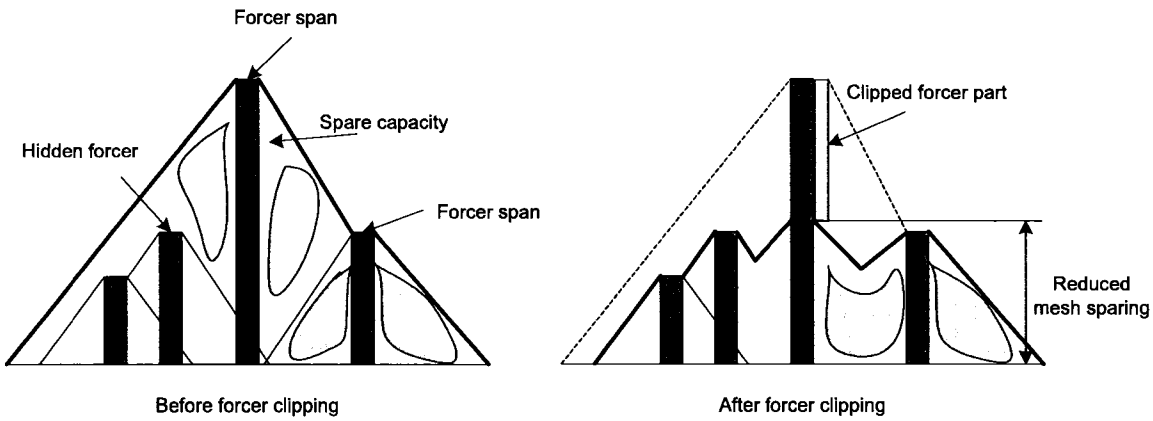


Figure 6.6. The Concept of "Forcer Clipping:" Rings that Make a Mesh More Efficient, in Page 755 of Reference [Grov04b].

B) Nonforcer-Oriented Application

Corresponding to forcer clipping, another possible application is termed "non-forcer filling," which is nonforcer-oriented [ShGr03b]. Under the "shadows" of the forcer skeletons, the assigned spare capacity for a network is actually over-provisioned for the protection of those non-forcers. To make full use of the over-provisioned spare capacity, an efficient option is to add more working capacity onto the non-forcers to make them forcers as well. The added working capacity can be used to serve future demands. Moreover, the addition does not cost more spare capacity. "Not requiring any additional spare capacity" is essentially the greatest advantage of the non-forcer filling application. The amount of extra working capacity that can be added reflects the potential of the network to serve future additional demands based on the initial network design. The larger the amount, the more future demands can be served. In the previous four test networks, it can be seen that significant differences exist between the working capacities of the initial work design and the design based on forcer analysis. Figure 6.7 illustrates the concept of non-forcer filling.

Essentially, the nonforcer-filling concept is the theoretical *foundation* of the following volume-maximization PWCE designs. We will provide more detail on this application in the context of PWCE design in Chapter 7.

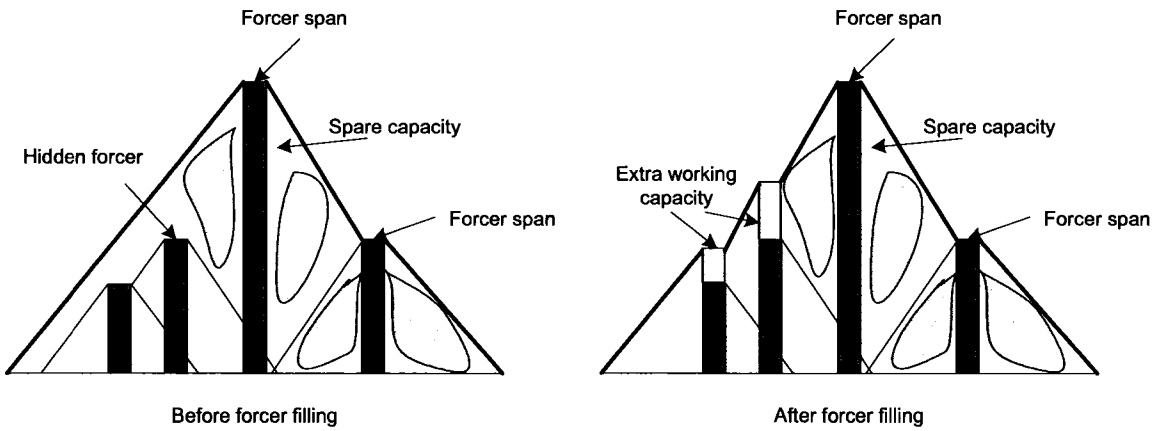


Figure 6.7. The Concept of "Non-Forcer Filling:" Serving More Future Demands without Increase of Spare Capacity.

6.6. SUMMARY

In this chapter, we introduced the forcer concept in the contexts of span-restorable network and span-protecting *p*-cycle network. A new forcer analysis method was proposed based on the non-forcer filling process. Its corresponding optimization model to identify the extra amount of working capacity before a non-forcer becomes a forcer was also presented. The concept of forcer and the related theories are important to function as the theoretical foundation of the following volume-maximization PWCE designs.

CHAPTER 7
P-CYCLE-BASED PROTECTED WORKING CAPACITY ENVELOPE (PWCE): CONCEPT, DESIGN, AND BASIC PERFORMANCE EVALUATION[6]

7.1. INTRODUCTION

In the literature such as [GeRa00] [ZaMu01] and in the prior chapters, discussion was presented on the importance of survivability and dynamic service provisioning for optical transport network design and operation. Shared Backup Path Protection (SBPP) [KiKo00] [KaSa94] is currently the *de facto* technique for survivable service provisioning. Under this technique, backup paths are allowed to share spare capacity providing their corresponding working paths do not share common failure. This greatly improves the capacity efficiency, but involves a long restoration time and complex network operation. Therefore, for SBPP a tradeoff exists among aspects of restoration speed, operation simplicity, and spare capacity efficiency.

Our work considers the new concept of a *Protected Working Capacity Envelope* (PWCE) [Grov04a] [Grov04b], which departs significantly from the most popular current strategy, i.e., SBPP, and involves a new approach to protection capacity design so as to create an optimized operating "envelope" of protected working channels distributed over the network. Within this envelope, dynamic protected demand provisioning is greatly simplified, relative to the current popular method SBPP.

While the PWCE concept is applicable to any form of span-protection mechanism, we consider *p*-cycles as the specific protection mechanism of interest. The particular contributions in this chapter are on the detailed implementation and various optimization strategies that can be used for such a protected envelope design. They include the strategies of envelope volume maximization design, related network state databases and routing algorithms, and performance evaluation and comparison in terms of blocking probability and control overhead with the SBPP-based provisioning method.

[6] This chapter contains some material previously reported in [Shen03], [ShGr04b], [ShGr04c], [ShGr05a], and [ShGr05b].

The chapter is organized as follows. Section 7.2 describes the PWCE concept and its advantages for dynamic survivable provisioning. Section 7.3 employs the GMPLS technology to control and operate PWCE networks. Section 7.4 presents various formulations for the design of "volume-maximized" PWCEs using p-cycles as the protection mechanism. Section 7.5 explains simulations and test methods used for PWCE design performance evaluation. Section 7.6 presents experimental results and compares performance of the PWCE and SBPP provisioning methods.

7.2. The PWCE Concept and Advantages

The PWCE concept for dynamic survivable service provisioning was first outlined in Chapter 5 of [Grov04b] and recently given further exposure in [Grov04a]. Here we excerpt from those sources to include the necessary basic background to go on to look at PWCE-oriented design problems. The concept of PWCE begins with consideration of a conventional survivable network design for static traffic demands. Given a demand forecast, a survivability scheme is applied where the restorability is guaranteed for any one failure at a time. In networks based on span protection, this results in a division of capacity into working and spare channels. The working capacity serves the demands in the forecasted matrix, while the spare capacity provides protection for the working capacity.

At first glance, such design methods seem limited to problems with a static demand matrix. In other words, they are intended to produce optimal designs only for cases where an exactly-known set of point-to-point lightpath requirements are available. The PWCE concept involves adaptation of these static design models to create not an exact solution for one single demand matrix, but an envelope of protected working channels, well suited for a large family of random demand instances that may be somehow related to a single representative demand pattern. To begin with, we realize that even when a model such as the one presented in Chapter 4 is solved for a specific demand matrix, the result is a set of working channels on each span that can actually support many different demand patterns, and a distribution of spare capacity that protects them all, providing none exceeds the w_i quantities of the initial design problem.

Figure 7.1 illustrates one of the many possible partitionings of total capacity into working and spare channel sets. What all such partitionings have in common is that the working channel count $\langle w_i \rangle$ on each span is fully restorable within the corresponding graph of spare channel capacities $\langle s_i \rangle$ on all other spans. Few arbitrary partitionings of capacity on each span will satisfy this property, but the theory for such fully-restorable partitionings is readily based on existing knowledge about span-restorable mesh network design [HeBy95]. Even while satisfying the restorability property, there are a large number of different partitionings, i.e., protected working envelopes that are feasible under an initial distribution of total capacity.

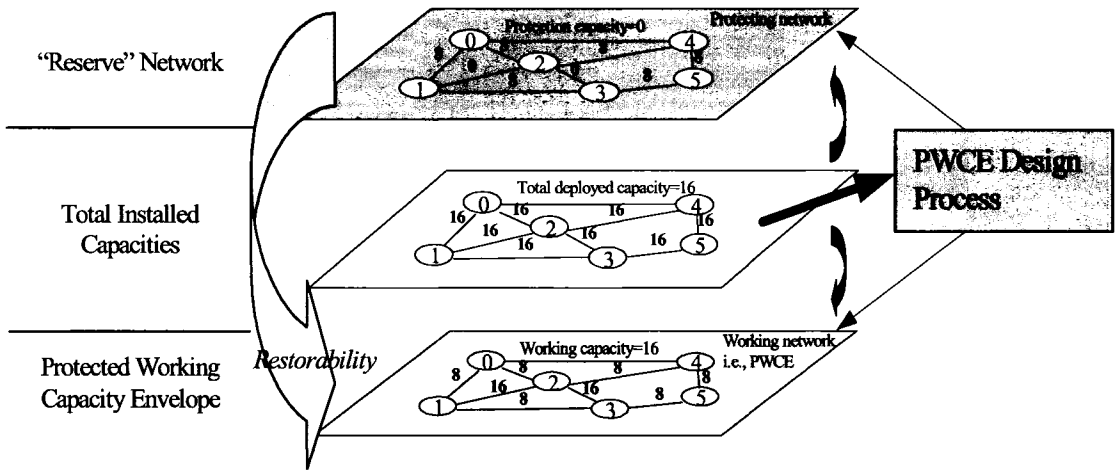


Figure 7.1. Partitioning of Total Installed Capacity into Working and Spare Units to Define One Possible Protected Working Capacity Envelope (Adapted from [ShGr04b]).

At the top of Figure 7.1, a set of spare channels on each span defines a *reserve network* of spare capacities $\langle s_i \rangle$. Under span restoration, any distribution of spare channels provides for a certain corresponding number of protected working channels $\langle w_i \rangle$ on each span below. To understand $\langle w_i \rangle$, one must in effect answer the following question: "If span *i* fails, then by rerouting through the spare capacity of the surviving graph between the end nodes of *i*, what is the *maximum* number of capacity units of replacement path segments that we can create?" A capable restoration algorithm will achieve $\langle w_i \rangle$ equal to the capacity of the minimum-cut between end-nodes of the failed span through the reserve network $\langle s_i \rangle$. After the partitioning is defined, any number or

combination of working paths can be routed through the envelope, up to the point where all $\langle w_i \rangle$ channels are used on some span, without any attention to protection arrangements. This is true because the channels used for provisioning in the working layer are themselves protected by the reserve network (and some embedded restoration or protection mechanism). As long as the quantities $\langle w_i \rangle$ on spans support routing of the demand, it is inherently protected end-to-end with no further action required. Once a connection is served, local marking on each span indicates to subsequent path setup processes that the individual channel is no longer available. No other nodes need an accounting of individual channel states, as they do in SBPP where sharing relationships are defined between individual paths and individual backup channels. Under PWCE other nodes need only know that the span continues having one or more provisionable channels available. This is the default case that requires no signaling for state update dissemination. Moreover, it can be appreciated that if span occupancies remain under $\langle w_i \rangle$ (i.e., within the envelope), then path setups and tear-downs can be going on arbitrarily. Consequently, it makes no difference—there is no signaling needed to arrange protection per-path and no global state update dissemination whatever. The only signaling is for the source-routed establishment and tear-down of each path by its originator node, and does not involve any nodes other than those on the paths themselves. Only if the *pattern* of random dynamic demand evolves in a way such that a span approaches the envelope is *any* updated network state dissemination required. A single Link State Advertisement (LSA) then either withdraws the highly utilized span from further routing or issues an updated cost for OSPF-type routing over that span.

Nodes operating under PWCE need only participate in simple Open Shortest Path First (OSPF)-type of topology orientation to support distributed end-node provisioning via a constrained source-routing protocol (such as RSVP-TE or CR-LDP). At this level of transport, basic topology is almost never changing. Therefore, this is virtually a one-time learning of the basic graph topology. Full-blown OSPF-TE dissemination of detailed changes in actual capacity and spare capacity sharing state on each span is not needed because every edge of the graph will remain available for routing providing its current in-use channel count is below the maximum number of working channels $\langle w_i \rangle$ that can be

protected on that span. Nodes can be informed via a centralized NMS what the dimension of the current PWCE is (i.e., the $\langle w_i \rangle$ value on each of its incident spans). This is a database of one number per span to be maintained within each node at the beginning, and does not need global dissemination. Alternately, the information may be infrequently discovered by each node by running a mock restoration trial using a distributed restoration algorithm executed in the background within the spare capacity [Grov97].

An important property of PWCE is that actions of any type related to ensuring protection occur only on the time-scale of the statistical evolution of the network load pattern itself (such as Erlang traffic load distribution), *not* on the time-scale of individual connections. Thus, any need for network management actions or state change dissemination is *far* less dynamic than the traffic itself. It takes a shift in the *statistics* of the demand pattern to require a logical change in the working envelope. Importantly, such adjustment actions also occur on a time scale where traffic exhibits correlated observable trends that can be taken into account in capacity-configuration planning. Variations in total demand and the pattern of demand have strong correlations day-over-day that would allow the advanced planning of several envelope configurations within the installed total capacities, each of which is known to suit the characteristic time of day to minimize any blocking. In contrast, SBPP works at the call-by-call time-scale where individual departures and arrivals always appear essentially random and trigger network-wide state updating, and routing is individually controlled by end-users. This is an environment of inherently incremental local reaction to the next arrival, not involving any opportunity for optimizing capacity use or routing strategies at global network level taking all demand into account at once. The protected envelope requirement is, however, slowly changing or static over long periods of time, even in the most frenetically dynamic network. No matter how rapidly individual lightpath demands come and go at random, the *envelope* requirement will not change at all if the demand process is at statistical equilibrium. The envelope is only sensitive to non-stationary drift in the underlying pattern of random arrival/departure processes. In Chapter 9 we will specifically study PWCE in the environment wherein the statistics of traffic demand pattern vary. However, for now, let us concentrate on a network in which the spare capacity of each span is pre-assigned and fixed. This defines a "static" PWCE.

PWCE is based on a locally-acting span restoration or protection mechanism. Examples of mechanisms that protect bearer capacity (i.e., channels, not paths) are span restoration [Grov97] or pre-planned span protection [HeBy95], cycle covers [ElHa00], generalized loopback networks [MéBa02], or *p*-cycles [Grov04a] [GrSt00]. BLSR rings also inherently protect bearer capacity and, under a suitable transformation (one that represents the constraints on routing between available rings) they are also amenable to the PWCE strategy. If network-level protection against node failure is required (in addition to span failure protection), the corresponding mechanisms can be node-inclusive span restoration [DoGr03], node-encircling *p*-cycles [StGr00b], or a centralized multi-commodity maximum flow rerouting solution for the transiting flows. All these options can be preplanned for a rapid localized response against single span failures, complemented by a slower but highly effective adaptive response to multiple failures including node failures [ClGr02]. Specifically, for multi-failure scenarios, to efficiently utilize network spare capacity, we can always employ the strategy of "first failure protection, later failure restoration." Although a bit slower in the restoration speed, a good percentage of "later failure restoration" can always be expected by using the spare capacity remained after the recovery of the first failure. Nonetheless, viewing the much lower occurring frequency and high design cost for multi-failure protection, this thesis has been focused on the study for a single-span failure under each network failure scenario.

A final preliminary comment is to eliminate a misunderstanding we have recently encountered about networks designed with *p*-cycles and/or networks operating under the PWCE/APWCE concept. The misconception seems to be that such networks inherently only anticipate an undifferentiated environment of service quality where all demands are assumed to require protection. In fact there is no such implication at all. Unprotected services are simply routed over working capacity and *left out of all* further PWCE considerations. That is why they are not mentioned. It is not that the methods considered cannot provide for unprotected services: rather, it is just that all we are considering here is *the subset of demands that do require protection.*

7.3. GMPLS-Based Service Provisioning under Fixed PWCE

In operation, PWCE-based dynamic survivable service provisioning is virtually the same as the service provisioning with no protection. The control and management applied for the non-protected network is therefore suitable to control the PWCE-based network as well. The focus of this section is on the GMPLS-based control architecture for PWCE dynamic survivable service provisioning. Although alternate architectures (such as a centralized one) are also eligible to control the system, GMPLS is chosen owing to its popularity and maturity. In addition, to avoid interference from the reconfiguration shift of an envelope, we also assume that each PWCE is fixed in both volume and structure (i.e., capacity distribution), though the control architecture described below is effective in any complex scenario (including PWCE reconfiguration).

As indicated, routing and signaling are the two most important components in the GMPLS control architecture. The routing component can be further subdivided into three sub-components, namely *network state database*, *network database synchronization*, and *routing algorithm*. Based on the description of GMPLS in Chapter 5, we discuss these components and related aspects in the context of PWCE below. The discussion will be made in comparison with the control system of SBPP in order to highlight the advantages of PWCE, particularly in terms of operational complexity.

7.3.1. Link State Database and Connection Database

The complexity of network state database is highly related to the robustness of an entire network control system. Under the GMPLS distributive control architecture, the network state database at each node in the network is required to be synchronized at any time. Incoherence of the network state database at different nodes may lead to a severe disaster to "bring down" the entire network control system and in turn the whole network. This is extremely vital to a network service provider, especially under today's highly competitive environment of network service provisioning. To ensure the coherence or synchronization of network state database and in other words the robustness of network control system, it is always preferable to maintain a smaller network state database, since a complicated network state database shows a higher probability of incoherence or unsynchronization under the GMPLS distributive control environment.

Network state database consists of link state database and connection database. In PWCE, link state information simply involves envelope resource state such as the remaining envelope capacity on each span. However, it involves none of information on network survivability such as protection capacity channels or spare capacity sharing relationship as in SBPP. Thus, compared to the latter, PWCE has a more simple link-state database, which enables PWCE to require a significantly lower control overhead for link-state synchronization and also much less network state memory to store the link-state information at each node.

The connection database of PWCE functions to store information on the *working paths* that only start or end at a local node, but none of information related to the paths that start or end at other nodes. Each piece of information corresponds to an established working path that contains information on required bandwidth and explicit route of the path, etc. Conversely, SBPP has a much larger connection database; it needs to store information on *all* the working and protection paths in the network, no matter whether or not started or ended at the local node. Moreover, any new connection establishment will trigger a new round of connection database synchronization to update the record of the new connection. Therefore, SBPP normally requires a high network control overhead. However, PWCE is inherently immune to this type of synchronization burden. Each node only concerns the working paths started or ended at *itself*, thus no synchronizations are required except that the envelope capacity on the spans, which the node is responsible for, is exhausted or re-available again. A similar situation occurs to the memory requirement for the connection database. A smaller amount of memory is usually required by PWCE at each node, for the scheme merely records the connections started or ended at the node. Moreover, for each connection only the information on working path is stored. However, under SBPP it is necessary for each node to record all the connections in the network, and even more, for each connection the information on both working and protection paths must be recorded.

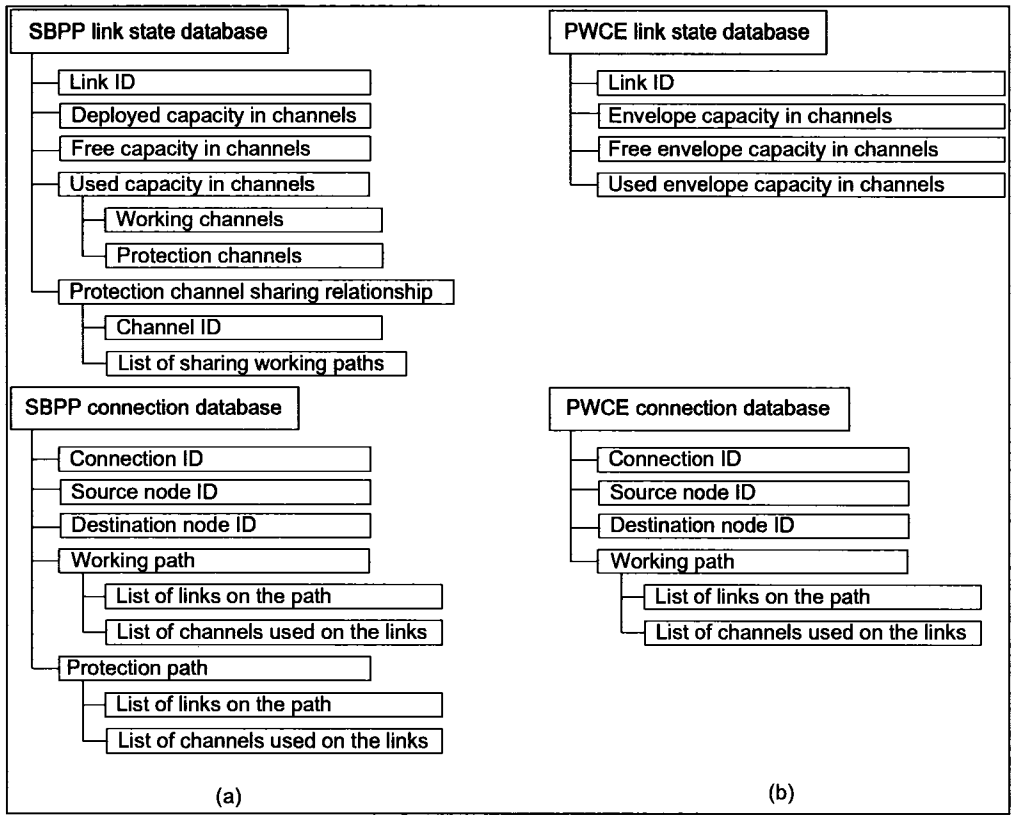


Figure 7.2. Comparison of Link State and Connection Databases between (a) the SBPP-Based and (b) the PWCE-Based Provisioning Approaches.

Figure 7.2 compares the databases between PWCE and SBPP. The databases of SBPP are the same as those shown in Chapter 5, while the databases of PWCE are virtually identical to those for the conventional service provisioning with no protection. From the figure, it is readily seen that PWCE has far simpler databases than SBPP.

Note that the above PWCE databases are presented in a general sense. Under certain circumstances it is possible to simplify them further. Specifically, if a span is assumed available providing it remains at least one unit of envelope capacity, then we can simplify the link state database by replacing the two integer fields, "free envelope capacity in channels" and "used envelope capacity in channels," with a simple binary field, "link availability." The old fields are only required when some threshold-based LSA flooding mechanisms or adaptive routing algorithms are employed. The new field is by default set as "one," which means that the span is available for envelope capacity. When all the envelope capacity on the span is used up, the field is set as "zero," which implies that the span is not available for capacity any more. Also, if each capacity unit is considered

identical or equivalent, we can condense the database further by removing the field of "list of channels used on the links." This field is necessary only when we consider special network cases such as the wavelength-routed networks with partial wavelength conversion or sparse OEO regeneration, where the wavelengths on the successive spans of a lightpath can be different and should be explicitly addressed.

7.3.2. LSA Flooding and Synchronization Process

LSA functions to synchronize the network state database around the network. Only with a synchronized state database can each node select proper routes for each service connection. This ensures sufficient network resources are available on the selected routes. Under the PWCE approach, three possible strategies can be applied for LSA flooding and database synchronization. The first strategy is the *most inactive*, which does not flood any LSA message unless the envelope capacity on a span is completely exhausted or available again. Due to its inactiveness, we term it *hibernating strategy*. The second strategy is somewhat *more active* than the first, which requires the network to flood an LSA message whenever the usage of the envelope capacity reaches a new threshold level. We therefore call it *threshold-based strategy*. The last strategy is the *most active*, which requires the network to flood an LSA message whenever there is any change within the network. Thus, any connection establishment or release will trigger a new round of LSA flooding, and the strategy has the same LSA flooding frequency as in SBPP. We call it *real-time strategy*. The details on these three strategies are presented below.

A) Hibernating Strategy

As already indicated, from the control and management point of view, PWCE-based service provisioning is virtually the same as service provisioning without protection. The hibernating strategy is proposed to minimize the network control overhead by always assuming a span is "connected" providing there is at least one unit of envelope capacity remaining on the span. No changes are assumed for the connectivity of the span if the span still retains free envelope capacity (even if only one unit). Only when all the envelope capacity on the span is exhausted should an LSA message be flooded to report this change. Specifically, the LSA message informs all the other nodes in the network that the span has been exhausted of envelope capacity and must be temporarily removed

from the network topology. The hibernating strategy generally requires a low control overhead for LSA message flooding. Given a certain amount of envelope capacity on each span, say 16 units, the control overhead for LSA flooding is almost negligible if the network traffic load is moderate. For example, assume that the average utilization of each capacity unit is ρ. The probability that all the capacity on the span is exhausted, which is equivalent to the probability that an LSA message is flooded, is ρ^{16}. Obviously, for a moderate network capacity utilization, the probability of LSA message flooding is very low (e.g., even if $\rho = 0.8$, ρ^{16} is less than 3%).

Despite the low network control overhead, the hibernating strategy suffers from the disadvantage of inaccurate network state information. This results in the inability to apply advanced traffic load-based adaptive routing algorithms [MoAz98] such as the Least Load (LL) algorithm [ShBo01b] to search routes. This is the case because the LL algorithm needs the detailed information of envelope capacity utilization on each span, while the hibernating flooding strategy can only provide a gross link state information, namely whether or not the span has free envelope capacity. Rather, under this strategy, only the hop-based shortest path routing algorithm can be used to search the routes. Therefore, all the disadvantages incurred to the hop-based shortest path routing algorithm, such as network congestion, are invariably coupled with the hibernating strategy.

B) Real-Time Strategy

As opposed to the hibernating strategy, the real-time strategy trades off the network capacity utilization with the network control overhead in other way around. Specifically, it achieves a better network capacity utilization at the cost of a higher network control overhead. As frequently as SBPP, the real-time strategy assumes an LSA message is flooded whenever there is any change of network resource on a span. For example, the establishment of a new connection consumes some network resources. Consequently, it is necessary for all the spans on the route to flood LSA messages to update the changes. The same holds true for the service release. Thus, the real-time strategy can always provide the most detailed and updated network state information, rather than a simple binary state information ("available" or "unavailable") as in the hibernating strategy. This enables the strategy to be eligible to apply the advanced load-based adaptive routing

algorithms that can greatly improve the network capacity utilization and correspondingly the blocking performance and throughput. Nonetheless, the strategy usually requires a much higher network control overhead than the hibernating strategy.

C) Threshold-Based Strategy

The threshold-based strategy is a compromise of the hibernating and real-time strategies. It compresses the network control overhead by decreasing the LSA flooding frequency, and improves the network capacity utilization by using the traffic load-based adaptive routing algorithms. Specifically, the strategy divides the envelope capacity usage on each span into several threshold levels. Whenever the unused (or used) envelope capacity reaches a new threshold level, a new round of LSA flooding is triggered. For a span with eight units of capacity, four threshold levels of capacity usage are evenly assigned. These threshold levels correspond to two, four, six, and eight units of unused capacity scenarios respectively, and each of them covers two continuous unused capacity situations. For example, the four-unit level covers the capacity situations of three and four units. To record the capacity left on each span, in the link state database all the continuous unused capacity situations in the same threshold level are rounded up to the ceiling capacity of the level. That is, if a span has unused capacity between two threshold levels, it is always rounded up to the level ceiling. In the previous example, the three-unit unused capacity will be rounded up to its level ceiling, four-unit unused capacity. The capacity change within the same capacity usage level, such as unused capacity change from four units to three units, both of which are in the same usage level, i.e., four-unit level, will not trigger any LSA message flooding. However, if the unused capacity reaches a new threshold level, a new LSA message should be flooded. For example, if the number of unused capacity units changes from three to two, which corresponds to a threshold-level change from four-unit level to two-unit level, an LSA message should be triggered to update this. A similar process can be undertaken for the service release. If the release of an old connection does not result in any change in the capacity usage level, no LSA message disseminations are required; otherwise, LSA messages should be flooded accordingly.

The benefit of the threshold-based strategy is that the network need not flood link state change as frequently as in the real-time strategy. Only when a certain (significant)

amount of capacity usage changes should an LSA message be disseminated to update this. This can greatly save the required network control overhead compared to the real-time strategy. It should be noted that under the threshold-based strategy, the information on the unused capacity on each span maintained in the link state database is normally not accurate, but only approximately measured by a capacity usage level. Some difference may exist between the real unused capacity and the approximate one. Yet the approximate estimation can still effectively reflect the network capacity utilization (or congestion), which is much more detailed than the simple binary state information ("available" or "unavailable") as in the hibernating strategy. Thus, the strategy is still eligible to apply the traffic load-based adaptive routing algorithms to search for a less congested route in order to improve network performance and capacity utilization. It is predictable that, with the increase of the number of threshold usage levels, the network capacity utilization will be improved as well, since the usage level increment yields more accurate network state information. Of course, this is at the cost of more frequent LSA flooding and correspondingly a higher network control overhead. Obviously, a tradeoff exists between the network control overhead and the capacity utilization under the threshold-based strategy. Thus, to reduce the network control overhead but meanwhile ensure a good capacity utilization, it is of significance to identify an optimal point at which a slight increase of control overhead can significantly improve the network capacity utilization.

Essentially, the hibernating strategy and the real-time strategy can be viewed as two extremes of the threshold-based strategy. Specifically, the hibernating strategy can be perceived as an extreme case where the number of assigned usage levels is one. In contrast, the real-time strategy can be envisioned as an extreme case where the number of assigned usage levels is equal to the number of maximal deployed span capacity units in the network.

In addition, under these strategies, there could exist a flopping situation that may increase the frequency of LSA flooding. We use the hibernating strategy as an example to explain this phenomenon as follows. Under the hibernating strategy, when a span is used up of its envelope capacity, a new LSA message that virtually removes the span from the network topology is flooded around the network. Later, a connection release may free

one unit of capacity on that span, which causes the network control system to flood an LSA message to declare the availability of the span again. However, sometimes such an operation may not be wise since the too-early declare of the availability of the span will force it to accept new service right away. As a result, if a new service is established through it, a new round of LSA flooding is required within a short time after the previous one. Essentially, this is just like a switch flipping to turn off and on, which can generate an oscillation of LSA flooding. To avoid this, a wise solution is to allow each recently-released span to take a rest (i.e., we apply a hysteresis policy to the threshold). That is, if a span changes its status from "fully occupied" to "with one unit of free capacity," no LSA message should be flooded to update this change. Rather, only when a sufficient amount of free capacity is accumulated on the span, should an LSA message be flooded to declare the span is available again. By doing so, the described flipping situation can be largely avoided. Moreover, specifically for the threshold-based strategy, we can further propose a more advanced strategy termed "progressive threshold." The key idea is that we do not assign the threshold levels in a uniform fashion. Rather, when a large volume of capacity is left on a span, we assign threshold levels coarsely, while when the capacity left on the span becomes smaller (a congestion sign appears), we assign threshold levels finely. By doing this, we can further compress the LSA control overhead for the threshold-based strategy but without affecting the network routing performance at all. As a generic study, we however did not implement the above two options in our experiments. Nonetheless, it can be predicted that the two options can always further compress the LSA flooding control overhead. Therefore, the results obtained in this thesis would function as a bound performance of these two options.

Finally, compared to the SBPP approach, all the above three strategies are advantageous in control overhead saving. Among them, the hibernating strategy is the most economical because it disseminates LSA messages only when the capacity on a span is totally exhausted or available again. Although the real-time strategy floods LSA messages as frequently as SBPP, the strategy only synchronizes the network state information on working paths, but not involving any information on protection paths or spare capacity sharing relationship. Ultimately, as a compromise of the hibernating and real-time strategies, the advantage of the threshold-based strategy relies on the number of

threshold levels. In general, the smaller the number of threshold levels, the more control overhead can be saved. However, this may sacrifice the accuracy of link-state information and in turn affect the network capacity utilization.

7.3.3. ROUTING ALGORITHMS

Depending on the LSA flooding strategies applied, various routing algorithms can be employed. As indicated, the hibernating flooding strategy provides the least information on link state, i.e., merely the information on whether or not a span has unused capacity. Therefore, only the hop-based shortest path routing algorithm is eligible to search routes under this strategy. In contrast, both the threshold-based strategy and the real-time strategy provide relatively more detailed link-state information Consequently, more advanced adaptive routing algorithms can be used to improve network capacity utilization and blocking performance, though the hop-based shortest path routing algorithm is also eligible to search routes for these two strategies.

Dijkstra's algorithm can be used to search both hop-based and traffic load-based shortest routes. The hop-based shortest routing algorithm uses the number of hops as the metric to search for the shortest route. In contrast, the traffic load-based shortest routing algorithms such as the Least Load (LL) algorithm [ShBo01b] take the traffic load or capacity utilization on each span as a metric to select a route, which has the lowest traffic load or is the most abundant of free capacity. Specifically for the LL algorithm, the metric used to measure the capacity utilization on a span is defined as u_i/t_i, where u_i is the number of used channels and t_i is the total number of deployed channels on span i. Particularly, for PWCE service provisioning u_i corresponds to the used envelope capacity and t_i corresponds to the total deployed envelope capacity on the span. As a special case, when $u_i = t_i$ (which means that all the capacity on the span is exhausted), the span should be virtually removed from the topology, or its cost should be increased to infinity, so that future connections will not access the span. Based on the above metric, Dijkstra's shortest path routing algorithm searches the route that has the minimum sum value of $\sum_{i \in R}(u_i/t_i)$, in which R represents the set of spans on the route. It is readily seen that the route found in this way has the least capacity utilization. Therefore, the algorithm

can evenly distribute the traffic load in the network and prevent network congestions. According to the results presented in [ShBo01b], the LL algorithm can significantly outperform the hop-based shortest path algorithm in blocking performance, although the former is a bit more complex in the searching-metric definition. The LL algorithm can be used to search routes under both the real-time and the threshold-based LSA flooding strategies. In the former, u_i represents an accurate value of the number of used channels, while in the latter, u_i is only a usage level that provides an approximate estimate for the capacity utilization on the span.

In addition, compared to the FF routing algorithm of SBPP, the advantage of PWCE is that for each survivable service request, neither the hop-based nor the LL algorithm is required to search for a protection route (but a working route only), whereas the FF algorithm of SBPP (in Section 5.3.3) always needs to search both working and protection routes. Moreover, the algorithms of SBPP need to consider spare capacity sharing, which however is not required by the PWCE routing algorithm. Therefore, PWCE shows another important advantage over SBPP, namely simpler routing algorithm.

7.3.4. SIGNALING PROCESS

The signaling process functions mainly to establish connections for survivable services. Protocols such as RSVP-TE and CR-LDP are qualified to fulfill that function. The details on how to use these protocols to set up paths were discussed in Section 5.1.3. Under PWCE, only a working path needs to be set up for each survivable service. In contrast, under SBPP two signaling sessions are required each for the working and protection path establishment. Thus, SBPP generally requires one more signaling session than PWCE to establish each survivable service.

Moreover, compared to SBPP, PWCE demonstrates another benefit of less performance degradation due to the network resource contention. As discussed in Section 5.1.3, it takes time to fully synchronize the network state database at each node when any given network state changes. Thus, during the service provisioning, it is possible that more than one service connection might be simultaneously computed to use the same network resource(s), but only one of them can eventually win the resource(s); all the others must give up and be blocked. For each survivable service, SBPP needs to

simultaneously establish two paths, i.e., working and protection. Consequently, it may incur a higher probability of this type of resource contention than PWCE that needs to establish a working path only[7].

7.4. Design of Protected Working Capacity Envelopes Using *p*-Cycles

The PWCE concept in the context of a *p*-cycle network is analogous to that in a span-restorable mesh network. The difference is that rather than the spare capacity being assembled on demand into a required path set for restoration of a specific failure that arises, a set of *p*-cycle structures is established in which all spare channels are pre-connected and have pre-defined the protection relationships with the individual channels of working capacity. Figure 7.3 illustrates a PWCE design based on the conventional *p*-cycle-based survivable network design. A six-node network and a demand matrix are shown on the left side, and the working and protection capacities based on the ILP model for the *p*-cycle network from [GrSt98a] are shown on the right. The set of working channels forms a PWCE capable of accepting dynamic survivable service requests from various node pairs. In this design, four *p*-cycles of various capacities protect all the working channels.

[7] However, in this study, we have ignored the above contention by assuming that the network state database at each node is always synchronized. This assumption may slightly underestimate the network blocking probabilities compared to real operations. However, it will not affect the effectiveness of overall performance comparison between PWCE and SBPP, because the contention always affects SBPP more than PWCE.

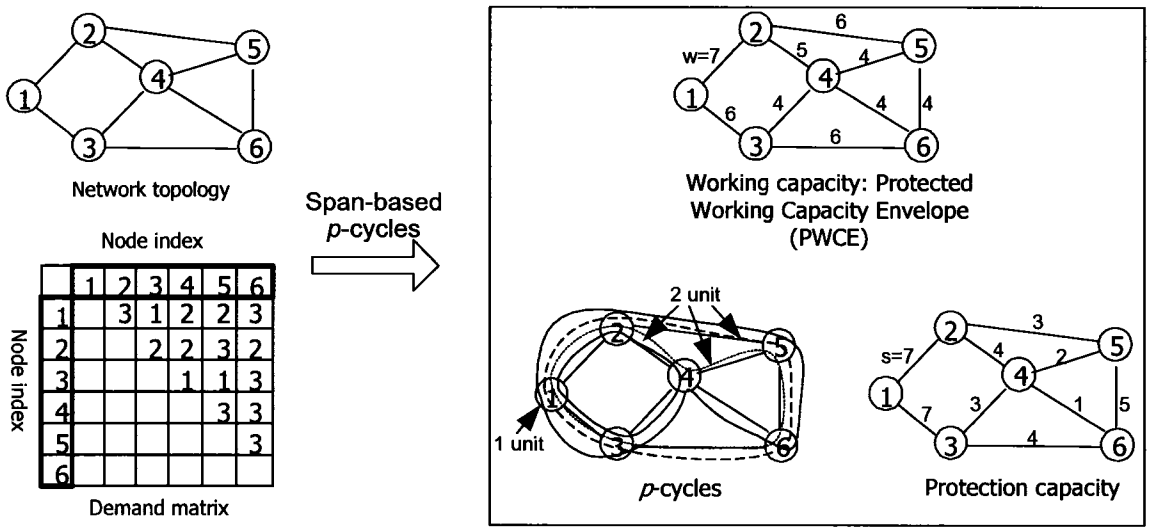


Figure 7.3. An Example of PWCE for the p-Cycle Network Based on the Conventional Survivable Network Design.

7.4.1. Forcer-Based Envelope Volume Maximization

In the example of Figure 7.3, a PWCE is constructed based on conventional p-cycle network design. Specifically, given a "static" matrix of exact demand requirements, working capacity requirements are first generated on each span by routing demands via shortest paths. Subsequently, based on the working capacities, p-cycles and associated spare capacities are designed to guarantee full protection of the working capacities. Given the total spare capacity that this creates, we can ask whether the corresponding set of working capacities we started with is in fact unique and maximal *in the envelope sense.* To address this, we consider the idea of the *volume maximization* in PWCE construction. In doing so we exploit the "forcer" structure of the initial p-cycle network design. The details on forcer theory have been described in Chapter 6. The idea here is, for envelope design, to make full use of the spare capacity in the network by raising the number of working channels on non-forcer spans so that they become co-forcer spans. This is exactly the same as the idea of non-forcer filling in Section 6.5 (see Figure 6.7). The result is that a greater total volume of working capacity can be established under the same spare capacity as the conventional survivable network design produced for the static target demand matrix. The PWCE designed with forcer structure exploitation has the largest envelope capacity, wherein all the spans become equal co-forcers of the spare capacity.

In Chapter 6, four approaches have been presented to analyze the forcer structure of a network. Among them, the *one-run identification* approach and the *non-forcer filling* approach are the simplest and most efficient for forcer identification. We employ the *non-forcer filling* approach to identify the extra protected working capacity given a certain spare capacity distribution.

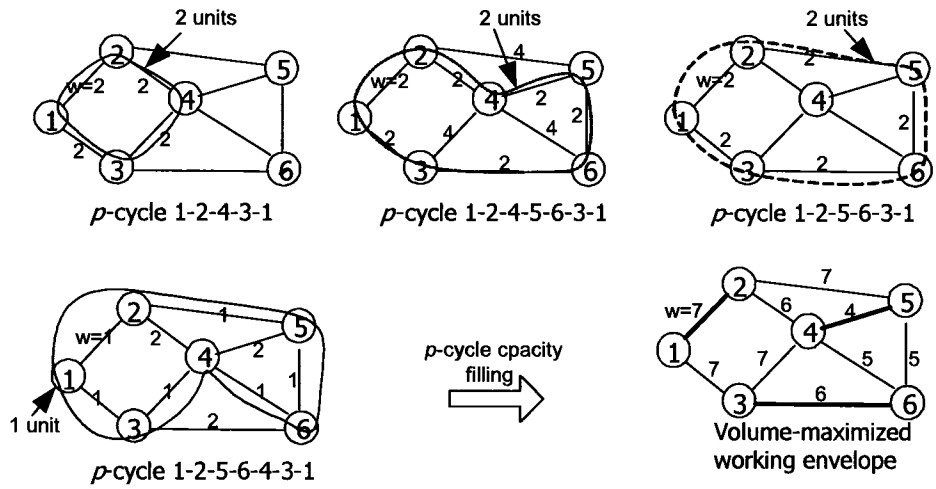


Figure 7.4. PWCE Construction Based on the Non-Forcer Filling or *p*-Cycle Capacity Filling Process.

Figure 7.4 illustrates the PWCE design based on the non-forcer filling approach for the network example in Figure 7.3. There are four *p*-cycles needed, as shown in Figure 7.3. Specifically, three *p*-cycles each require two units of spare capacity and one *p*-cycle needs one unit of spare capacity. In addition, we find that in the design there are three forcers, i.e., spans (1-2), (3-6), and (4-5), and all the others are non-forcers. We employ *non-forcer filling* or *p-cycle capacity filling* process to exploit the forcer structure to maximize the PWCE volume. For each *p*-cycle with a certain spare capacity we try to fill a working capacity, which is the same as the *p*-cycle capacity, to each on-cycle span, and a working capacity, which is a double of the *p*-cycle capacity, to each straddling span. The filled working capacities are fully protected by the *p*-cycles. For example, in Figure 7.4, *p*-cycle (1-2-4-5-6-3-1) has two units of spare capacity. Therefore, two units of working capacity can be filled on each of the on-cycle spans (1-2), (2-4), (4-5), (5-6), (6-3), and (3-1), and four units of working capacity can be filled on each of the straddling spans (2-5), (3-4), and (4-6). For the remaining three *p*-cycles, similar filling processes can be carried out. All the working capacities filled for the *p*-cycles are finally summed to

form the protected working envelope shown on the right side in Figure 7.4, which is a volume-maximized PWCE. Specifically, it has a total of 54 protected working channels, which are eight channels more than the initial design.

7.4.2. Target Pattern Matching Principle in PWCE Design

However, PWCE volume maximization alone simply creates the greatest total number of protected working channels over the entire network. They may not always be in the "right place" in the network to be most effective. The idea of *target pattern matching* is therefore to improve the performance of an envelope by *structuring* or *shaping*[8] the distribution of the protected working channels on spans to be at least reflective of some basic pattern of relative intensities expected of future demand. Mathematically, this pattern specification can be identical in form to a static demand or load matrix, but its interpretation and meaning is no longer that of an assumed exact forecast. Rather, it is simply a shaping template to help structure the PWCE as it is simultaneously maximized in volume. The philosophy is that even though demand is random and the future pattern has uncertainty in forecasting, it is still worthwhile to provide a "best view" of the relative future intensities of demand on each node pair. This serves as a guideline for structuring the PWCE to generally fit plausible future demand scenarios better than if pure volume maximization is effected with no such guidance at all. Given a plausible forecast of relative loads between node pairs (a *network load template*), we can identify which spans are traversed by high volumes of traffic loads (or simply traversed by many shortest-routes between all node pairs) and hence should preferably have a large working capacity assigned to them during the envelope volume maximization. Given a certain total design budget, we can then design a volume-maximized PWCE with a structure matching the distribution of network relative traffic loads. The obvious hypothesis is that such structuring should improve blocking performance when later subjected to random demand having an overall pattern of intensities similar to the structure of the designed PWCE. Later, to test this hypothesis, we quantify the match between the two distributions by measuring the correlation $\phi(\langle w_i \rangle, \langle l_i \rangle)$ between the PWCE structure (i.e., channel

[8] In the remainder of the thesis, the terms of *structuring* and *shaping* are interchangeable; so are the terms of *structure* and *shape*.

capacity distribution) $\langle w_i \rangle$ and the set of relative load accumulations on spans $\langle l_i \rangle$ that arise from demand-splitting shortest-path routing of the template matrix. The expectation is that, under similar volumes of envelope capacity, blocking performance should be enhanced with higher correlation of the PWCE capacity distribution to the forecasted relative load pattern even though actual demands arrive randomly, as long as their overall pattern of relative intensity is reasonably consistent with the forecast.

While *pattern matching* by itself will make the "structure" of PWCE reflective of an expected relative load pattern, at some point of asserting shaping-matching this may come into conflict with the joint goal of volume maximization. An interesting research question therefore arises: Which factor is more dominant in influencing network blocking performance, sheer volume or shaping of the PWCE? We will discuss this matter in Section 7.6.5.

7.4.3. Various Design Capacity Budgets

To construct volume-maximized PWCEs, a budget-limit of spare (or total) capacity investment must be defined to constrain the problem. The budgets can be span-based or network-wide. A span-based budget means that a specific maximum number of spare or total channels is allowed on each span. With a network-wide budget, the constraint is only on total capacity of the network as a whole. A network-wide budget normally has more freedom in the PWCE construction than a span-based budget. In summary, there are four possible types of budget scenarios to consider:

Span-based spare capacity budget where a certain maximum number of spare channels is allowed on each span. The limit can differ for each span.

Span-based total capacity budget where a budget of total capacity is set on each span. The total capacity is the sum of working and spare capacity, but there is no constraint on how to split the total span capacity into the working and protection capacities.

Network-wide spare capacity budget where a total spare capacity budget is set on a network-wide basis. No constraints are set on the distribution of this total spare capacity.

Network-wide total capacity budget, where a limit applies to the sum of all network-wide working and protection capacities, but without any constraint on distribution or working/spare split.

7.4.4. ILP Design Models for PWCEs

There are several approaches to constructing PWCEs. One way is to use the conventional span-based *p*-cycle design method. Under this method, the designed PWCE often contains many non-forcer spans. Therefore, the envelope is not optimal and does not fully exploit the spare capacity in the PWCE volume-maximized sense. To achieve a better efficiency, we employ the forcer-structure-exploitation process to construct volume-maximized PWCEs, where the working capacity of the PWCE on each span is elevated to bring the span into a co-forcer relationship with the initial forcer spans. In [ShGr03b], we developed three ILP models to exploit this form of extra PWCE capacity under the forcer structure of the conventional designs. Here, we extend these models to construct PWCEs under various budget constraints. We also consider the *pattern matching* (structuring) effect in the designs. This involves a total of eight possible design combinations, summarized in Figure 7.5.

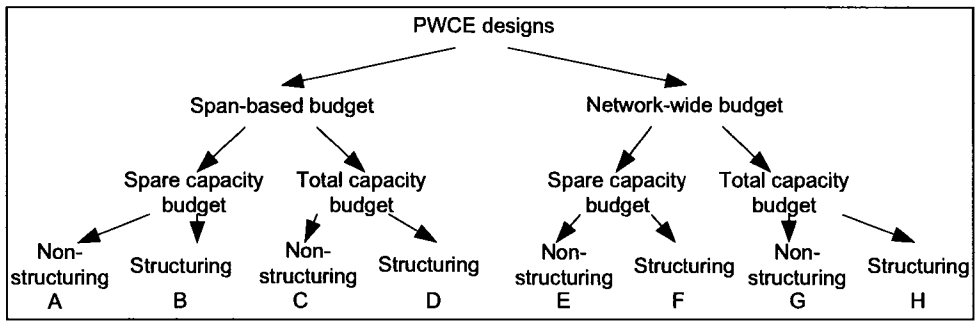


Figure 7.5. Taxonomy of PWCE Design Models.

In addition to the sets, parameters and variables defined in Section 4.1.2, new parameters and variables are given as follows:

New Parameters:

- l_k is the predicted relative load on span k, which defines a structuring pattern for the structuring PWCE designs. For the study of a fixed PWCE in this chapter, it can be computed from a given demand or load-forecasted matrix based on the

demand-splitting shortest path algorithm. If there is more than one shortest route existing between the same node pair, the demand or load units are evenly split onto each of the shortest routes subject to retaining integer demand flow on each route. Sometimes l_k can also be a more complex case such as the *expected envelope capacity*, which is the product of the link utilization and the envelope capacity of span k of currently operating envelope. This type of template will be specifically used in the case of Adaptive Protected Working Capacity Envelope (APWCE), which will be discussed in more detail in Chapter 9.

- T_k is the total number of deployed channels on span k, some of which will be assigned as the working capacity and the rest as the shared spare capacity.
- B_s is the total network-wide spare capacity budget.
- B_{w+s} is a total network-wide capacity budget, which is the sum of working and spare capacity budgets.
- α is a weighting factor to mediate the tradeoff between shape-conformance structuring and volume maximization of a PWCE.

New Variables:

- λ is a shape-asserting factor that structures the PWCE relative to the target load distribution. (real variable)

Now the ILP models under the eight combinations are presented as follows:

Model A: Volume Maximized under Span Spare Capacity Constraints (Non-Structuring)

Objective: maximize $\sum_{k \in S} w_k$ (7-1)

Constraints:

$$\sum_{j \in P} X_i^j \cdot n_j \geq w_i \quad \forall i \in S \tag{7-2}$$

$$s_k \geq \sum_{j \in P} P_k^j \cdot n_j \quad \forall k \in S \tag{7-3}$$

In this model we assume that a given set of spare channel counts $\langle s_k \rangle$ on each span. These may have arisen from a nominal design to a nominal forecast, or they may simply be the unused capacities of existing spans. Within this spare capacity environment the problem is to form a set of p-cycles that protects the largest number of working channels on spans of the network as a whole, i.e., to maximize the bulk volume of PWCE. Constraint (7-2) states that all the working capacity of the envelope is protected by the p-cycles. Constraint (7-3) ensures that the spare capacity used to form the set of p-cycles never exceeds the budget on each span.

Model B: Combined Structuring and Volume Maximization under Span Spare Capacity Constraints (Structuring)

Objective: maximize $\left\{ \lambda + \alpha \cdot \sum_{k \in S} w_k \right\}$ (7-4)

Constraints:

Subject to constraints (7-2) and (7-3) above, to which we add

$$w_k \geq \lambda \cdot l_k \quad \forall k \in \boldsymbol{S} \tag{7-5}$$

In this model we again assume that a given set of spare channel counts $\langle s_k \rangle$ on each span. Within this spare capacity environment the problem is to form a set of p-cycles that then protects the largest possible total number of pattern matching working channels on the spans of the network as a whole. However, we now have a bi-criterion objective function that attempts to shape the envelope to be similar to the network load distribution by bringing in a shape factor λ and constraint (7-5), which puts utility on the similarity between the PWCE and the target network load distribution. The overall objective becomes a tradeoff between envelope volume and shape matching with load distribution. In the test results, however, the tradeoff factor α is given a very small value, which guarantees that the maximization of the shape factor λ as the primary objective. As a result we tend to see λ maximized until the spare capacity cannot guarantee restorability of the envelope if the factor is further increased. The secondary objective is to maximize the envelope volume after it cannot be further enlarged under the exact shape of the predicted network load distribution.

This model can be applied to identify forcers as well. If we set the load distribution parameter l_k to be the existing working units that were obtained previously from the shortest path routing algorithm, then λ will always be equal to one and the maximization objective will solely maximize the extra protected working capacity on the non-forcer spans. The spans without any extra protected working capacity can then be identified as forcers.

Model C: Volume Maximization under Span Total Capacity Constraints (Non-Structuring)

Note that with this and subsequent models, s_k becomes a variable as well. In this model we assume a given set of total deployed channel counts on each span. This is closest to the situation of an existing deployed network of transmission systems. Within this total capacity environment the problem is to split the total capacity on each span into two parts. One functions as the PWCE and the other as the protection capacity to form a set of p-cycles offering protection for the envelope.

Objective: maximize $\sum_{k \in S} w_k$

Constraints:

Subject to constraints (7-2) and (7-3) above, to which we add

$$T_k \geq w_k + s_k \quad \forall k \in \boldsymbol{S} \tag{7-6}$$

The objective is to maximize the volume of PWCE. The total capacity used as working and spare channels on each span must never exceed the total present.

Model D: Combined Structuring and Volume Maximization under Span Total Capacity Constraints (Structuring)

This model is similar to model B to maximize the envelope in the shape in line with the forecasted network load distribution. The only difference between them is that this model has a span-based *total* capacity budget, while model B has a span-based *spare* capacity budget.

Objective: maximize $\left\{\lambda + \alpha \cdot \sum_{k \in S} w_k\right\}$

Constraints:

Subject to constraints (7-2), (7-3), (7-5), and (7-6) above.

Model E: Volume Maximization under Network-wide Spare Capacity Constraints (Non-Structuring)

Objective: maximize $\sum_{k \in S} w_k$

Constraints:

Subject to constraints (7-2) and (7-3) above, to which we add

$$B_s \geq \sum_{k \in S} s_k \quad \forall k \in S \tag{7-7}$$

This model is similar to model A, except that instead of a set of spare channel counts on each span, a network-wide total spare capacity budget is given in this design. The objective is to maximize a PWCE with the protection of the *p*-cycles formed by the network-wide spare capacity budget. We do not put a constraint on how the spare capacity budget should be assigned onto each span as long as their sum never exceeds the total network-wide spare capacity budget. Since the constraint on the spare capacity budget is network-wide, instead of span-based, such a design can construct a larger envelope than model A. The additional constraint (7-7) is to ensure that the sum of the spare capacity assigned to each span never exceeds the total network-wide spare capacity budget.

Model F: Combined Structuring and Volume Maximization under Network-wide Spate Capacity Constraints (Structuring)

Objective: maximize $\left\{\lambda + \alpha \cdot \sum_{k \in S} w_k\right\}$

Constraints:

Subject to constraints (7-2), (7-3), (7-5), and (7-7) above.

This model is similar to model B to maximize the envelope in the shape in line with the predicted network load distribution. The only difference is that the current model has a network-wide spare capacity budget, while model B has a span-based spare capacity budget.

Model G: Volume Maximization under Network-wide Total Capacity Constraints (Non-Structuring)

Objective: maximize $\sum_{k \in S} w_k$

Constraint:

Subject to constraints (7-2) and (7-3) as above, to which we add

$$B_{w+s} \geq \sum_{k \in S} \left(w_k + s_k \right) \quad \forall k \in S \tag{7-8}$$

This model is similar to model C, except that we assume the total channel count budget is network-wide, not of a per-span nature. The design assigns the total network-wide channel count onto each span. The capacity on each span can be further divided into the working portion and the protection portion. The working portions on all the spans make up a PWCE, and the protection portions on all the spans become the spare capacities to protect the envelope. There are many possible combinations for the assignment. The best assignment is the one that generates a PWCE having the largest volume. Constraint (7-8) is added to guarantee that the sum of network-wide working and protection capacity never exceeds the total network-wide capacity budget.

Model H: Combined Structuring and Volume Maximization under Network-wide Total Capacity Constraints (Structuring)

Objective: maximize $\left\{ \lambda + \alpha \cdot \sum_{k \in S} w_k \right\}$

Constraints:

Subject to (7-2), (7-3), (7-5), and (7-8) above.

This model is similar to model D to maximize the envelope in the shape in line with the predicted network load distribution. The only difference between them is that this

model has a network-wide total capacity budget, while model D has a span-based deployed capacity budget.

7.4.5. Practical Applications of the Design Models

According to various assumptions and environments, different volume-maximized PWCE design models can be applied in practice. For example, if a set of spare capacities on each span is given, which can be the spare capacities that have been assigned there by the conventional design, we can employ the model A or B to construct envelopes for the existing network, and some extra protectable working capacity can be exploited without adding any extra spare capacity. Similarly, if a set of total capacities on each span is given, which can be the systems having been deployed, we can re-optimize the network design by employing the models C or D to make the spare capacity more efficient for protection, and exploit more protected working capacity by maximizing the PWCE. Finally, for greenfield network designs, where a network-wide total capacity budget is given, we can design a new network that is the most efficient in the network resource utilization for a certain network topology. For this, we may employ models E, F, G, or H to construct the largest-volume PWCE and to shape it to accommodate the most demand.

7.5. Simulation Tests for Survivable Service Provisioning

In this section we describe the detailed experimental methods for implementing and comparing SBPP and PWCE schemes in terms of control overhead and blocking performance. Under dynamic service provisioning, a network can be seen as a discrete event-driven system with two types of random event, *service connection arrival* and *service connection release*. For comparative studies of blocking performance, we follow widely-adopted practice and assume that arrivals follow a Poisson process with an arrival rate of ρ per second, and each established service has a mean holding time of $1/\mu$ with negative-exponential distribution. We normalize time measurements using $1/\mu = 1$ so that the service traffic load between each node pair can be considered in units of Erlangs as ρ. The arrival and release event sequences run independently on each node pair concurrently. The blocking probability is defined as the ratio of the network total blocked service requests to the number of total arriving requests over a simulation period. We compute blocking from the simulations of both PWCE and SBPP based on survivable provisioning

processes operating under equivalent capacity in the same networks with identical time sequences of random arrivals and departures. The following sections explain in detail how PWCE and SBPP schemes were implemented for these comparisons.

7.5.1. Simulation for PWCE Provisioning

Under the PWCE scheme, once a PWCE is defined, provisioning within it is the same as service provisioning with no protection. We can employ the hop-based shortest path algorithm[9] or the LL algorithm [ShBo01b] to search for the first feasible route for each newly-arriving service request. If the search succeeds, then the path is established and the status of the available network resources is updated for each span with the consumed resources set as *unavailable*. This is a centrally computed result, but it exactly emulated the operation that is supported in practice with distributive source routing based only on a simple OSPF-type view of the topology of currently non-exhausted spans. Only when there is no free capacity left on a span is the span effectively removed from the graph seen by the routing algorithms so that the route search process will not take these spans into account. In practice this corresponds to issuing a single LSA withdrawing the span from further provisioning operations. For the LL algorithm, it is possible to use OSPF-type edge costs in shortest path calculations and allowing the edge costs to be updated slowly in response to the current filled levels on each span. In practice such an updating process can follow either the threshold-based or the real-time LSA flooding strategy. If no route is feasible at the time of a given service request, the request is blocked and dropped[10]. For service release events we simply unmark all the working channels of the released path to make them available for new use. In the general concept, when a release restores a certain minimum number of unused channels to a span that was previously withdrawn, then a single LSA is issued reinstating that span for provisioning.

7.5.2. Simulation for SBPP Provisioning

Our experience in terms of the complexity of the computer programming to implement SBPP for blocking simulations, compared to that for PWCE, almost directly reflects the

[9] Note that multiple shortest routes possibly exist with the same hop length. For simplicity without discovering all of them, we always select the one that is first found by the algorithm.

[10] In our study we do not consider the case of connection redialing: as with Erlang-B traffic theory, all the blocked requests are assumed to be dropped without re-try.

same difference in complexity that also exists operationally in practice. The basic simulation steps for the SBPP provisioning are similar to those of the simulation for the PWCE provisioning. For the service request arrival event, the process will seek the route-searching algorithm, the FF algorithm, in Section 5.4.3 to find a pair of first-fit working and protection routes. The two routes are guaranteed to be link disjoint, and the protection path is ensured to maximally share the spare capacity in the network. Once the working and protection paths are established, the network state should be updated to record information on working capacity, spare capacity, and spare capacity sharing relationship on each span. Meanwhile, the information on the working and protection route pair should be recorded in the connection database for future use when computing a new service connection. Of course, if the searching process cannot find such a pair of working and protection routes, the service request must be blocked and discarded.

Upon service release, the simulation process will release the resources consumed by the working path, and any spare channels used solely by its backup path. Here "solely" means that the released backup path is the only one currently assuming use of that spare channel if needed. A spare channel is only truly released back to an available state when all sharing relationships on it are removed.

7.5.3. Test Networks and Envelope Patterns

We evaluated the various design cases in simulations on several test networks. They involve a 21-node 25-span ARPA2 network, a 14-node 21-span NSFNET, a 10-node 22-span SmallNet, and an 11-node 26-span COST239 network, which were shown in Figure 6.5. Among them, ARPA2, NSFNET, and COST239 are the topologies of real networks. SmallNet is an artificial one, but was widely studied by researchers. Therefore, these networks can provide effective studies for most of practical designs. Moreover, we see that the sequence of these test networks actually reflects an increase of average network nodal degree. ARPA-2 is the sparsest network with a nodal degree of 2.38. NSFNET and SmallNet have nodal degrees of 3.00 and 4.40, respectively. Finally, COST239 has the highest nodal degree of 4.73. Thus, by selecting these networks, it is also possible for us to evaluate how the performance of the proposed p-cycle PWCE will be affected by network nodal degree.

We dimensioned these networks to define the capacity constraints for different PWCE cases, with a conventional *p*-cycle spare capacity minimization model and demand-splitting shortest path routing of an initial demand matrix to determine an initial set of working and spare capacities. We assumed that there were on average two, three, three, and two units of demand intensity on each node pair respectively for the networks above in order. These were the initial demand matrices used to generate the capacity requirements of the conventional design, which provide baseline capacity constraints for the following volume-maximization PWCE designs and performance comparison between PWCE and SBPP. Given the above demand intensity matrices, the demand-splitting shortest path algorithm operated as follows: If there was only a single shortest route between a node pair, then all the demand between the node pair was carried by the route. Otherwise, if there was more than one shortest route between the node pair, then the total demand was allocated as evenly as possible while remaining integer onto each of the routes. The required spare capacity was computed by the *p*-cycle SCA model in Section 4.1.2. The resulting spare capacities were the basis for span and network-wide total spare capacity constraints where applicable, and the working capacity $\{w_i, i \in \boldsymbol{S}\}$ was used as the *structuring pattern* $\{l_i, i \in \boldsymbol{S}\}$ for these structuring designs. The sum of working and spare capacities was used where total capacity constraints applied. All the above problems were solved quickly to optimality by AMPL/CPLEX 7.1 on an Ultrasparc Sun Server at 450MHz with 4GB of RAM except COST239, which took several hours for some models.

To fairly compare the performance of the PWCE and SBPP, we ran simulations for both with equivalent network capacity. For the survivable network based on the conventional *p*-cycle design, we used the resulting protected working capacity to function as a PWCE. Corresponding to this, we used the same total capacity on each span as being available to provision services with SBPP. SBPP therefore sees the total capacity of the network as being available, without any *a priori* designation of channels as working or (shared) protection. The identical demand intensities in Erlang between node pairs were presented for both SBPP and PWCE provisioning methods, but the latter provisioning process only "sees" the PWCE capacities of each span as being available for service

provisioning. A total of 10^5 arrival events were simulated for each test point for all the networks except ARPA-2, for which a total of 10^4 arrival events were simulated. This was due to the extremely lengthy simulations required by ARPA-2 for 10^5 arrival events.

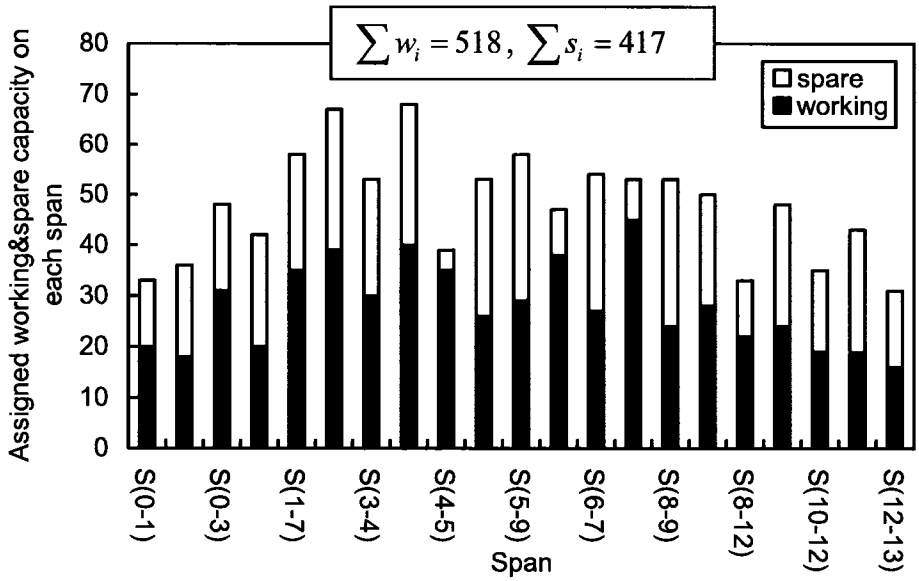
7.6. Results and Discussion

In this section, we present and discuss our experimental results. We first analyze the structural properties and capacities of PWCEs designed for the four test networks. This is followed by the blocking performance comparison between the PWCE and SBPP provisioning methods. The required control overhead and memory for network state database storage are compared as well. The blocking performances of the two routing algorithms, i.e., hop-based shortest path algorithm and the LL algorithm, are discussed in the context of PWCE to see how a traffic load-based routing algorithm can improve blocking performance compared to an algorithm that is not concerned about such a measure. In addition, for the threshold-based LSA flooding strategy, the effect of the number of threshold levels on the PWCE's blocking performance and control overhead is studied as well. Finally, we also look into the tradeoff effect between envelope volume and shape for the sake of identifying whether there exists an optimal point at which the designed envelope can achieve the best performance.

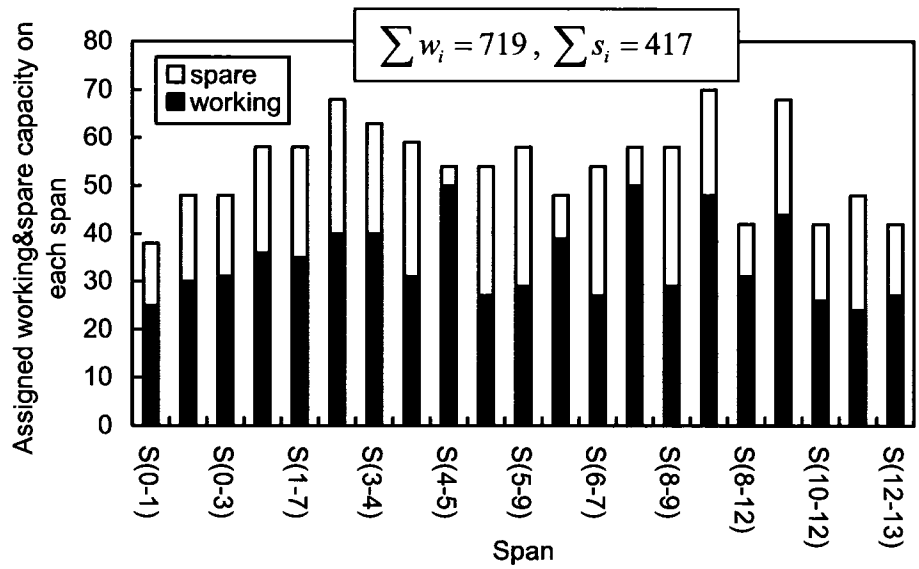
7.6.1. PWCE Structural Properties and Capacities

This section mainly reports the detailed information on the designed envelopes that were later used for the performance comparison with SBPP. Let us first inspect design results for models B, E, and F in terms of how volume maximization and combined structuring and volume maximization affect the working and spare capacity structures of the networks. Figure 7.6 portrays the structure of PWCEs designed by the conventional model and volume-maximized models for the NSFNET network with three units of forecasted demand on each node pair as the structuring pattern. Figure 7.6(a) is for the conventional p-cycle design model, from which the initial spare capacity and total capacity budgets for the volume-maximized PWCE designs were determined. The corresponding volume-maximized PWCE designs are as shown in the figures from Figures 7.6(b) to 7.6(d). The result in (b) was obtained by employing the spare capacity on each span as the constraints (i.e., model B). The results in (c) and (d) were obtained by

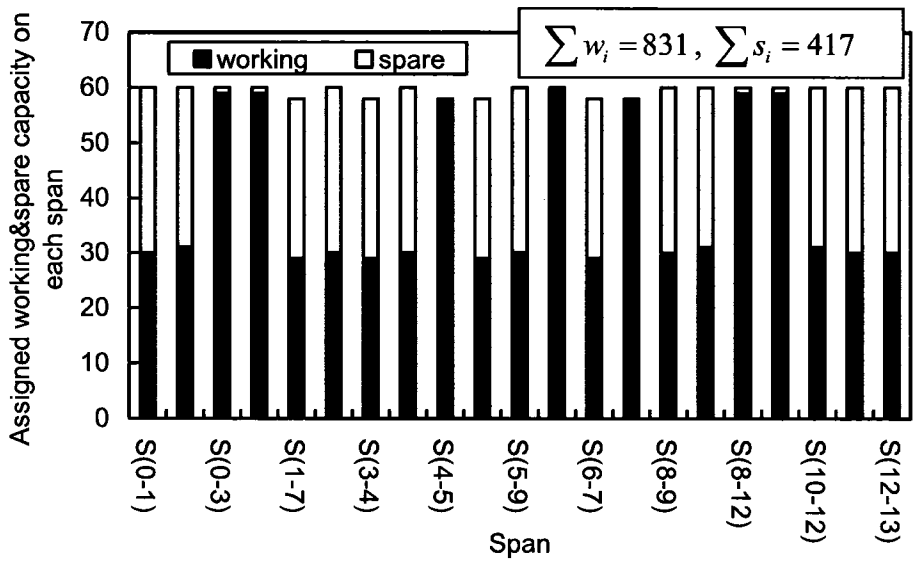
employing the network-wide total spare capacity, i.e., 417 units, as the constraint (i.e., models E and F). Their difference is that model E is a non-structuring design, while model F is structuring. We find that after implementing volume maximizations, all three designs can exploit significant volumes of extra protected working channels for their PWCEs (respectively 38.8%, 60.4%, and 44.2% relative to the original PWCE volume), and models E and F can construct larger envelopes than model B as the former are using a network-wide total spare capacity budget. Note that all these increments do not require any additional spare capacity increase.



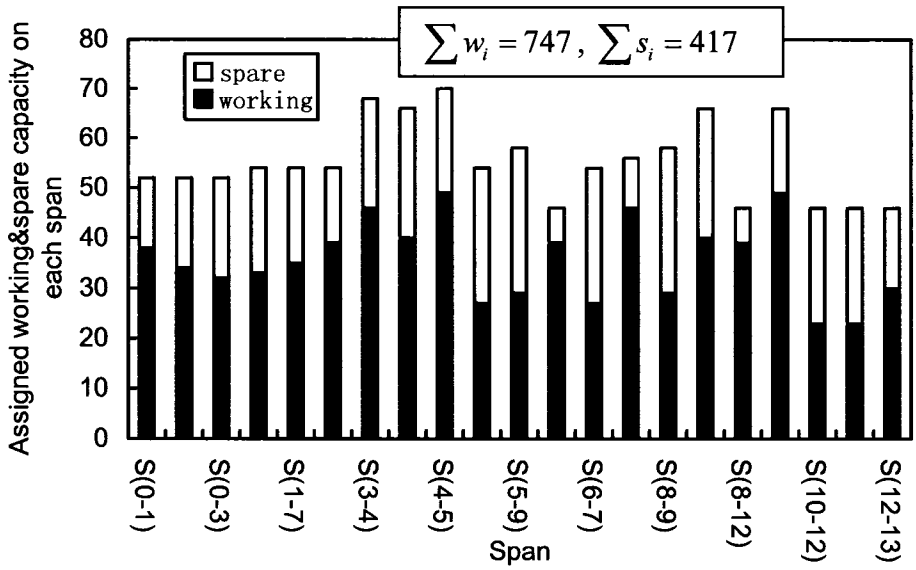
(a) Envelope of Conventional Design (NSFNET)



(b) Envelope of Model B (NSFNET)



(c) Envelope of Model E (NSFNET)



(d) Envelope of Model F (NSFNET)

Figure 7.6. Protected Working Capacity Envelopes (PWCE) of the NSFNET Network with Three Units of Uniform Forecasted Demand on Each Node Pair.

In the following, we discuss the structures of envelopes designed by the models. Model B has the same spare capacity structure as the conventional design, but because the working capacity of the conventional design has been assumed as the target pattern for the PWCE designs, the PWCE designed by model B can always "wrap" all the working capacity of the conventional design, for it raises protected working capacity on each span until spare capacities in the network become equally binding constraints on

further volume maximization. Figures 7.6(a) and 7.6(b) show that the PWCE capacity of model B is always greater than that of the conventional design on each span, though the spare capacities of the conventional and model B designs are exactly the same. We therefore describe such a situation as the PWCE of the conventional design *inside* the PWCE of model B. Similarly, because the working capacity of the conventional design is used as the structuring pattern for model F, the envelope of the conventional design is also inside the envelope of model F. However, in contrast to model B, model F need not follow the same spare capacity distribution as model B; it need only ensure that the total spare capacity never breaks the overall constraint. Thus, model F usually exhibits more flexibility in spare capacity arrangement and can construct a larger PWCE than model B. However, this does not mean that the PWCE of model B must be inside the PWCE of model F; here the word "larger" only refers to the volume. As shown in Figures 7.6(b) and 7.6(d), although the PWCE of model F is "larger" than that of model B, on span (8-11) model B has more PWCE capacity than model F. Finally, because no structuring effort is involved, model E always has the largest PWCE among all the design cases, but this again does not mean that its PWCE can fully *contain* the envelope of the conventional design. For example, in Figures 7.6(a) and 7.6(c) the conventional design has more PWCE capacity on span (2-5) than model E, although the former has a much smaller overall PWCE volume than the latter. In addition, without any structuring effort, the PWCE capacity distribution designed by model E is often not ideal. An interesting observation for model E is that the design has almost a *flat* capacity distribution when summing the spare and working capacities on each span. For example, in NSFNET we can see that the network designed by model E has almost the same capacity (i.e., 60 units) on each span. In summary, model E designs the largest envelope, but it may have an envelope structure that does not match the target design pattern, model B designs the smallest envelope, but it has a perfect match with the target pattern, and model F is one between the previous two models to have a large envelope and a good structure match to the target pattern.

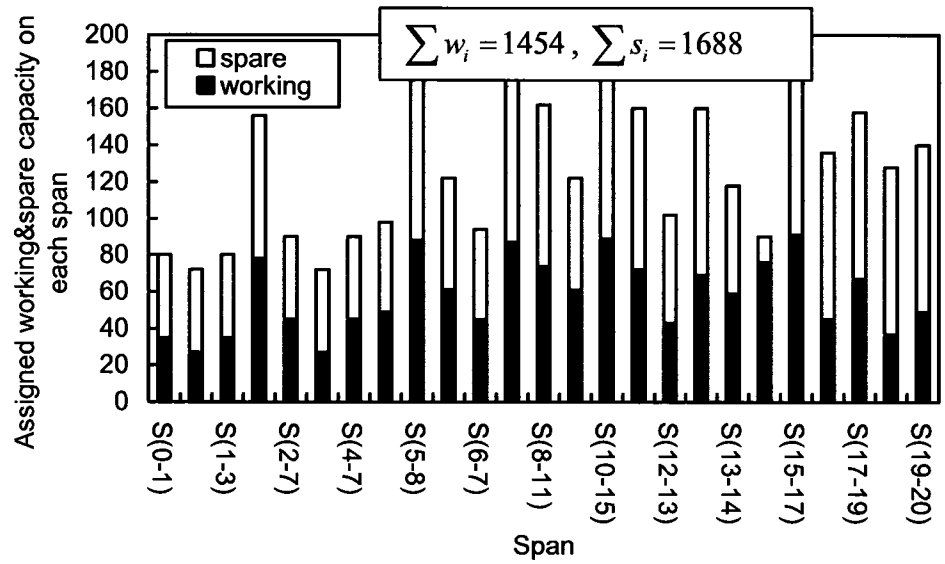
Similar results were obtained for the ARPA-2, SmallNet, and COST239 networks, as shown in Figures 7.7, 7.8, and 7.9 respectively. In all the test networks, the model E always designs the largest envelopes while model B generates the smallest envelopes.

Additionally, as before, compared to the envelope of the conventional design, all three models significantly expand the designed envelopes. Specifically, in ARPA-2, models B, E and F expand the envelope in volume 20.0%, 32.6%, 20.0% more than the conventional design respectively; in SmallNet these increments become 19.7%, 27.7%, and 19.7%; and finally, in COST239 the increments are 11.6%, 20.9%, and 16.3%.

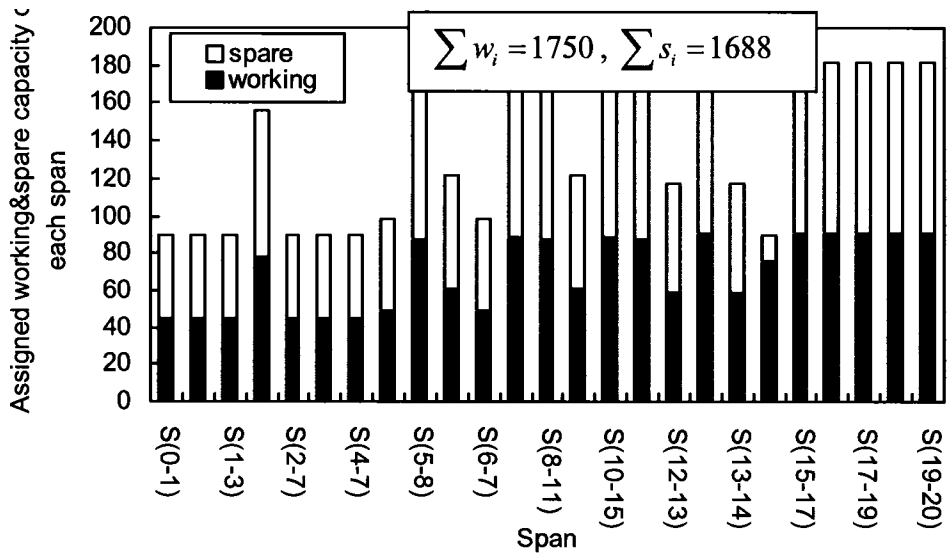
It is interesting to see that in both ARPA-2 and SmallNet networks, models B and F have the same volumes of envelope. Moreover, in ARPA-2 models B and F even demonstrate the same envelope capacity distribution. Although in SmallNet models B and F do not yield the same envelope capacity distribution, the same total volume of envelope proves that the envelope of model B can also be an optimal solution to model F since the solution can meet all the constraints of the model. This finding can be ascribed to the non-uniqueness of the ILP solution of the problem. Specifically, in ARPA-2 the solution spaces of models B and F are so specific that their optimal solutions must converge. In contrast, in SmallNet the solution space of model F is broad enough to have multiple solutions to reach optimality, among which the solutions of model B and F are two special cases. For model F the final optimal solution actually relies on the starting point of each optimization process. However, no matter which optimal solution is selected, it can be ascertained that all the solutions to model F must share the same shape factor λ if the secondary objective factor α is given a small value during the optimization.

In addition, for ARPA-2 another important observation is that under the design of model E, there are six spans that are assigned no working or spare capacity. They include spans (1-3), (3-4), (4-7), (5-8), (8-11), and (11-12). This essentially means that all these spans are virtually removed from the network topology, and no survivable services are allowed to traverse them. This therefore makes nodes 3, 4, 8, and 11 completely isolated and not allowed to establish any connections with other nodes. Such a design is admittedly awkward. The underlying reason can be ascribed to the non-structuring feature of model E—the model purely maximizes the envelope volume but ignores controlling the envelope structure. In contrast, in all the other structuring designs

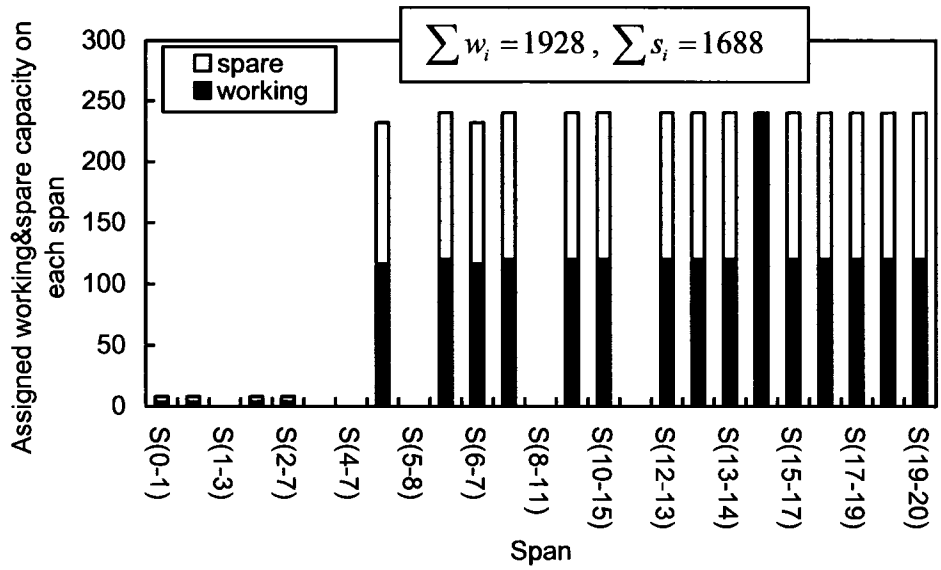
including models B and F, this type of situation does not occur. This verifies the importance of structuring effort to the PWCE design.



(a) Envelope of Conventional Design (ARPA-2)

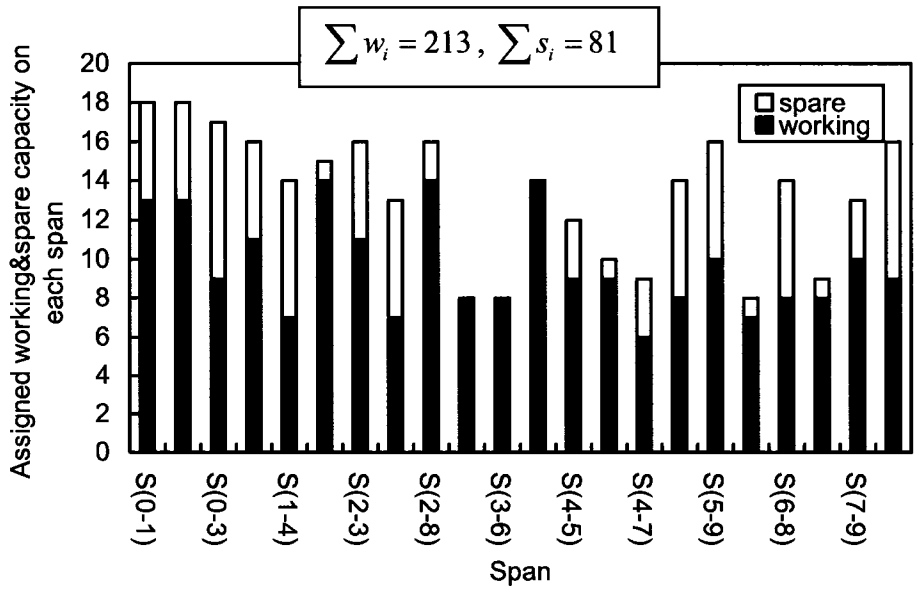


(b) Envelope of Models B and F (ARPA-2)

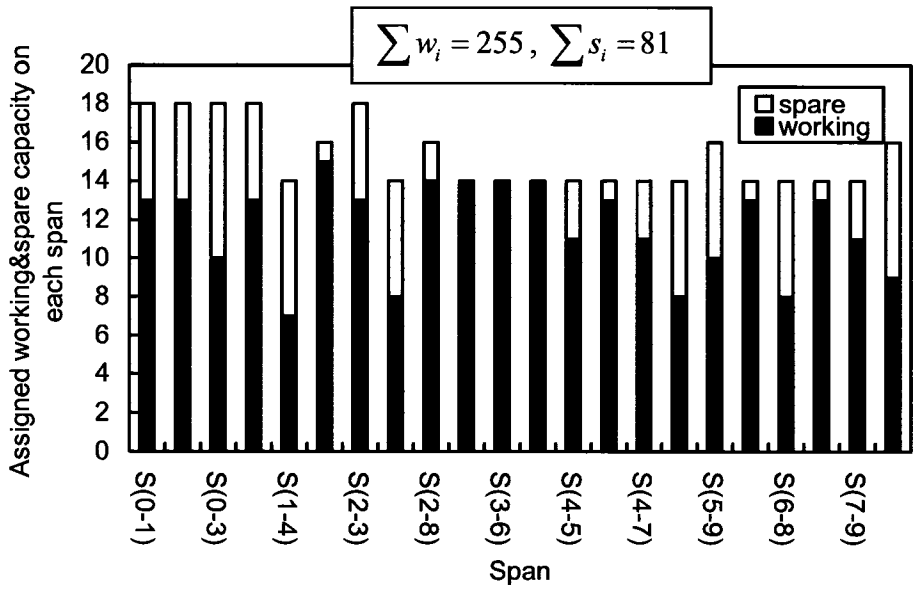


(c) Envelope of Model E (ARPA-2)

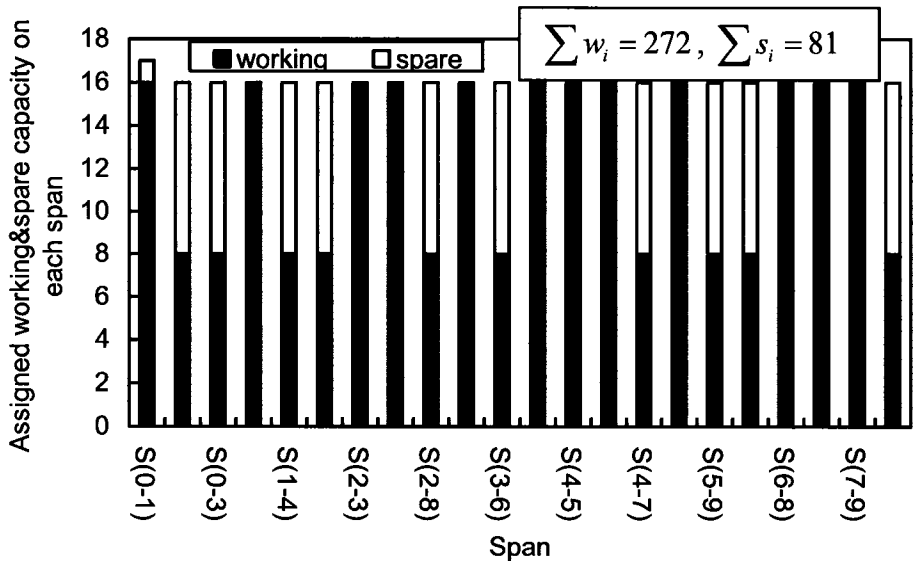
Figure 7.7. Protected Working Capacity Envelopes (PWCE) of the ARPA-2 Network with Two Units of Uniform Traffic Demand on Each Node Pair.



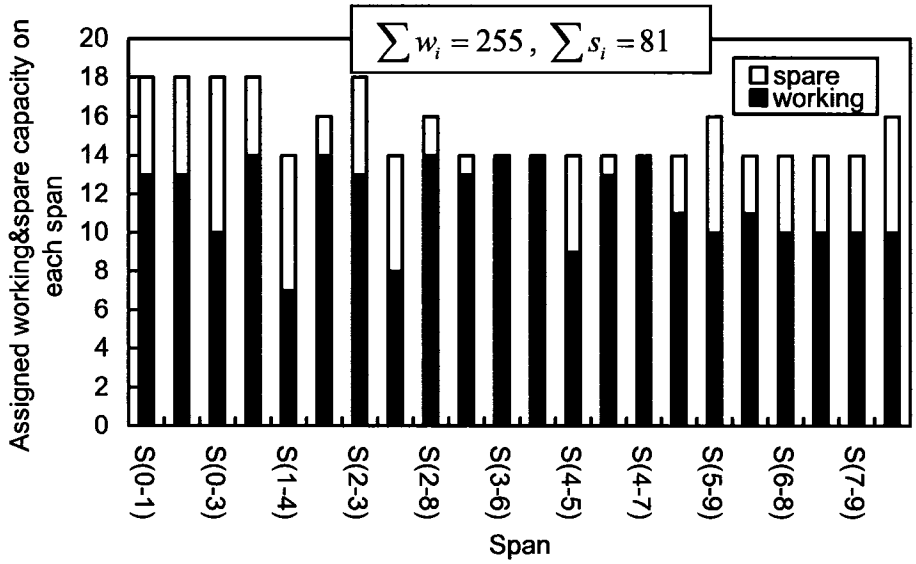
(a) Envelope of Conventional Design (SmallNet)



(b) Envelope of Model B (SmallNet)

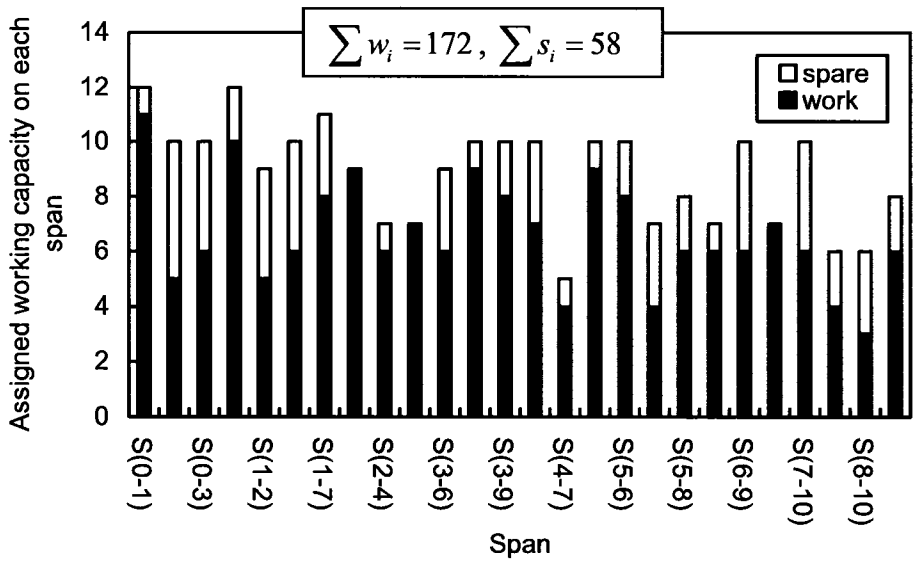


(c) Envelope of Model E (SmallNet)

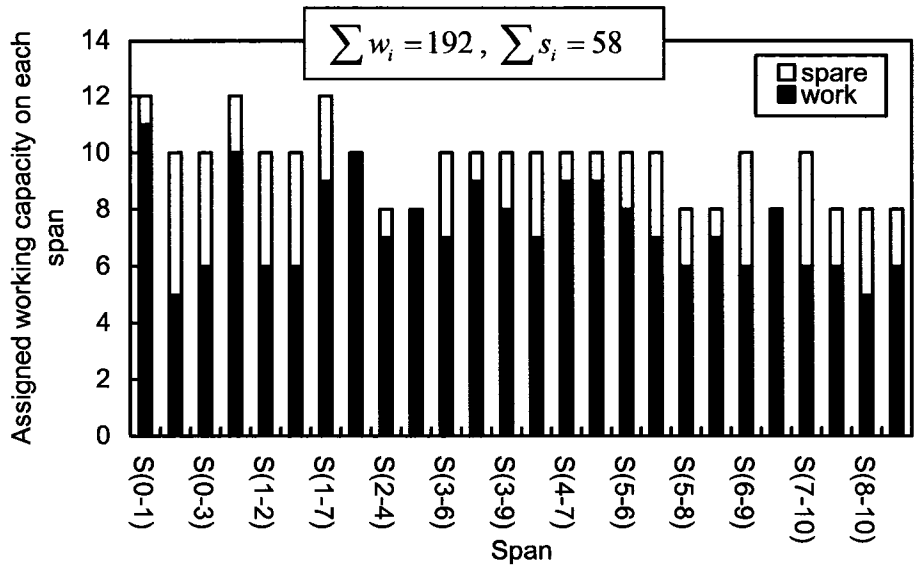


(d) Envelope of Model F (SmallNet)

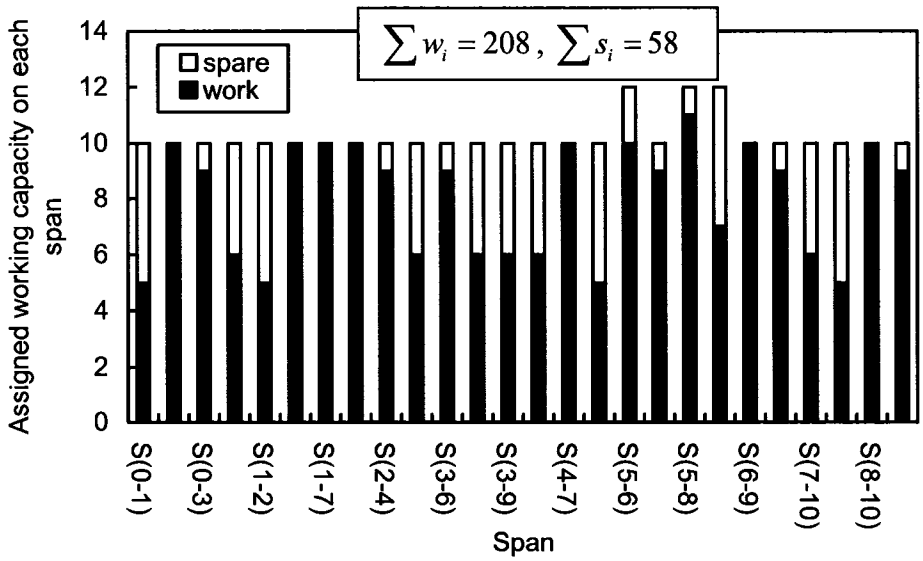
Figure 7.8. Protected Working Capacity Envelopes (PWCE) of the SmallNet Network with Three Units of Uniform Traffic Demand on Each Node Pair.



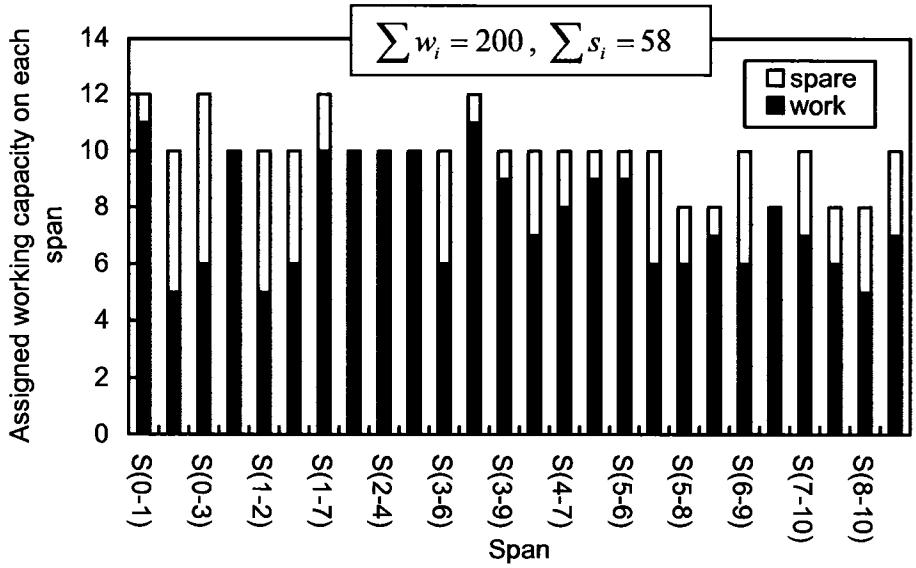
(a) Envelope of Conventional Design (COST239)



(b) Envelope of Model B (COST239)



(c) Envelope of Model E (COST239)



(d) Envelope of Model F (COST239)

Figure 7.9. Protected Working Capacity Envelopes (PWCE) of the COST239 Network with Two Units of Uniform Forecasted Demand on Each Node Pair.

For the structure of envelope, we can further use a parameter termed correlation coefficients between the structuring pattern and actual PWCE channel capacity distribution (i.e., the shape of the envelope) to evaluate how the designed envelope matches the target design pattern. Table 7.1 shows the correlation coefficients. The structuring patterns are the working capacity distributions of the conventional design as shown in Figures 7.6(a), 7.7(a), 7.8(a), and 7.9(a), while the envelope capacity distributions designed by models B, E, and F are shown in Figures 7.6(b)-(d), 7.7(b)-(c), 7.8(b)-(d), and 7.9(b)-(d) respectively. It is seen that models B and F always have larger correlation coefficients than model E. This is the case again because models B and F force the designs to do their best to conform to the structuring pattern, while model E does not. Therefore, it again confirms that the envelope designed by models B and F have better matches to the target design pattern than model E.

Table 7.1. Correlation Coefficients between Structuring Pattern and Actual PWCE Capacity Distributions in Models B, E, and F (E is Non-Structured)

Network	Model B	Model E	Model F
ARPA2	0.691	0.269	0.691
NSFNET	0.601	0.233	0.520
SmallNet	0.568	-0.007	0.486
COST239	0.801	-0.381	0.805

7.6.2. CONTROL OVERHEAD AND NETWORK STATE MEMORY REQUIRED BY SBPP AND PWCE

This section makes a performance comparison between PWCE and SBPP in the aspects of network control overhead for LSA flooding and the required network state memory. In summary, we find that the PWCE approach shows a much lower LSA flooding control overhead and a much smaller amount of network state memory than SBPP.

To verify the operational simplicity of PWCE, we conducted experiments to collect the statistics for the network control overhead and the average memory required at each node for the storage of the network state and connection information. Assumptions were made as follows. First, for PWCE we applied the hibernating LSA flooding strategy. This means only when the capacity on a span is exhausted or available again would an LSA message be flooded with an information pair like (Span ID, Status). As long as the capacity on a span is not exhausted, the span status is always conceived as "available" by default. Corresponding to the hibernating flooding strategy, the hop-based shortest path routing algorithm was applied to search routes within envelopes.

For SBPP, in order to attain a good network capacity utilization, it was assumed that an LSA message would be flooded whenever a new connection was established or an old connection was released. There are two possible methods to flood LSA messages, but they both fulfill the function to synchronize the network state database. The first method is similar to the PWCE scheme, in which each end node of a span is responsible for flooding the state change information within the network whenever any state change occurs on the span. Alternatively, one may ask the source node of each connection to flood the information on the change of connection status. Each of the LSA messages should contain information as follows: (1) working route, which is represented by a sequence of span IDs, (2) protection route, which is represented by a sequence of span IDs as well, and (3) information of spare capacity sharing on each span along the protection route. The LSA message flooded by the first method consists of only an information pair (Span ID, Status), which hence is smaller in size than the one flooded by the second method wherein each LSA message consists of a sequence of (Span ID, Status)

that makes up the information for a working path or a protection path. On the other hand, for each new established or released connection, the number of nodes that participates in the flooding processes is different for the two methods. In the first method, all the nodes on a working or protection route are required to flood the information on the state change of the spans enroute. In the second method, only the source or destination nodes of the connection participate in the flooding process. All the intermediate nodes do nothing but receive the messages on new connection establishment or old connection release, and update their network status databases. In summary, the first method has more nodes participating in the flooding process, but the size of each LSA message is much smaller than that of the second method. In contrast, the second method has a larger LSA message, but only two end nodes of each connection are required to flood connection state information. Thus, if we define the *information pair*, i.e., (Span ID, Status), as the basic unit of control overhead, we find that the two flooding methods actually have an equivalent total amount of control overhead. The first method assumes that at least one of the end nodes of each span is responsible for flooding the span state change. For an N-hop route there are at least N nodes participating in the flooding process when a connection is established or released. As such, for each survivable service, the total required control overhead would be equal to $W+P$ units of information pair, where W is the hop-length of the working route, and P is the hop-length of the protection route. Corresponding to this, in the second method, the source or destination node of each connection is responsible for flooding information on the state change of the connection. For a survivable service with a W-hop working route and a P-hop protection route, to flood such a message the network also needs to consume $W+P$ units of basic information pair since the information of each route consists of a sequence of Span IDs and their individual status[11].

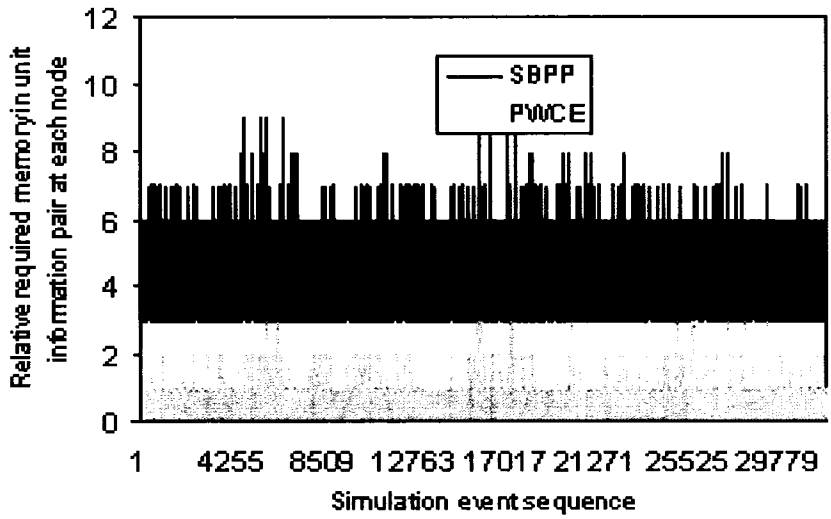
Compared to SBPP, PWCE is expected to have a much lower control overhead for LSA flooding owing to (1) lower LSA flooding frequency, and (2) smaller LSA message size. The lower LSA flooding frequency is attributed to the feature that only when the

[11] Here we assume that the information on spare capacity sharing has been piggybacked on the information of working and protection routes, which hence underestimates the control overhead of SBPP and will not affect the effectiveness of the comparison between SBPP and PWCE (if PWCE is found to require a lower control overhead).

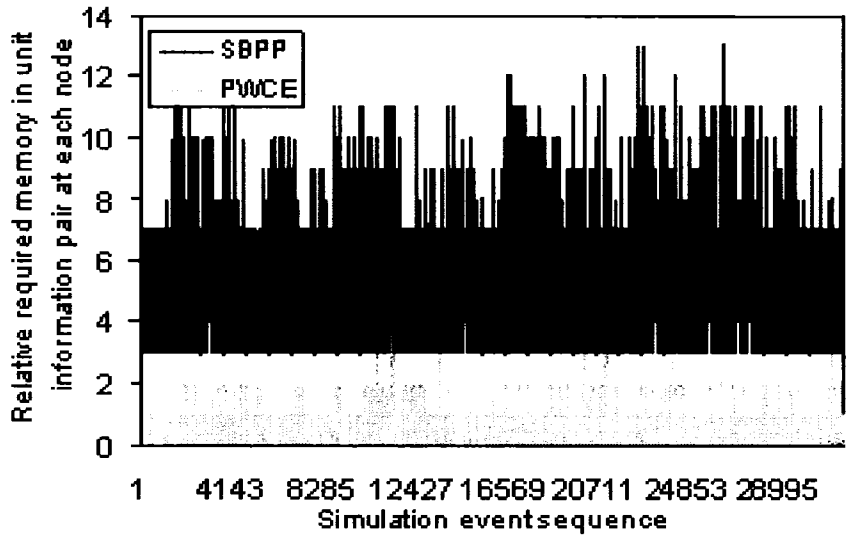
capacity of a span is exhausted or available again should an LSA be flooded. The smaller message size exists because each time PWCE needs to flood only *one* information pair when needed, while SBPP always needs to flood *a sequence of* information pairs (i.e., $W+P$ units) for each state change of a connection. As a quantitative measure to reflect the relative amount of control overheads, we counted the signaling in terms of the basic overhead unit of an information pair above.

The three test networks, including SmallNet, NSFNET and COST239, were investigated in the context of design model B. Results are shown in Figures 7.10(a), 7.10(b), and 7.10(c) respectively. The x-axis shows simulation event sequence, while the y-axis shows the amount of control overhead in unit of information pairs. The simulation event sequence corresponds to all the arrival or departure requests. For each event, the y-axis records the directly-resulting amount of control signaling for network state updating. The data of SmallNet was collected when the traffic load on each node pair is 2.0 Erlangs, NSFNET was studied when the traffic load is 2.4 Erlangs, and COST239's traffic load is 1.2 Erlangs. From the graphs, it is readily seen that SBPP requires a much higher control overhead than PWCE. For example, in SmallNet, the control overhead of PWCE is on average about 0.1 information pairs disseminated per event, while the corresponding value of SBPP is about 3.9, a factor of ~40 times higher. Similar observations can be made for the other two test networks. For NSFNET and COST239 the difference is a factor of ~100 times and ~20 times, respectively. From the above results, it appears that under a sparser network, PWCE can save more control overhead. This is reasonable since in a sparser network, the working and protection routes are normally longer than those in a denser network, which requires a relatively larger value of $W+P$ for each LSA message under SBPP. Thus, from the control-overhead-saving point of view, a sparser network seems to have more advantages than a denser network. Nonetheless, when we consider blocking performance of PWCE later, it will be found that only when the average network nodal degree exceeds a certain threshold level can PWCE outperform or perform close to SBPP in the blocking performance. In the NSFNET test case, it will be found that PWCE cannot outperform SBPP. This therefore forms a tradeoff situation between control overhead and blocking performance.

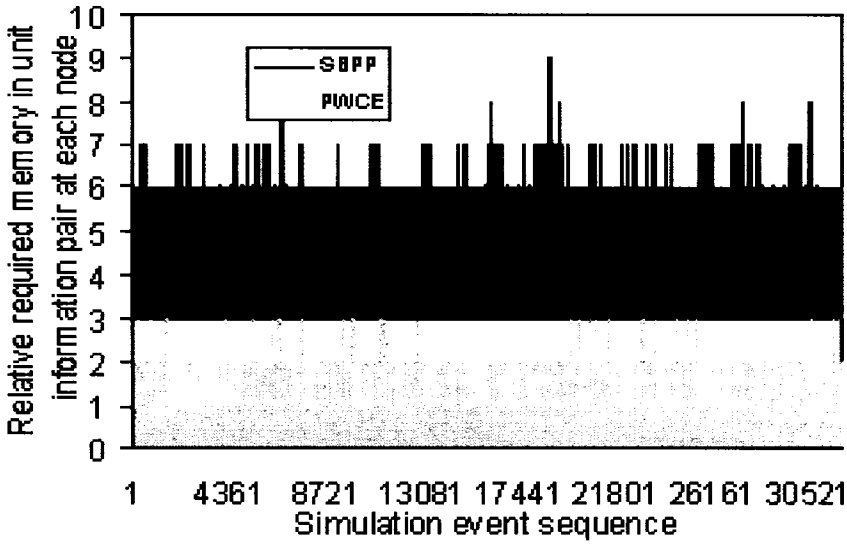
In interpreting these data it should be noted that, low as it is, the PWCE signaling volume also depends on the blocking probability of the simulation, which in turn depends on the total Erlang load of random demand relative to the physical capacity of the test case networks. Indeed, as blocking becomes negligible, PWCE signaling (for state update) vanishes in the limit. To first order, however, SBPP update signaling load is not responsive at all to blocking level (and thus to relative offered load to capacity ratio). Only at very high blocking levels would the effect be noticed—more blocked connections means less state update because fewer connections are actually established relative to offered requests. It follows, therefore, that while these results experimentally demonstrate and validate claims of PWCE signaling reduction relative to SBPP (20 to 100 times), the reader should appreciate that in fact any desired ratio between the two could actually be obtained experimentally because as the relative load of the test case scenarios lowers, to very low blocking, PWCE update signaling will in fact vanish.



(a) SmallNet, 2.0 Erlangs Uniform Traffic Load between Each Node Pair



(b) NSFNET, 2.4 Erlangs Uniform Traffic Load between Each Node Pair



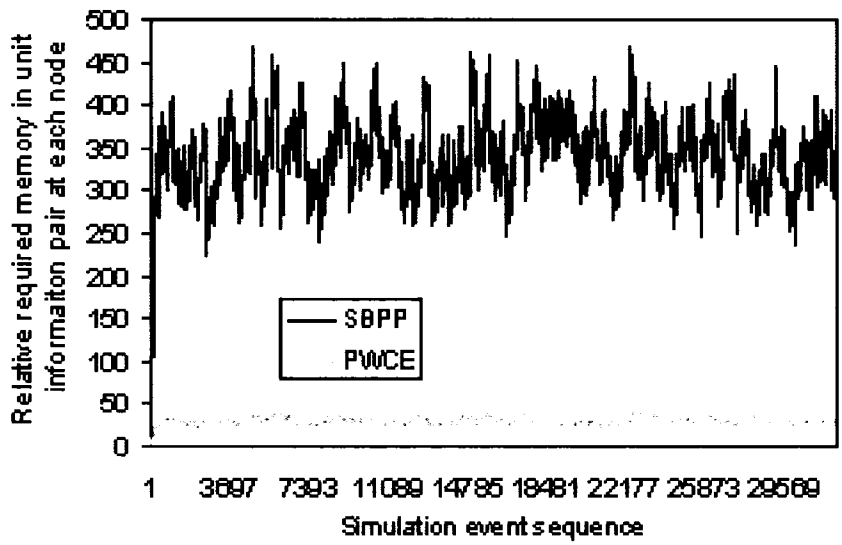
(c) COST239, 1.2 Erlangs Uniform Traffic Load between Each Node Pair

Figure 7.10. Control Overhead Comparison between SBPP and PWCE.

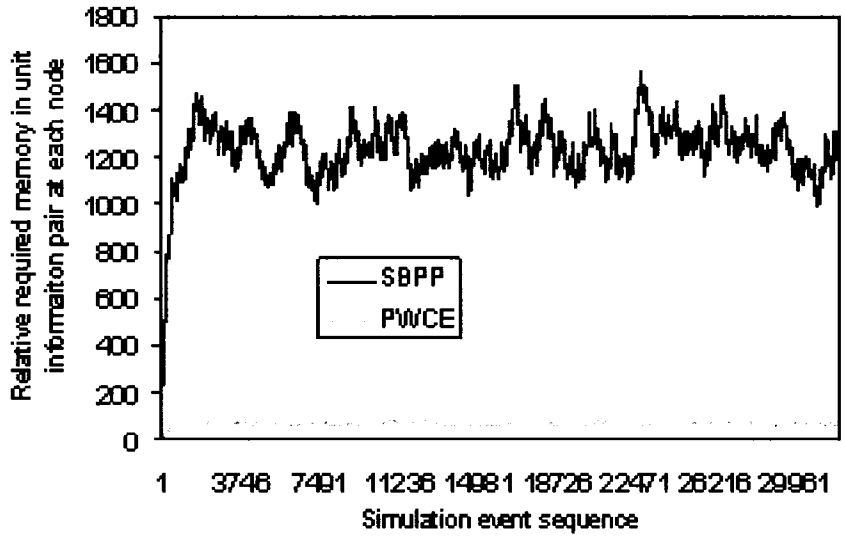
In a network control system such as GMPLS, all the network state information is stored at each local node, so that whenever a new connection request arrives, the node can make an independent decision to determine whether or not a new connection can be established. For SBPP, the network state information includes the states of *all* the connections in the network, of which each consists of a pair of working and protection routes and the corresponding spare capacity sharing relationship. In addition, the state database needs to maintain the information on the network topology, such as which span

capacity has been used up, and which one is still available. In contrast, under PWCE each node only needs to maintain the information on the connections started or terminated at the node itself and the OSPF-like span availability information. PWCE does not need to store any information related to the connections that are not started or ended at the local node itself, and it is also not required to store any information related to the protection routes because under PWCE no protection routes exist within the envelope.

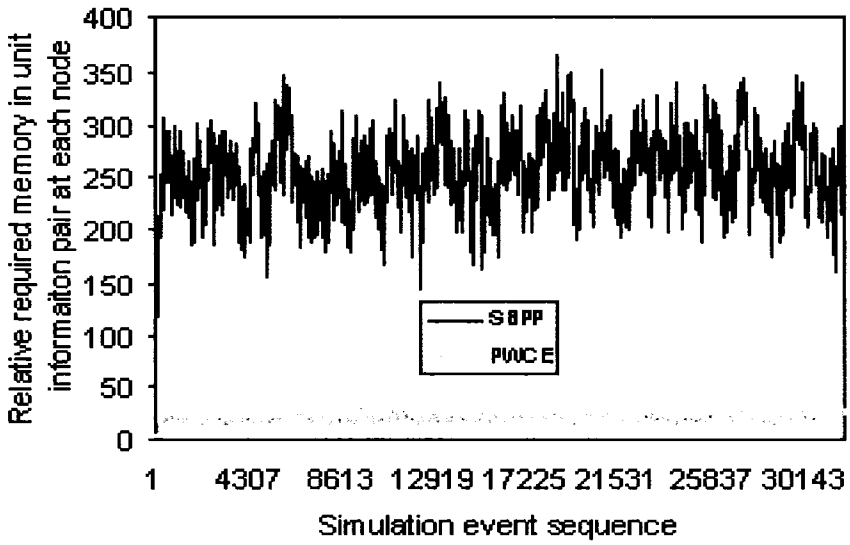
To measure the required network state memory, we continue using the information pair (Span ID, Status) as our basic unit. From the same experiments, results were obtained and are shown in Figure 7.11. The x-axis is the same as before, but now the y-axis records the average network state memory required at each node. Under SBPP, a connection with W-hop working route and P-hop protection route can be measured to require a memory of $W+P$ information pairs. Similarly, under PWCE, a connection with W-hop working route (no need to store any information related to protection routes in PWCE) is estimated to require W information pairs in memory. From the figure, it is readily seen that SBPP requires a much higher memory at each node. On average the differences are about 10 times for SmallNet and COST239, and in the sparser NSFNET, the difference is more than 18 times. The same reasoning can be used to explain the difference. This is the case because a sparser network normally has a larger value of $W+P$.



(a) SmallNet, 2.0 Erlangs Uniform Traffic Load between Each Node Pair



(b) NSFNET, 2.4 Erlangs Uniform Traffic Load between Each Node Pair



(c) COST239, 1.2 Erlangs Uniform Traffic Load between Each Node Pair

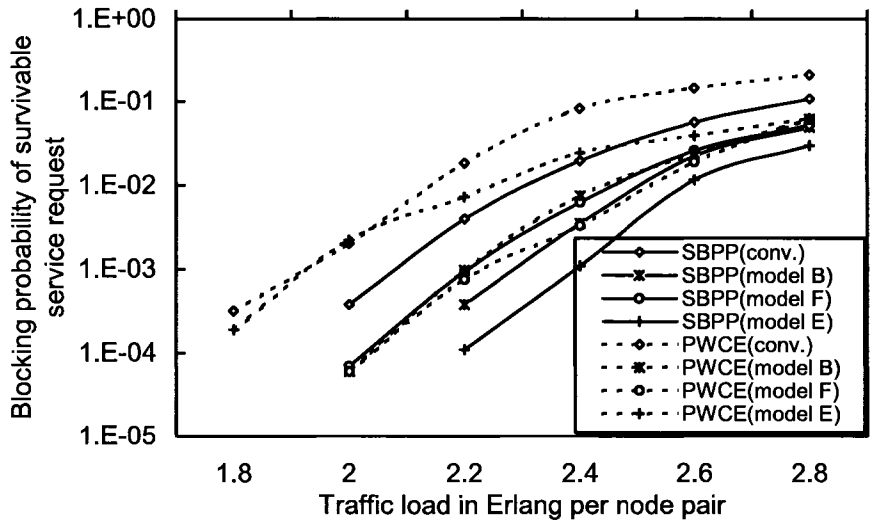
Figure 7.11. Required Network Status Memory at Each Node: an Approximate Comparison between PWCE and SBPP.

7.6.3. Comparative Blocking Performances of SBPP and PWCE

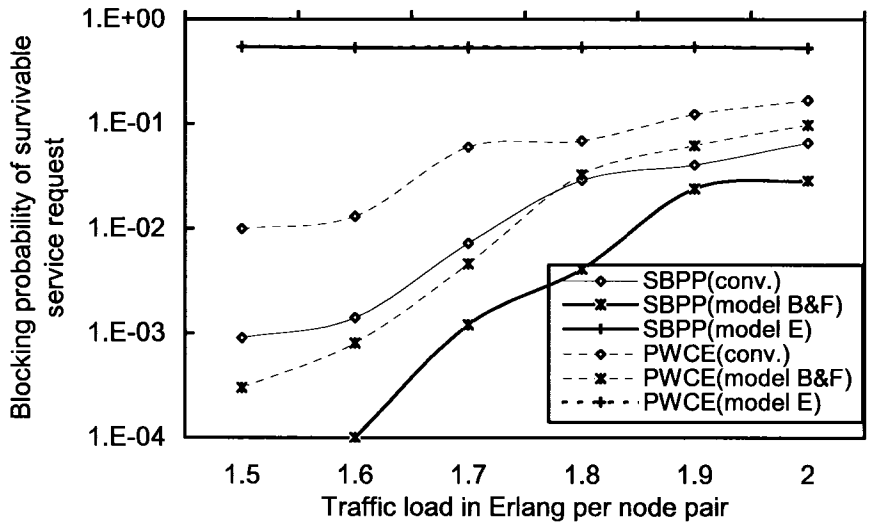
In this section, we report the blocking performances for PWCE and SBPP. Only the results of the hop-based shortest path routing algorithm are considered for PWCE. The LL algorithm that can achieve better blocking performances than the hop-based shortest path algorithm will be discussed in the next section, where the effect of routing algorithm on the network blocking performance will be addressed in detail. It is found that both

PWCE and SBPP can achieve a comparable blocking performance, which thereby means that the operational simplicity of PWCE does not cost any blocking performance degradation compared to SBPP.

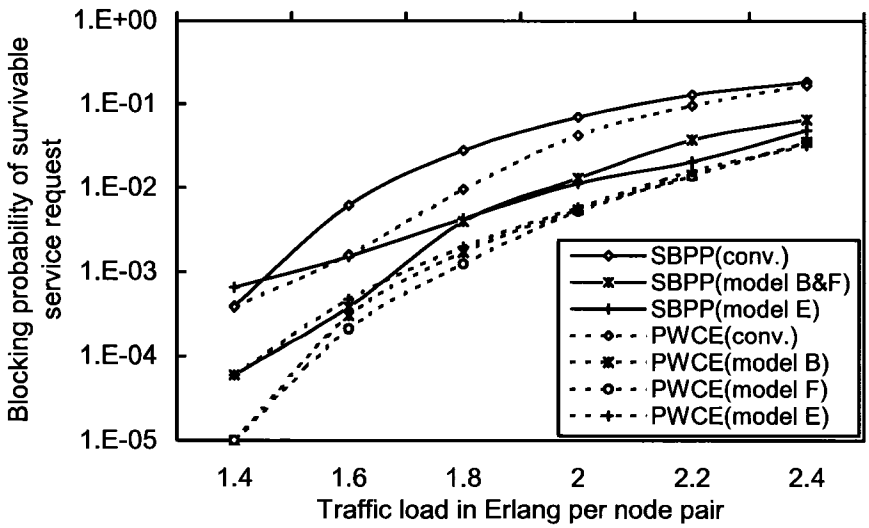
Figure 7.12 shows the simulation results for the test networks NSFNET, ARPA-2, SmallNet, and COST239 under various design cases and the PWCE and SBPP provisioning methods. In the legend, "conv." denotes the network based on the conventional design, and "models B, E, F" denote the networks designed by models B, E, and F respectively. Figure 7.12(a) shows the blocking probabilities of the NSFNET network. It is found that SBPP outperforms PWCE for each of the design cases. The same happens to the ARPA-2 network (see Figure 7.12 (b)), in which the blocking probabilities of SBPP are lower than PWCE. However, the results of the more highly connected SmallNet and COST239 network (average nodal degree are 4.4 and 4.7 respectively) shown in Figures 7.12(c) and 7.12(d) are the other way around. Here, PWCE outperforms SBPP in all the design cases. The performance difference is so pronounced that in Figure 7.12 (d), the curves of the PWCE-based schemes (i.e., PWCE models B, E and F) are clearly grouped together below the other SBPP-based design cases. We can explain these results as follows:



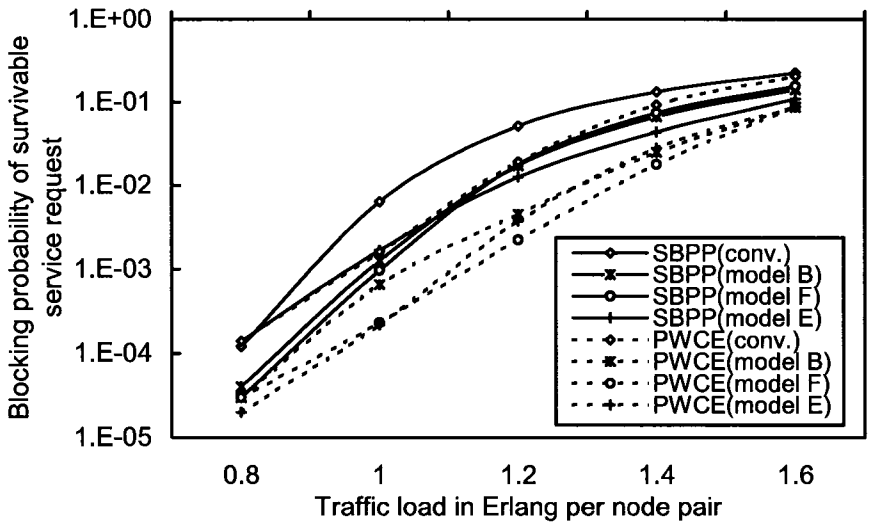
(a) NSFNET



(b) ARPA-2



(c) SmallNet



(d) COST239

Figure 7.12. Blocking Performance of PWCE-Based and SBPP-Based Provisioning Schemes Based on Hop-Based Shortest and the FF Algorithms in the (a) NSFNET, (b) ARPA-2, (c) SmallNet, and (d) COST239 Networks. All Simulation Rounds=10^5 Except that ARPA-2 Has 10^4 Simulation Rounds for Each Testing Point.

The differences between SBPP and PWCE provisioning methods seem to lie at the heart of the experimental behaviors observed. One set of considerations relates to the basic protection mechanisms that they employ. The other is countering considerations about the scope of optimality inherent to each architecture under random arrivals.

SBPP is a path-oriented protection mechanism, while *p*-cycles are a span-oriented protection mechanism. A path-oriented protection mechanism normally has a wider spare capacity sharing scope than a span-oriented protection mechanism. Therefore, in this aspect the SBPP provisioning method can more widely share spare capacity during provisioning process than the PWCE provisioning method. On the other hand, the PWCE method always involves a more global optimal relationship between the spare capacity and protected working capacity even during the provisioning process. After a PWCE is constructed, the protection capacity is never involved in any direct operation (or change) related to the dynamic survivable service provisioning and thus cannot evolve incrementally into a poor global configuration under random (statistically stationary) demand. In contrast, the optimization of SBPP is greedy in nature, on an incremental per-connection basis. For each incoming survivable service, the SBPP provisioning process

finds the first shortest route to establish a working path and second shortest route, which is link-disjoint from the first route, to establish a backup path. While setting up the backup path, although it is allowed to maximally share spare capacity en route, such sharing can still be only a local optimization of the situation, and may be sub-optimal in the wider sense of how it relates to the next several arrivals or departures. It is impossible to reconfigure other existing connections or even foresee the future connections to make a network-wide optimization. PWCE is, however, by its nature free of this effect. Its working-to-spare relationships are fixed and built-in, determined once with a global optimization based on an at least representative view of the overall pattern of relative demand intensities.

In summary, there are two counteracting effects, i.e., *inherent optimality* and *scope* of spare capacity sharing that affect overall network performance. SBPP shows advantage in the scope of spare capacity sharing, while PWCE can ensure the network-wide optimality of spare capacity sharing during the whole provisioning process. Thus, all the results reported for the test networks are in fact the consequences of the interaction, or tradeoff of the above two effects. In the COST239 and SmallNet networks, where the relative benefit of wider sharing scope of SBPP is weakened due to the high connectivities of the networks, the effect of network-wide optimality overwhelms the effect of sharing scope. This in turn causes the *p*-cycle-based PWCE provisioning method to display a lower blocking probability. In contrast, due to the relatively low connectivities, in the NSFNET and ARAPA-2 networks, the wider sharing scope of SBPP demonstrates a stronger effect on the network blocking performance than the network-wide optimality of PWCE. This therefore leads the SBPP provisioning method to achieve a better blocking performance than the PWCE provisioning method.

Based on the above observation, it is clear that the blocking performances of the SBPP and PWCE provisioning methods are related to the network connectivity. To further validate this reasoning and to study how the respective blocking performances of SBPP and PWCE are affected by the graph connectivity, we designed a series of artificial network topologies based on a 10-node ring network as an initial topology as shown in Figure 7.13. Step by step we added spans to the network to gradually increase the network nodal degree. Because the number of nodes in the network is ten, any addition of

a span will increase the network nodal degree by 0.2. Starting from the ring network with nodal degree 2.0, we added ten spans to the network until the network nodal degree is 4.0 and generated a total of eleven network topologies. We assumed two units of uniform forecasted demand on each node pair. We designed PWCEs for these networks based on the previous volume-maximized models and ran simulations for them. We have provided the blocking performances for PWCE and SBPP under the design of model B in Figure 7.14. Indeed, it was found that at a low nodal degree, SBPP achieves a better blocking performance than PWCE, but with the increase of the nodal degree, the blocking performance of PWCE relative to SBPP improves progressively and outperforms SBPP when the network nodal degree is equal to 3.2 or higher. This is quite in line with the results that we have obtained for the above three specific test networks and verifies the above interpretation. NSFNET has an average nodal degree of 3.0, in which the performance of SBPP is better than PWCE. SmallNet and COST239 both have average nodal degrees larger than 4.0, in which the performances of PWCE are better than SBPP. Practically speaking, however, what is of most significance here is simply that in no case is PWCE blocking very different from SBPP. From a practical operating standpoint, if the blocking is on essentially the same order of magnitude, then the differences in blocking hardly matter—it means that all the other advantages, such as operational simplicity of PWCE, can be accessed for their own purposes and benefits.

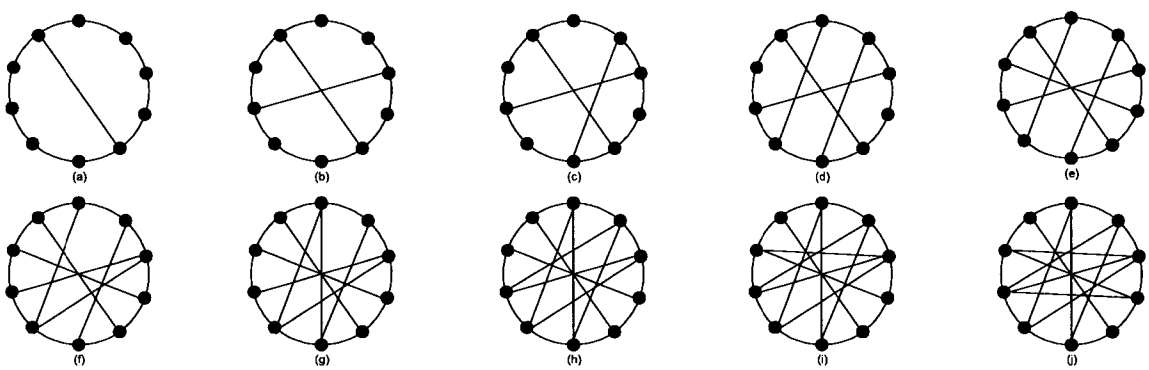


Figure 7.13. Derived Topologies from a 10-Node Ring Network.

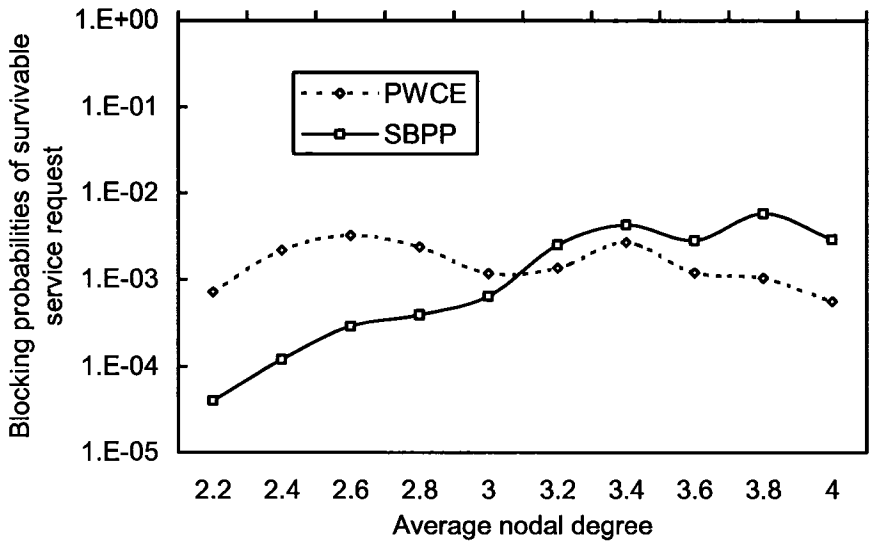


Figure 7.14. Blocking Performances of the Derived Network Topologies under Different Nodal Degrees. Comparison under the PWCE Model B Design and Two Units of Uniform Forecasted Demand on Each Node Pair.

In addition, as conjectured earlier, the similarity between the forecasted demand intensity pattern and actual assigned network capacity can strongly affect blocking performance if total PWCE volume does not suffer greatly. To confirm this, by running experiments for models E and F we observe the effect of the correlation of structuring on the blocking performance. It was found that model F always outperforms model E to achieve a lower blocking probability even though the network designs of model E have lager PWCEs than those of model F in all the test networks. This confirms that the structuring effect is important to PWCE design.

Finally, in Figure 7.12(b), we see that ARPA-2 displays the same blocking performance under models B and F. This is due to the same envelope designed by both of the models as shown in Figure 7.7(b). In addition, model E shows extremely high blocking probabilities under both PWCE and SBPP. This is the case because the awkward design of model E makes nodes 3, 4, 8, and 11 isolated from all the other nodes and all the requests related to them are blocked. Also, for the SmallNet network, we see that SBPP has the same blocking performance under both model B and model F [see Figure 7.12 (c)]. Again, this can be attributed to the fact that the two models have created envelopes, similar in both volume and structure [see Figure 6.8 (b) and (d)].

7.6.4. Impact of LSA Flooding Threshold Levels on Control Overhead and Blocking Performance

This section provides research results on how the frequency increase of LSA flooding will improve the blocking performance of PWCE. Overall, it is found that a higher flooding frequency allows for a better blocking performance. Nonetheless, there exists a threshold point, after which the further increase of LSA flooding would not bring much blocking performance improvement.

In the above results for PWCE, LSAs are emitted only when the envelope capacity of a span was strictly exhausted or became available again. In addition the hop-based shortest path routing algorithm was used to search working routes through the PWCE. The advantage of this is that we can save much control overhead by assuming each span is always available on the topology without flooding any LSA message to update the current capacity usage level on each span. Nevertheless, searching for the working route simply based on hops within non-exhausted spans may not achieve the best blocking performance. This is the case because there is no sense in which that form of routing tries to "hedge" against using relatively heavily-loaded spans. The idea in this portion of the study is to use more fine-grained information on relative capacity usage on each span to compute the routing cost. In other words, the "shortest" route will now appear to be the least-loaded path between source and destination, also known as the Least Load (LL) algorithm [ShBo01b]. The cost of this is more control overhead to update the status of network capacity usage more frequently. This constitutes a tradeoff between control overhead and network blocking performance. With a lower control overhead, the blocking performance may not be the best. However, to improve the blocking performance by using the LL algorithm, more control overhead is required.

As already addressed, the hibernating and real-time LSA flooding strategies can be regarded as two special cases of the threshold-based flooding strategy. In this section, we study how the different flooding strategies affect the performance of PWCE in terms of control overhead and blocking probability. We want to see whether there exists such a situation that a marginal increase in control overhead can generate a significant improvement in blocking performance. Detailed experiments were designed as follows.

Given the envelope capacity on a span designed by model B, we assign a certain number of threshold levels. Based on the remaining envelope capacity on a span, we determine the usage level to which the capacity belongs using the following equation:

$$Usage \quad level = \left\lfloor \left(c-1\right) \cdot l \Big/ w_i^0 \right\rfloor + 1 \tag{7-9}$$

where c is the current unused envelope capacity on the span and w_i^0 is the total envelope capacity on the span. This divides the envelope capacity into nearly evenly spaced "trip points" at which the utilization state will be updated by an LSA. For example, if w_i^0=9, then with four usage levels an update will occur every time channel usage rises or falls by two. If the remaining capacity is one, two or three channels, it belongs to usage level one; four or five channels belong to level two, etc. The LL routing algorithm searches working routes based on the following usage-threshold link cost criterion:

$$Span \quad cost = 1 - \left(Usage \quad level \Big/ l \right) + \varepsilon \tag{7-10}$$

The term $Usage \quad level \Big/ l$ represents the relative remaining capacity on the span. Eq. (7-10) thus charges a routing cost, which is proportional to the current relative utilization of total envelope capacity on each span. Assuming a route is at or near the shortest hop as well, a route with the minimum sum of span cost means that the route has the lowest capacity utilization and is thus the least congested. It is possible for the same route cost to be seen between say a short moderately-loaded route and a long lightly-loaded route because usage levels are simply summed over all spans traversed. To ensure that the LL algorithm always selects the shorter route in the above situation, a small value ε (=10^{-6}) is added in the definition of span cost.

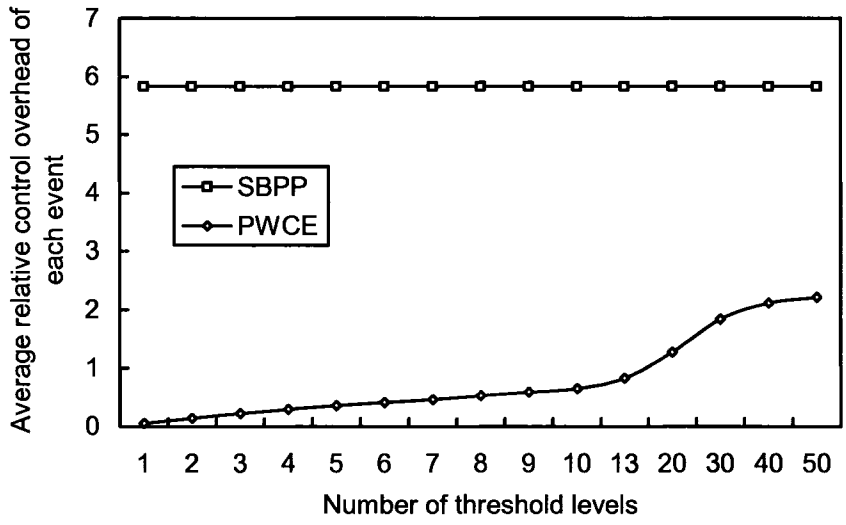
Simulation studies were conducted for the three test networks, which include NSFNET (2.4 Erlang traffic load per node pair), SmallNet (2.0 Erlang traffic load per node pair), and COST239 (1.2 Erlang traffic load per node pair). The required control overheads and blocking were measured as the number of usage levels for reporting was increased. The results are compared with those for SBPP scheme, which was assumed to follow the real-time LSA flooding strategy (any of the service arriving or release events

will result in an LSA flooding). This seems necessary because PWCE can function correctly with any number of usage levels. The effects involved are only matters of performance, and not correct functionality. By its nature, SBPP does not have the same luxury. For error-free operation, it must update the global network state databases after every provisioning change (except, arguably, releases could be batched-up and released in summary form with only a performance loss, not a failure to function correctly). Thus, SBPP has a constant amount of control overhead for different numbers of the threshold levels.

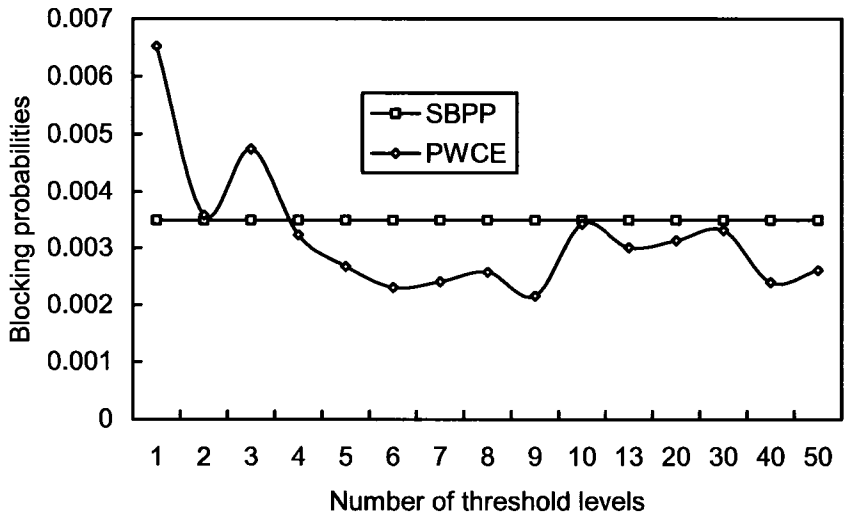
Figures 7.15-7.17 show the results of required control overheads and corresponding blocking performances under different number of usage-reporting levels. Figure 7.15(a) shows the average of required control overhead in unit of "information pair" for each network operational event (arrival or release) in NSFNET. Again we see that in general, SBPP requires a much higher control overhead than PWCE. The difference ranges between 2.5 times and about 100 times when the network is at two extremes. Under PWCE we see that the larger the number of LSA flooding threshold levels, the more the control overhead is required. The difference between 50 levels of thresholds and one level of threshold is about 40 times. This is so because the finer the threshold levels, the more frequently LSA messages are flooded. However, for PWCE we also see that in moving from one usage-reporting level to six, the control overhead increases rather lazily in a linear fashion. In contrast, the blocking performance improvement, in Figure 7.15(b), varies dramatically. When the number of usage levels is greater than four, the PWCE blocking always outperforms SBPP. This was never the case in the prior NSFNET results, no matter which design model or experimental condition is considered. We ascribe this to the usage-sensitive LSA reporting strategy and the additional effectiveness this information gives to the LL route computation method. The net effect is to enable PWCE to outperform SBPP at the cost of a small increase of control overhead. Ultimately, increasing the number of usage levels above six brings no blocking improvements. In Figure 7.15(b) we even see that the blocking probabilities increase when the number of threshold levels lies between ten and forty. As would be expected, this indicates that under threshold-based updating there is a saturation effect at which a limited resolution of usage levels is good enough to achieve blocking performance as good as full real-time

updating. In addition, on the curve of PWCE blocking probability, we see that there is an abrupt jump from threshold level 2 to 3, which seems counteractive to the overall conclusion that with the increase of number of threshold level, the blocking performance of PWCE will be improved. We attribute this "abnormality" to the limit sample size of experimental data. The "jump" can be seen as something like a noise around the overall major trend (i.e., the blocking probability decreases with the increase of number of threshold level). We expect that such a kind of "abnormality" would disappear with the increase of number of simulated arrival events.

A note on the range of levels in the test is warranted. The top number of about 50 usage-reporting levels was chosen because that matches the largest envelope capacity (in channels) of any span in the test-case design. This number of usage levels corresponds to "real time" reporting from that largest span. In most other cases, when the number of levels is high but a span has fewer actual channels than usage levels, the correct interpretation is that the span is reporting updates in real time on every change. In other words, fractional-channel reporting thresholds are not being suggested in practice, but in the simulations their effect is simply the same functionally as reporting per-channel changes, i.e., the real-time LSA flooding strategy.



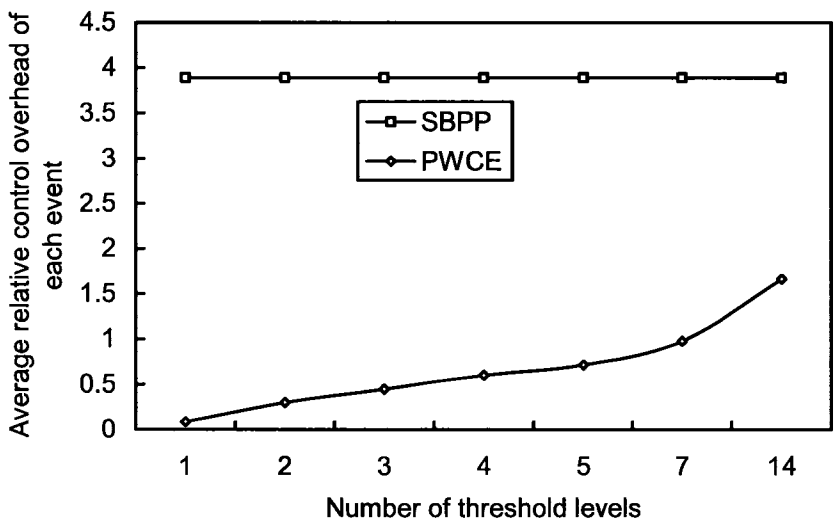
(a) Control Overhead vs. Number of Threshold Levels



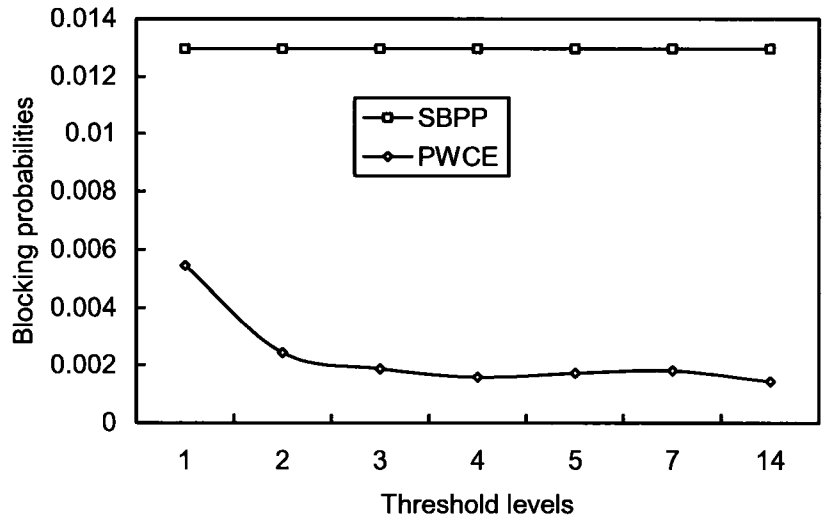
(b) Blocking Performance vs. Number of Threshold Levels

Figure 7.15. NSFNET, 2.4 Erlangs Uniform Traffic Load Between Each Node Pair.

Similar results are also shown in Figures 7.16 and 7.17 respectively for SmallNet and COST239. The only difference from the results of NSFNET is that the blocking performance of PWCE is always smaller than that of SBPP, no matter how many levels of threshold are assigned. We ascribe this to the dense network connectivity, as in the previous analyses.

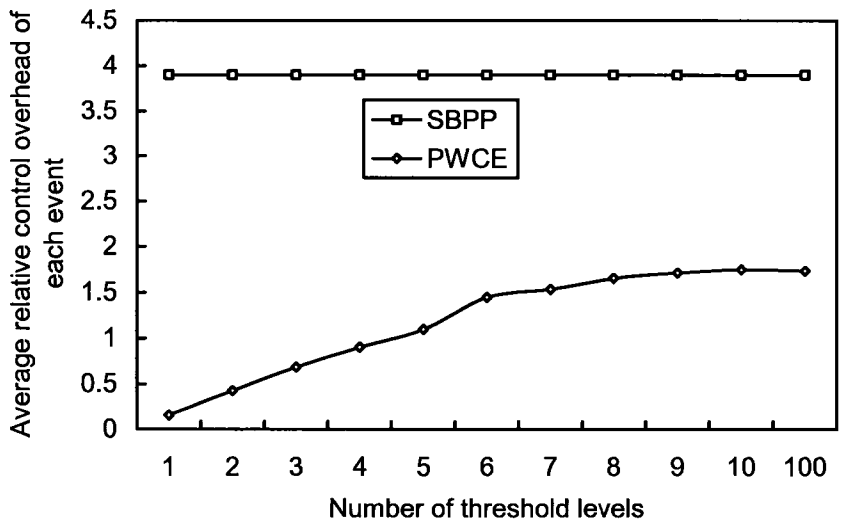


(a) Control Overhead vs. Number of Threshold Levels

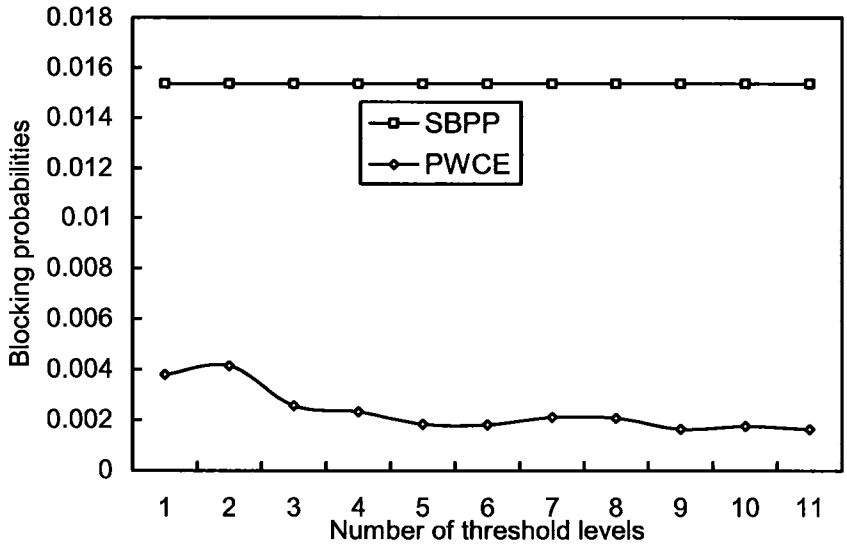


(b) Blocking Performance vs. Number of Threshold Levels

Figure 7.16. SmallNet, 2.0 Erlangs Uniform Traffic Load between Each Node Pair.



(a) Control Overhead vs. Number of Threshold Levels



(b) Blocking Performance vs. Number of Threshold Levels

Figure 7.17. COST239, 1.2 Erlangs Uniform Traffic Load between Each Node Pair.

7.6.5. Tradeoff Effect between Envelope Volume and Shape

In this section, we study how the two aspects of envelope, *structure* and *volume*, will co-affect the overall blocking performance of PWCE. It is found that it seems that under similar envelope volumes, the structuring effect can generate a dominant impact on the blocking performance. The better an envelope matches the target design pattern, the more efficiently the envelope can serve dynamic survivable services.

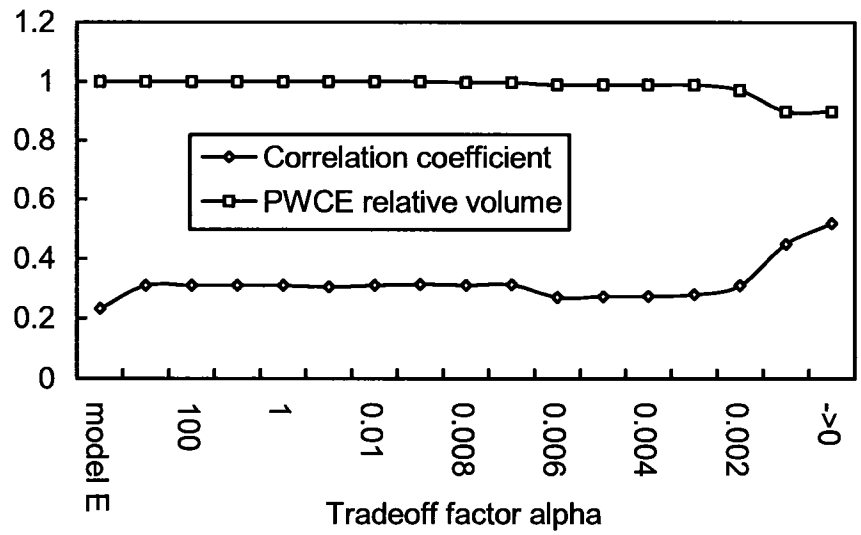
The earlier experimental results for models E and F suggest how the two aspects of an envelope, namely *volume* and *shape*, affect its overall blocking performance. An envelope with a larger volume is expected to serve more dynamic services. However, if the shape of the envelope cannot properly match the network load distribution, the overall networking blocking performance can still be poor even though its volume is large. The results of models E and F confirmed this. Model E has a larger envelope than model F, but the blocking performance of the former is worse than the latter. The underlying reason is that the serious mismatch between the envelope shape of model E and the real network load distribution significantly degrades the network blocking performance. In the previous experiments, we have only looked into the two extreme designs, model E and model F (with $\alpha \rightarrow 0$). Actually, between them there exist many possible

combinations of envelope volume and shape given different tradeoff factors α. Thus, it is important to study how the network performance varies with change in value α. For this purpose, experiments were conducted in the two test networks, NSFNET and SmallNet. Following the same experimental conditions, the tradeoff factor α varies on the range from zero to infinity in model F to design a host of envelopes. For each of the envelopes, simulations were run to obtain the corresponding blocking probabilities.

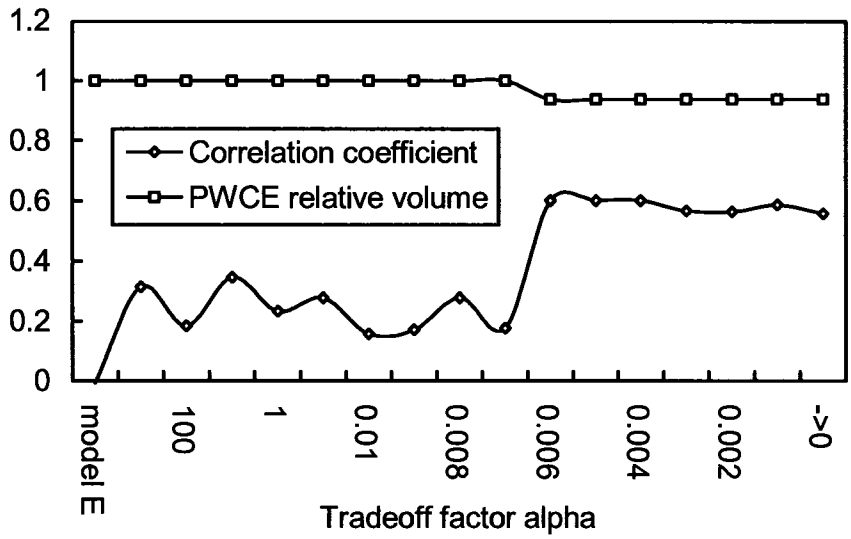
Figure 7.18 plots the relative envelope volumes and their individual correlation coefficients with the forecasted network load template under various tradeoff factors α. The envelope volumes are scaled relative to the envelope designed by model E, which has the largest volume among all the design scenarios. For NSFNET we can see that, from model E (i.e., $\alpha \to \infty$) to α=0.007, all the designed envelopes have the same maximum volume, but with the further decrease of α, the volume of the envelope starts to shrink until it reaches the smallest one at $\alpha \to 0$, which is about 10% smaller than the largest one. However, all the envelopes with the largest volume show different correlation coefficients with the forecasted network load template. The envelope of model E demonstrates the smallest correlation coefficient among all the design scenarios, since it considers volume maximization as the unique optimization objective. However, with the decrease of α, the factor of structure matching starts to show an effect on the envelope design, which causes the correlation coefficient to gradually increase and eventually reach the largest one (when $\alpha \to 0$). An envelope with the largest volume coupled with a good correlation coefficient can be anticipated to show a better blocking performance than others. As shown in Figure 7.19(a), from $\alpha = 1000$ to $\alpha = 0.007$ the envelopes of NSFNET show the largest volume and comparatively high correlation coefficients (about 0.3). Thus, on the three curves, the regions ranging from $\alpha = 1000$ to $\alpha = 0.007$ as shown in Figure 7.19(a) all display lower blocking probabilities than others. This implies that indeed the best tradeoff factors α exist, at which the two counteracting efforts, *volume maximization* and *structure matching*, can be balanced properly to achieve the best blocking performances. Nonetheless, comparing the performance difference between the envelope designed at $\alpha \to 0$ and those within the optimal region, it is likely that the performance improvement is marginal. Therefore, it is acceptable for

us to ignore such a marginal performance improvement to design an envelope directly based on $\alpha \to 0$, because this can save a large number of extra tests dedicated to finding an optimal α, but will not affect the performance significantly. In addition, comparing the overall blocking performances between PWCE and SBPP, we find that in NSFNET, PWCE cannot outperform SBPP either, even within the optimal α region ranging from α=1000 to α=0.007. Nonetheless, the difference between them is reduced.

Similar results can be obtained for SmallNet. As shown in Figure 7.19(b), in the region starting from $\alpha = 0.006$ to $\alpha \to 0$, all the envelopes have much larger correlation coefficients than those in the region starting from model E to α=0.005. Although the envelope volumes between $\alpha = 0.006$ and $\alpha \to 0$ are somewhat smaller than those between model E and α=0.005, they are still quite close, with the percentage difference less than 7%. As such, the overall blocking performances of PWCE appear to be better when the envelopes fall within the region $\alpha = 0.006$ to $\alpha \to 0$, as shown in Figure 7.19(b). In contrast to NSFNET, due to the higher network connectivity of SmallNet, the overall blocking performances of PWCE are found to be better than those of SBPP. Moreover, such a difference is enlarged when the envelopes fall within the region range $\alpha = 0.006$ to $\alpha \to 0$.

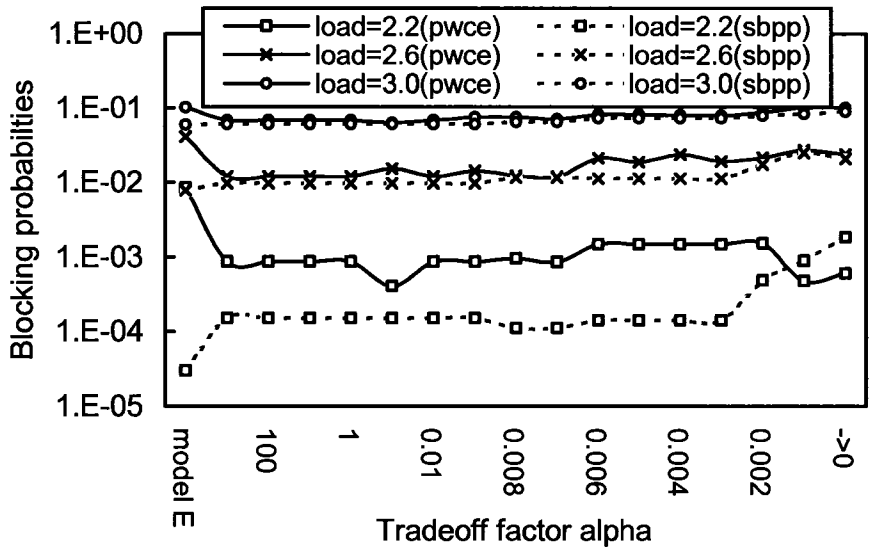


(a) NSFNET

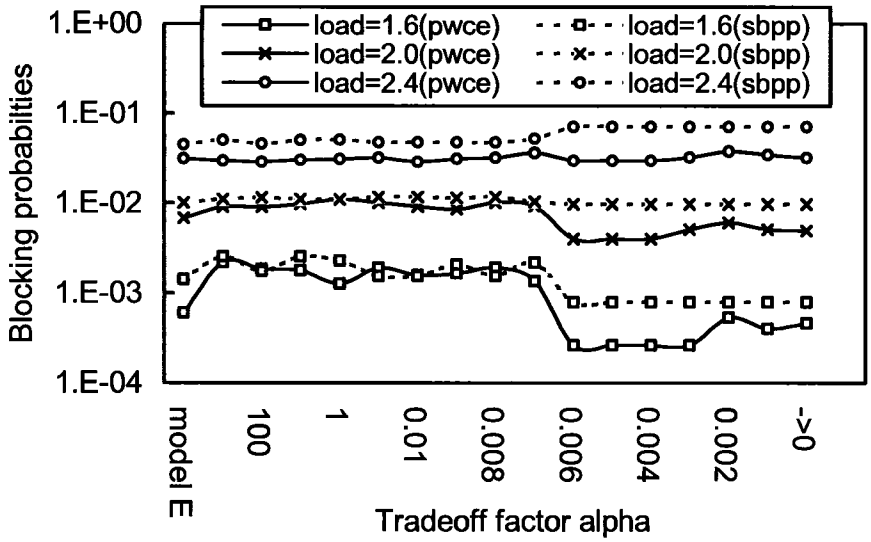


(b) SmallNet

Figure 7.18. The Relative Envelope Volumes and Their Correlation Coefficients with the Forecasted Network Load Template under Various Tradeoff Factors α. The Volumes are Relative to the One Designed by Model E.



(a) NSFNET



(b) SmallNet

Figure 7.19. The Blocking Probabilities of the Envelopes Designed under Various Tradeoff Factors α.

7.7. SUMMARY

We presented p-cycle-based PWCE as an alternative to SBPP for automated provisioning of dynamic survivable services. The combination of the p-cycle technique coupled with the concept of PWCE was motivated by two aspects. p-Cycles are the first protection technique to break the tradeoff between the restoration speed and spare capacity efficiency. They exhibit ring-like switching speed and mesh-like spare capacity efficiency. Using PWCE to provision dynamic protected lightpath services is also a unique technique to date for breaking the tradeoff between operational simplicity and spare capacity efficiency. Operationally, PWCE is even simpler than 1+1 ASP provisioning, but more capacity efficient, since it operates in a shared mesh network.

This chapter has been focused on developing the envelope optimization models, implementing the control infrastructure, and evaluating the performances of PWCE in comparison with the traditional SBPP-based methods. Specifically, the optimization models exploited forcer structure in the conventional designs to maximize envelope volumes, the control infrastructure was realized based on the GMPLS technology, and the performance was evaluated in the terms of restoration speed, operation simplicity, and blocking performance.

The experimental results indicated that, compared to SBPP, PWCE required a much lower control overhead for network state synchronization and significantly less memory for the storage of network state database. Furthermore, the comparison of blocking performance revealed that, although in a sparse network the blocking performance of PWCE was somewhat poorer than SBPP, with the increase of network nodal degree, PWCE performed better to eventually outperform SBPP when nodal degree exceeded about 3.2. In addition, the study on how the number of LSA flooding threshold levels affected the network blocking performance indicated that a small increment of network control overhead to update approximate network states could bring a significant improvement in blocking performance. Moreover, there existed a threshold value of the number of LSA flooding threshold levels, after which the blocking performance improvement was marginal if more LSA flooding threshold levels were added. The tradeoff factor of envelope shape and volume was investigated to find that an optimal tradeoff region existed to balance the above two counteracting aspects, thereby achieving an optimal blocking performance.

To summarize, *p*-cycle-based PWCE design and operational concept may provide considerable advantages over SBPP for dynamic survivable service provisioning with faster restoration speed, better blocking performance, and simpler network operation.

CHAPTER 8
PERFORMANCES OF PWCE UNDER UNIFORM AND MODULAR CAPACITIES[12]

In Chapter 7, we introduced the concept of PWCE and carried out fundamental research on survivable service provisioning under PWCE in comparison with SBPP. It is found that *p*-cycle-based PWCE shows advantages of simpler operation, lower blocking performance, and faster restoration speed over SBPP. Nonetheless, in all these studies, the network capacities were assumed to be heterogeneous (non-uniform) and non-modularized. In this chapter, we will further investigate the PWCE provisioning method under some more strict capacity environments that involves the networks with uniform capacity and modularized capacity.

8.1. DESIGN AND OPERATION OF PWCE NETWORK WITH UNIFORM CAPACITY

In the designs of Chapter 7, we determined the total network capacities for the performance comparison based on the *p*-cycle optimization models, which showed heterogeneous amounts of capacity on spans. We define such a network as a *network with non-uniform capacity*. In this type of network, although both PWCE and SBPP use the same total network capacity to provision dynamic survivable services, the comparison is probably biased towards PWCE. This holds true because the designs are based on *p*-cycles, which may inherently tune the network capacity distribution of the design to be more efficient for PWCE. To be more valid, in this section we will study another capacity scenario: a network with homogenous capacity on each span. Likewise, we define this type of network as a *network with uniform capacity*. Although it may not be as practical as a network with non-uniform capacity, the study of PWCE under such a network can yet ensure the fairness of the comparison between the two provisioning methods. If the resultant findings are analogous to those obtained for the network with non-uniform span capacity, then this study can confirm the conclusions drawn earlier. Additionally, we believe that the investigation will interest many researchers since over these years studies

[12] This chapter contains some material previously reported in [Shen03], [ShGr04c], and [ShGr05b].

on optical networking have been mostly focused on the network with uniform capacity [ChGa92].

For a network with uniform capacity, we specifically considered models C and D because the remaining models are more amenable to the network with non-uniform capacity. The four test networks—ARPA-2, NSFNET, SmallNet, and COST239—were considered, and a total of 16 units (e.g., wavelengths) of capacity were assumed on each span. Various envelopes were designed and their performances evaluated based on the simulations. For PWCE, the hop-based shortest path algorithm and the LL algorithm were employed to route lightpaths, while for SBPP the FF algorithm was used. Particularly, the real-time LSA flooding strategy was applied when running the LL algorithm. To generate the structuring patterns for the model D designs, without losing generality, we assumed that there were uniformly *two* units of forecasted demand between each node pair in each of the four test networks. As before, the demand-splitting shortest path routing algorithm was used to route the forecasted demand matrices, and the resultant working capacity distributions functioned as the structuring patterns or templates for model D.

Figures 8.1-8.4 provide the structural information for the designed envelopes. In all the four test networks, the designed envelopes either by model C or model D have identical volumes, but different envelope capacity distributions. Overall, the envelopes designed by model C (without the structuring effort) are more regular in capacity distribution than those designed by model D (see the results of SmallNet and COST239). Specifically, the working capacity on each span in the envelopes designed by model C is either half of or equal to the total deployed capacity. As a potential benefit, this regularity feature allows the deployment of a network to encapsulate all the working channels in one fiber and all the protection channels in another collateral fiber. This may facilitate network administration and maintenance, and thereby save network operational cost. Nonetheless, the above regularity feature does not imply a superior blocking performance. Conversely, the envelopes designed by model C display blocking performances inferior to those of model D because model C maximizes envelope volumes but overlooks the

structuring efforts. Details of the blocking performance comparison between the two models will be discussed next.

Additionally, as a special case, we find that the designs of models C and D generate an identical envelope for the test network ARPA-2. This can be attributed to the sparse feature of the ARPA-2 topology, which causes the solution spaces of the two designs so similar that they converge.

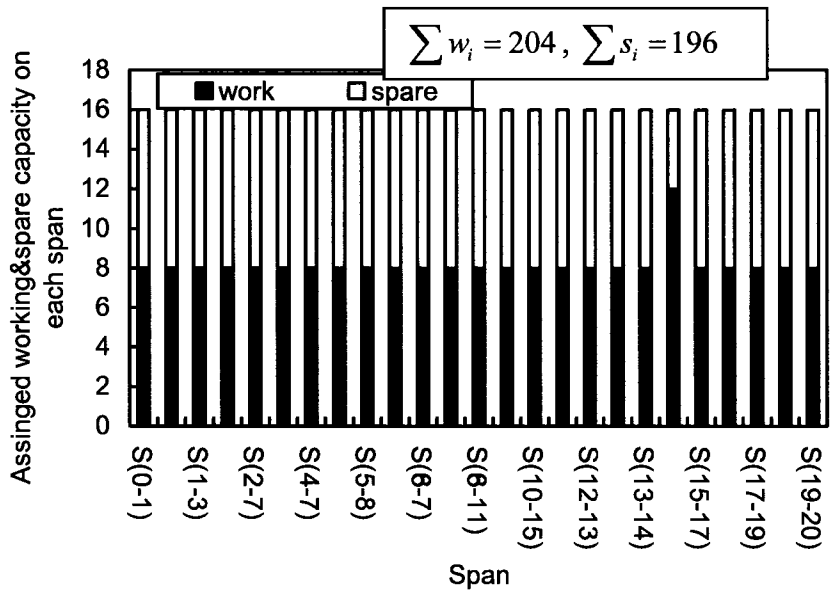
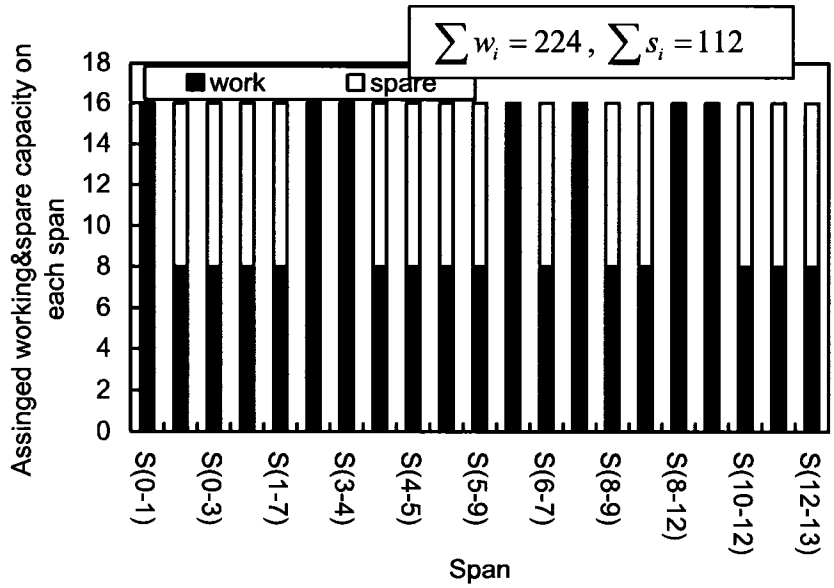
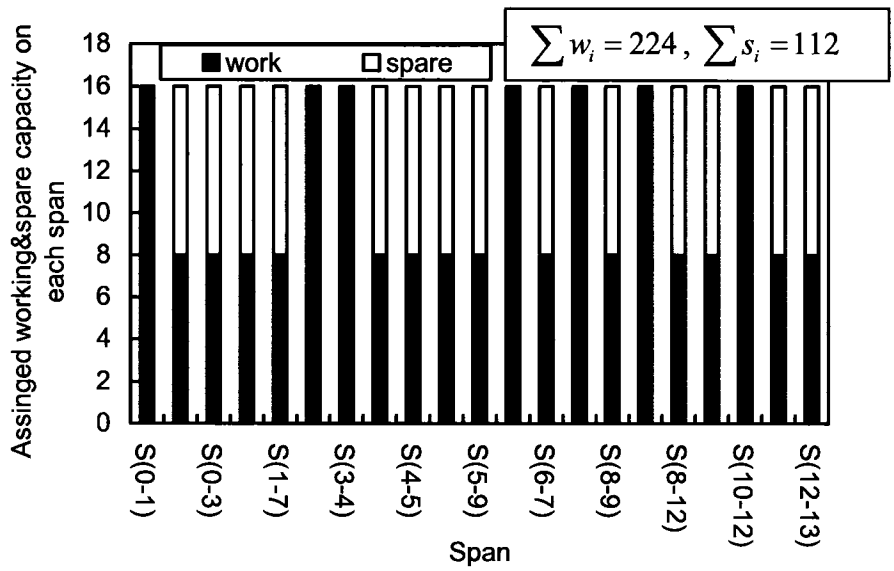


Figure 8.1. Protected Working Capacity Envelope (PWCE) of Model C and Model D for the ARPA-2 Network with a Total of 16-Unit Capacity on Each Span.



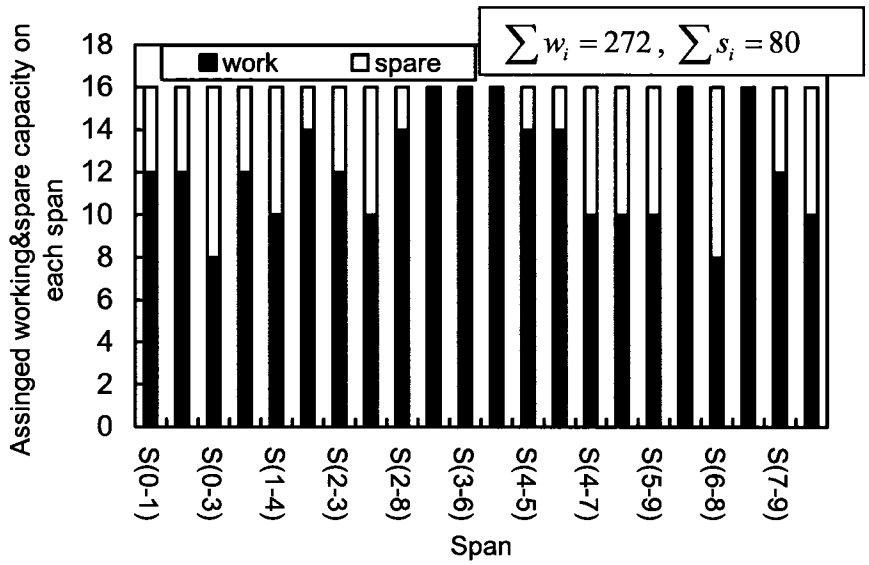
(a) Envelope of Model C (NSFNET)



(b) Envelope of Model D (NSFNET)

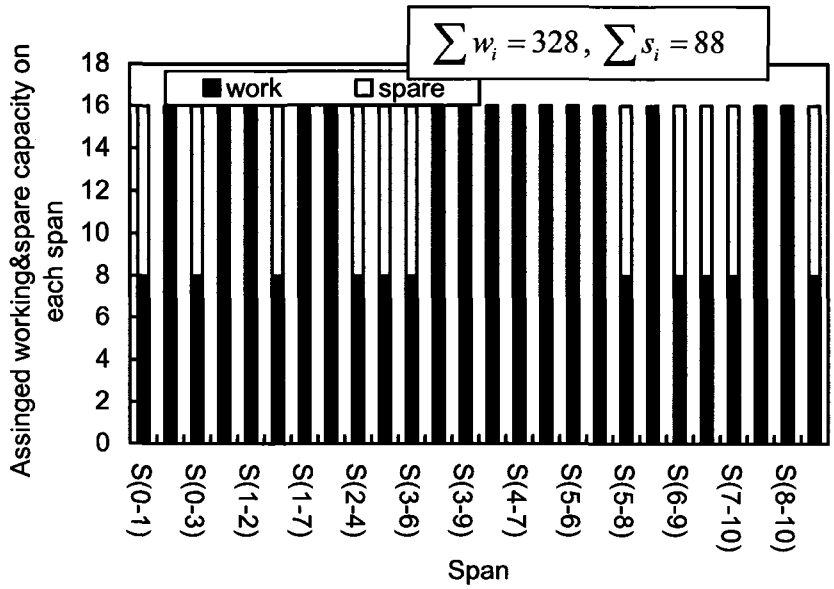
Figure 8.2. Protected Working Capacity Envelopes (PWCE) of Model C and Model D for the NSFNET Network with a Total of 16-Unit Capacity on Each Span.

(a) Envelope of Model C (SmallNet)

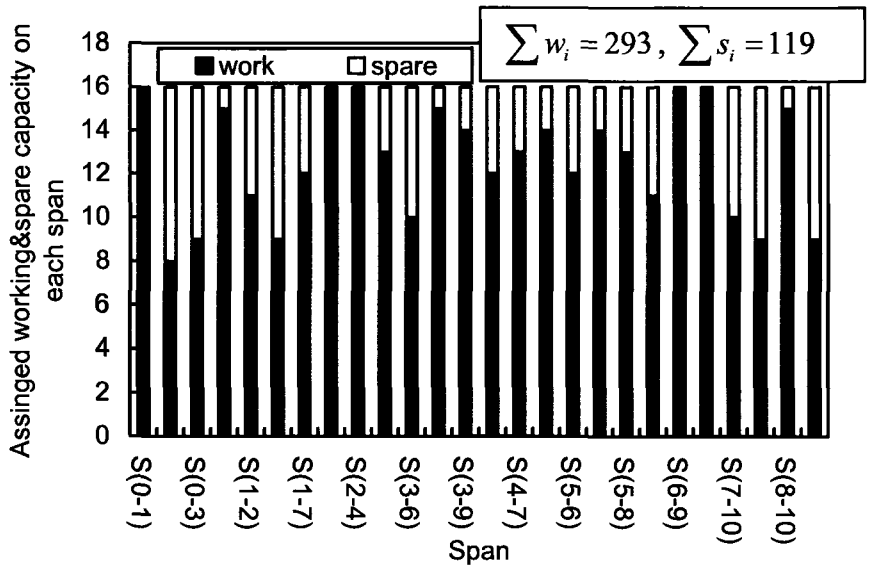


(b) Envelope of Model D (SmallNet)

Figure 8.3. Protected Working Capacity Envelopes (PWCE) of Model C and Model D for the SmallNet Network with a Total of 16-Unit Capacity on Each Span.



(a) Envelope of Model C (COST239)



(b) Envelope of Model D (COST239)

Figure 8.4. Protected Working Capacity Envelopes (PWCE) of Model C and Model D for the COST239 Network with a Total of 16-Unit Capacity on Each Span.

Despite the regularity feature of envelopes designed by model C, these envelopes usually demonstrate inferior correlations with the forecasted structuring templates. Table 8.1 presents the individual correlation coefficients with the initial structuring design templates for the envelopes designed by models C and D. Similar to the network with non-uniform capacity, the envelopes designed with structuring effort (i.e., model D) overall display higher correlations coefficients than those without structuring efforts (i.e., model C). Furthermore, the conclusion drawn for the network with non-uniform

capacity—namely that an envelope with a higher correlation coefficient can achieve a better blocking performance under a comparable envelope volume—also applies to the network with uniform capacity. Overall, the envelopes designed by model D always outperform those by model C (see the results of SmallNet and COST239 in Figure 8.5). However, the results of ARPA-2 and NSFNET are exceptions. Because in ARPA-2 models C and D generate exactly the same envelope, they achieve an identical blocking performance. Similarly, because in NSFNET the envelopes designed by models C and D show close capacity distributions and the same volume, they perform similarly in the blocking performance as well.

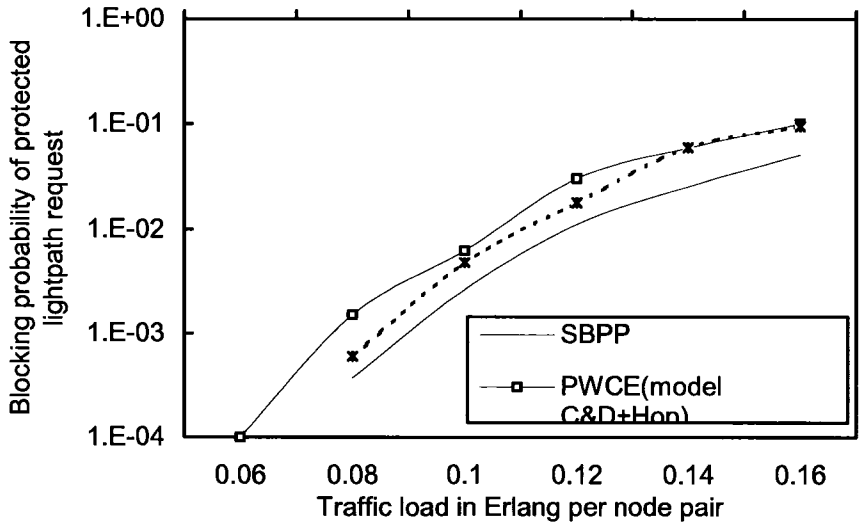
Table 8.1. Correlation Coefficients between the Envelope Structuring Patterns and the Actual Envelope Working Capacity Distributions (i.e., the Shapes of Designed PWCEs) of Models C and D

Network	Model C	Model D
ARPA-2	0.185	0.185
NSFNET	0.344	0.307
SmallNet	-0.046	0.112
COST239	-0.212	0.418

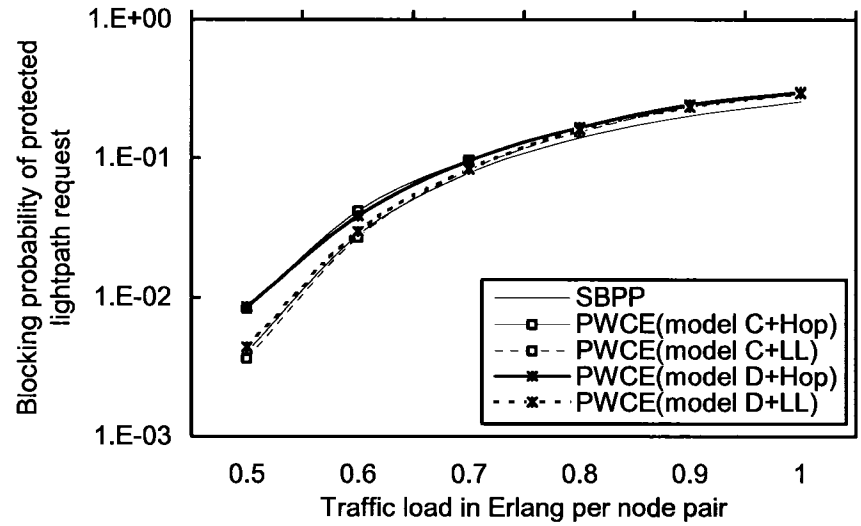
Figure 8.5 compares the blocking performances of PWCE and SBPP. As in the case of the network with non-uniform capacity in Chapter 7, SBPP performs better than PWCE in a sparsely-connected network (e.g., ARPA-2 network). However, with the increase of nodal degree (from ARPA-2 to NSFNET, SmallNet, and COST239), the two schemes tend to perform closely, and finally PWCE outperforms SBPP. Therefore, the same conclusion can be made for the networks with uniform capacity—PWCE performs better when the network connectivity grows denser. The reason for this is the same as the one for the network with non-uniform capacity, which can be attributed to the counteraction between the two factors, namely, *inherent optimality* and *scope* of spare capacity sharing.

Finally, for PWCE we studied the impacts of different route searching algorithms on the network blocking performance. In all the test networks, the LL algorithm outperforms the hop-based shortest path routing algorithm. This is consistent with the results obtained for the network with non-uniform capacity and those in [ShBo01b]. However, in the above comparison we did not consider the effect of LSA flooding frequency (i.e., the

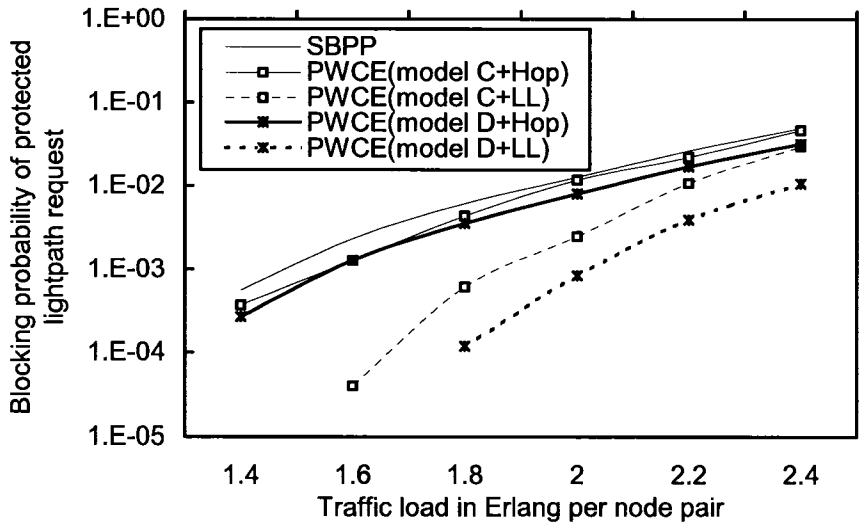
threshold-based LSA flooding strategy) on the blocking performance and the required control overhead. Nonetheless, it can be anticipated that they will perform similarly to the network with non-uniform capacity.



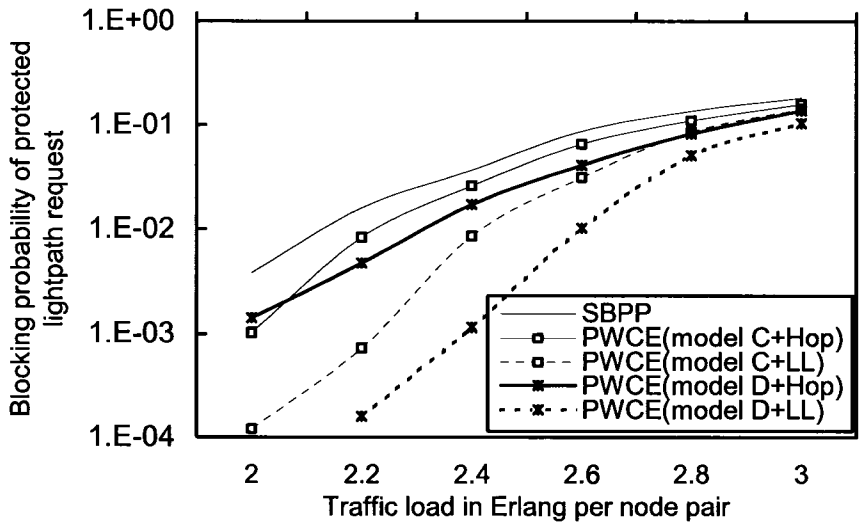
(a) ARPA-2



(b) NSFNET



(c) SmallNet



(d) COST239

Figure 8.5. Blocking Performance Comparison between PWCE and SBPP Provisioning Methods Based on Hop-Based Shortest Path, LL, and FF Algorithms in the (a) NSFNET, (b) ARPA-2, (c) SmallNet, and (d) COST239 Networks with a Total of 16-Unit Capacity on Each Span. Simulation Rounds=10^5 except for ARPA-2, of Which Simulation Rounds=10^4.

8.2. Design and Operation of PWCE Network with Modular Capacity

Thus far, we have assumed that the network capacity is integrally continuous, not modularized. However, in many cases the deployed capacity is modularized such as OC-

12, OC-48, and so on. The aim of this section is to study how capacity modularity affects the performance of PWCE.

Specifically, given a module set and the cost of each module, we modularize the capacity for a design *without modularity* aiming to minimize the total network design cost. After modularization, it is likely that the modular capacity is often over-provisioned, greater than the one before the modularization. For example, the capacity of a design before modularization is five units. After modularization, an eight-unit module may be assigned to accommodate the capacity. As a consequence, three units of capacity are over-provisioned. Two possible strategies can be adopted to cope with the over-provisioned modular capacity. The first strategy leaves the over-provisioned portion of the modular capacity unattended, or uses it to transmit low-priority traffic such as best-effort applications. In contrast, the second strategy exploits the over-provisioned portion to carry more PWCE services by repartitioning the modular capacity to construct a larger envelope. Regarding the second strategy, an extra step is required to re-split the total modular capacity into working and spare units again. Two possible models may be adopted for this purpose. Model D should be applied if the shape of the new envelope must be retained similar to a structuring template. Otherwise, to merely maximize the envelope volume, model C is sufficient. Generally, the repartitioning process can be carried out for both conventional design with modularity and model B volume-maximization design with modularity. For the latter, we first use model B, on the basis of the span spare capacity obtained by the conventional design *without modularity* as a budget, to maximize a PWCE. We then modularize the integrally continuous span capacity obtained from model B. Finally we apply the repartitioning models to exploit the total modular capacity to seek a larger envelope. Figure 8.6 depicts the details of the above procedure.

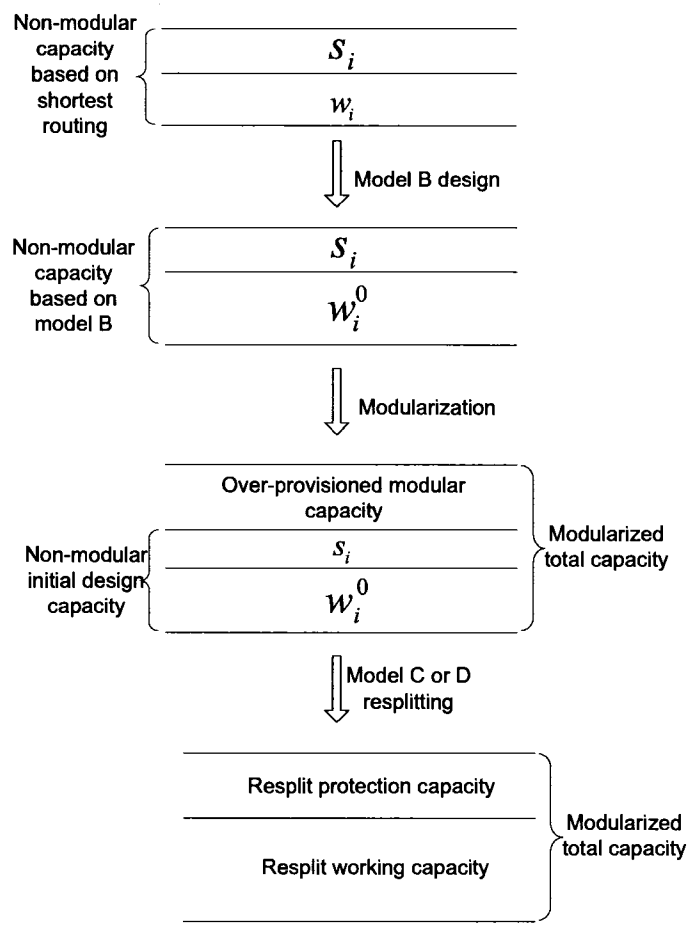


Figure 8.6. An Illustration of Capacity Modularization and Repartitioning Procedure of Model B Design.

8.2.1. Model for Capacity Modularization

The ILP model of capacity modularization is presented next.

In addition to the sets presented thus far, one more set is specifically required for the modularization model:

- M denotes the set of available module types.

Additional parameters are required as follows:

- C_i^m represents the cost of module m on span i. Note that if the network design assumes that the cost of the same type of module on any span is identical, then we can simplify the notation as C^m, which denotes the cost of module m.
- Z^m represents the capacity of module m.

New variables are listed below:

- t_i^m represents the number of module m required on span i.

Finally, the objective and constraints of the modularization model are:

Objective: minimize $\sum_{m \in M} \sum_{j \in S} C_j^m \cdot t_j^m$ (8-1)

Constraints:

$$\sum_{m \in M} Z^m \cdot t_k^m \geq T_k \quad \forall k \in S \tag{8-2}$$

The objective is to minimize the total network design cost after implementing capacity modularization. T_k represents the total integrally-continuous capacity on span k after the model B volume maximization design,. Consequently, constraint (8-2) implies that sufficient modules should be deployed to accommodate all the above integrally-continuous capacity on each span.

8.2.2. Test Description

We assumed that there are five types of modules, namely 4, 8, 16, 32, and 64-unit modules. The cost standard of the modular capacity was assumed to follow a 4-to-2 ratio [DoGr00], and the cost of each module on any span was assumed to be the same. The 4-to-2 ratio reflects the benefit of economic scale of modularity, which means that the cost of a module whose capacity is four times of another module is only twice as expensive as the cost of the lower capacity module. For example, if the cost of a 4-unit module is one, then the cost of a 16-unit module is only two. For the modules whose capacities did not form a four-time relationship, we assumed that the cost ratio followed a square-root relationship with the corresponding capacity ratio. Thus, if the cost of a 4-unit module is $\sqrt{2}$, then the cost of an 8-unit module is two.

As before, we employed the conventional non-modular p-cycle survivable network design as a baseline design, and used its spare capacity as the budget in the PWCE design of model B. The working capacity distribution of the envelope designed by the conventional model $\{w_i, i \in S\}$, which follows the previous demand-splitting routing, was chosen to function as the *structuring pattern* $\{l_i, i \in S\}$ for the shaping purpose, i.e., models B and D.

Experimental tests were run for the NSFNET, SmallNet, and COST239 networks and a series of derived topologies based on a 10-node ring from average nodal degree 2.2 to 4.2 (as shown in Figure 7.13). It was assumed that there are three, three, and two units of forecasted demand on each node pair respectively in the first three test networks. Two units of identical forecasted demand were assumed between each node pair on the latter derived topologies. Three types of design scenarios were considered: (1) conventional design with modularity and model D resplitting (conv-mod-D), (2) model B with capacity modularity and model C resplitting (B-mod-C), and (3) model B with capacity modularity and model D resplitting (B-mod-D). A total of 10^5 arrival events were simulated for each test point.

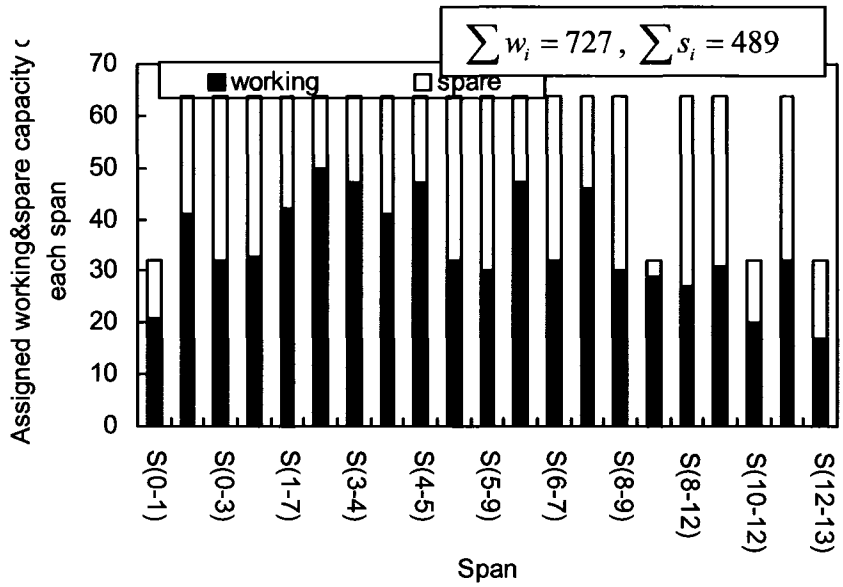
8.2.3. Results and Discussion

A) Structural Properties of Envelopes

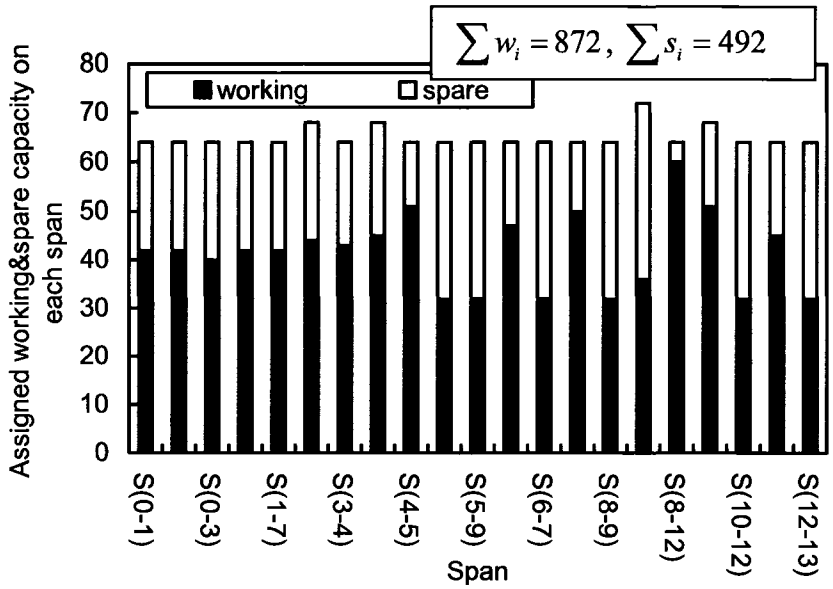
The results in this section provide and compare the structures and volumes of envelope designed by various modular models. It is found that through resplitting, we can always efficiently utilize the over-provisioned portion of modular capacity to augment the volume of envelope, but in the meanwhile, guarantee the structure of envelope to still match a target design pattern.

Figure 8.7 portrays the structures of PWCEs attained by the conventional modular design with model D working/spare (w/s for short) splitting and the model B modular design with models C and D w/s splitting for the NSFNET network. Figure 8.7(a) provides a baseline modular capacity design obtained from the conventional *p*-cycle optimization design followed by model D w/s capacity resplitting. The results in Figures 8.7(b) and 8.7(c) were obtained by employing the model B modular design and then applying models D and C to repartition the modular capacity. The distinction between them is that the result in Figure 8.7(b) considers the structuring effect when resplitting the modular capacity, while Figure 8.7(c) doesn't. We find that in all three design cases, the modularization process yields over-provisioned capacities compared to designs without modularity. This is straightforward, since the capacity of each module is discrete. In order to accommodate an integrally continuous capacity, which may only partially fill the capacity of a complete module, a new module must be deployed, leaving some capacity

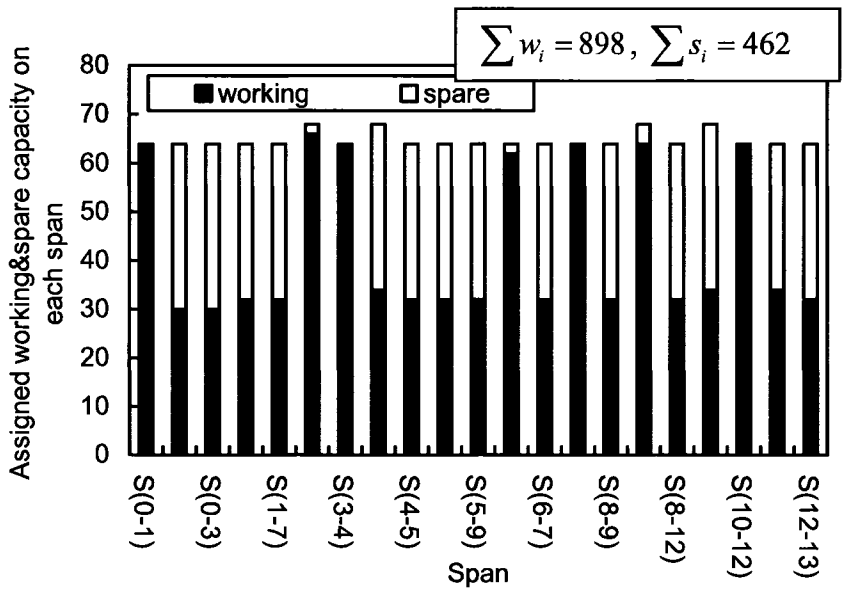
unused. Thus, compared to the capacity without modularity, we see that there are 30% and 20% modular capacity over-provisioned respectively for the conventional modular design and model B modular design. Likewise, for the envelopes there are 40%, 21%, and 25% volume augmentations respectively for the conventional modular design with model D w/s resplitting and model B modular designs with model D and C w/s resplitting. With the same percentage of modular capacity over-provisioned, model C resplitting generates an envelope 4% larger than that of model D resplitting. This can be ascribed to the structuring effort of model D when resplitting the modular capacities, whereas model C merely considers the volume as the unique optimization objective.



(a) Envelope of Conventional Design with Modularity and Mode D Resplitting (NSFNET)



(b) Envelope of Model B Modular Design and Mode D Resplitting (NSFNET)

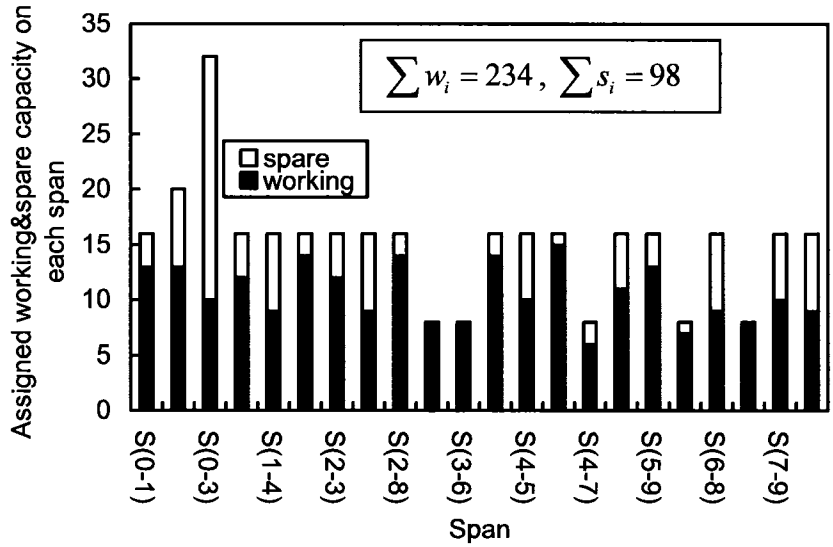


(c) Envelope of Model B Modular Design and Mode C Resplitting (NSFNET)

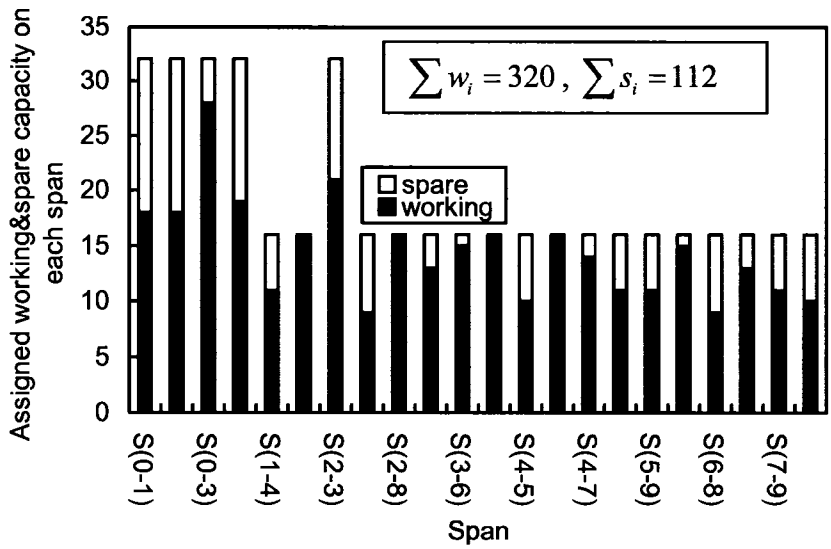
Figure 8.7. Protected Working Capacity Envelopes (PWCE) of the NSFNET Modular-Capacity Network with Three Units of Uniform Forecasted Demand on Each Node Pair.

Now let us consider how the PWCE structures change after implementing modularization design compared to those without modularity. The conventional design with modularity employs model D, and the forecasted working capacity of the conventional non-modular design as the *structuring pattern*, to resplit the modular capacities. Therefore, the new designed envelope always *contains* the envelope of the non-modular design. Similarly, since model B modular design with model D w/s

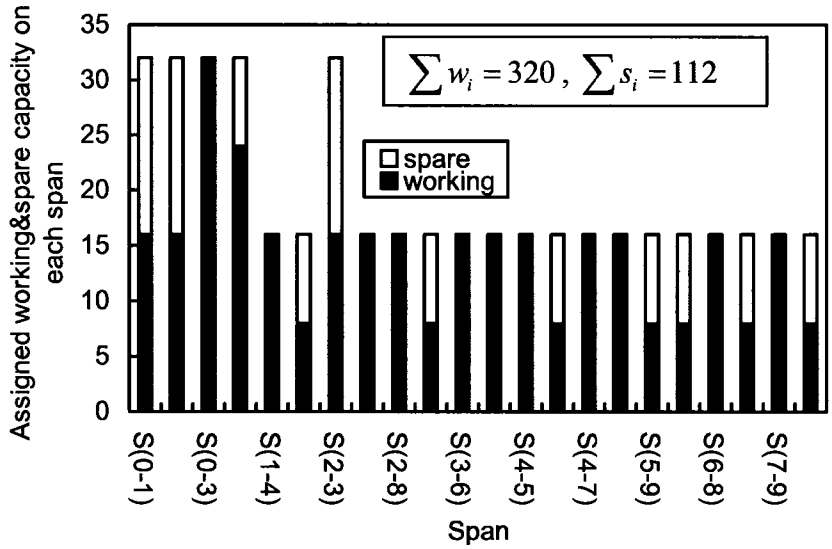
resplitting employs the forecasted working capacity of the conventional non-modular design as the structuring pattern in both model B volume-maximization and model D resplitting steps, the new designed envelope always contains the envelope of the conventional design without modularity. However, for model B modular design with model C w/s resplitting, although the step of model B volume maximization employs the forecasted working capacity as the structuring pattern, without considering the structuring effort in the following w/s re-splitting step, the design yet cannot ensure that the developed envelope always contains the envelope of the conventional non-modular design. For example, in the conventional non-modular design, the envelope capacity on span (1-7) is 35 units [as shown in Figure 7.6(a)], but the corresponding capacity in the modular resplitting design [as shown in Figure 8.7(c)] is only 32 units.



(a) Envelope of Conventional Design with Modularity and Mode D Resplitting (SmallNet)



(b) Envelope of Model B Modular Design and Mode D Resplitting (SmallNet)

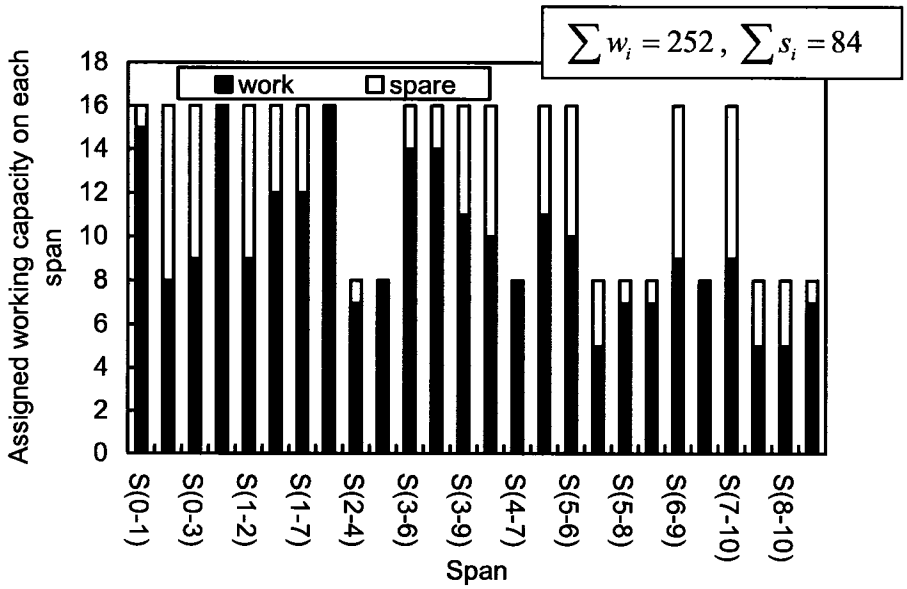


(c) Envelope of Model B Modular Design and Mode C Resplitting (SmallNet)

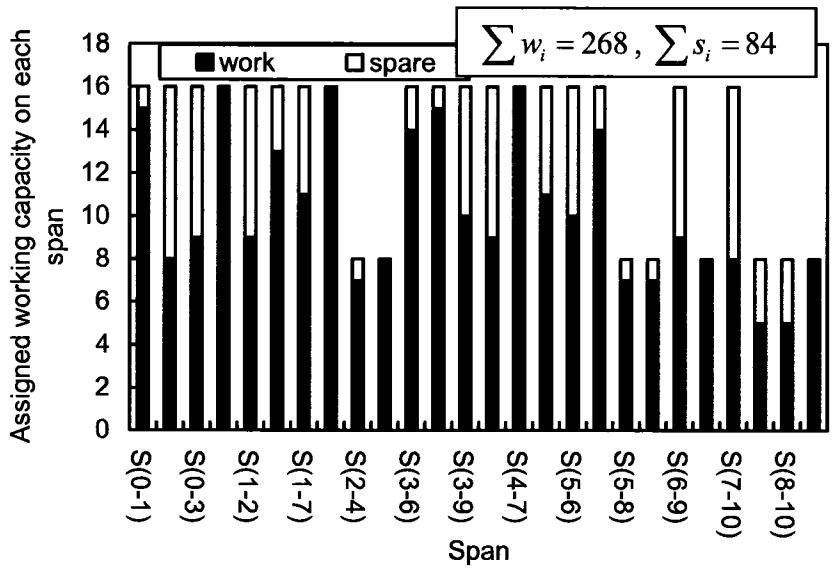
Figure 8.8. Protected Working Capacity Envelopes (PWCE) of the SmallNet Modular-Capacity Network with Three Units of Uniform Forecasted Demand on Each Node Pair.

Similar envelope distributions are displayed in Figures 8.8 and 8.9 for the SmalleNet and COST239 networks respectively. Compared to the corresponding non-modular designs, the designs with modularity are found to overprovision 13% and 29% extra modular capacities for the conventional design and the model B volume maximization design respectively in the SmalleNet network. These two percentages are 46% and 41% in the COST239 network. Also, after modularization, we see that all the envelopes are

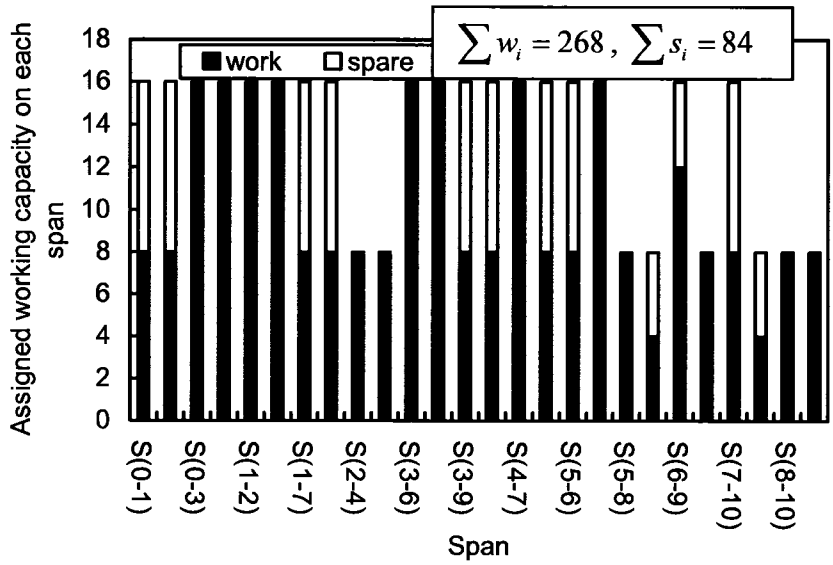
expanded for certain degrees. Specifically, in the SmalleNet network, the expansions are about 10%, 25%, and 25% respectively for the conventional modular design and model B modular deigns with model D and C w/s resplitting. In the COST239 network, the expansions are 47%, 40%, and 40% respectively.



(a) Envelope of Conventional Design with Modularity and Mode D Resplitting (COST239)



(b) Envelope of Model B Modular Design and Mode D Resplitting (COST239)



(c) Envelope of Model B Modular Design and Mode C Resplitting (COST239)

Figure 8.9. Protected Working Capacity Envelopes (PWCE) of the COST239 Modular-Capacity Network with Two Units of Uniform Forecasted Demand on Each Node Pair.

Regarding the structuring effect, Table 8.2 presents the correlation coefficients between the envelope structuring patterns and the working capacity distributions of actual designed PWCEs (i.e., the shapes of envelopes). The envelope working capacity distributions yielded by various modularization-resplitting designs of the above three test networks are displayed in Figures 8.7(a)-(c), 8.8(a)-(c), and 8.9(a)-(c) respectively. Because the conventional design with modularity only modularizes the total baseline capacity, it is reasonable to see that it shows the highest correlation coefficient. Meanwhile, comparing the two coefficients of the model B modular designs with model D and C w/s resplitting, it is not surprising that model D resplitting shows a higher correlation coefficient than that of model C, since the former always splits the total capacity taking the structuring effort into account.

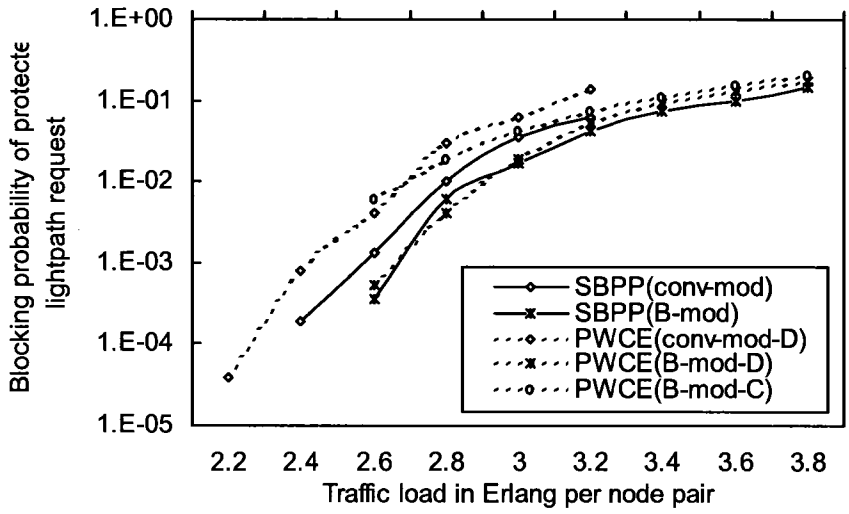
Table 8.2. Correlation Coefficients between the Envelope Structuring Patterns and the Envelope Working Capacity Distributions (i.e., the Shapes of PWCEs) of Conventional Design with Modularity, Model B Volume Maximization Designs with Model D and C Resplitting (Conv = Conventional; mod = Modularization)

Network	Conv-mod-D	B-mod-D	B-mod-C
NSFNET	0.76929	0.307885	0.305929
SmallNet	0.830134	0.392464	0.05792
COST239	0.816281	0.501305	-0.06419

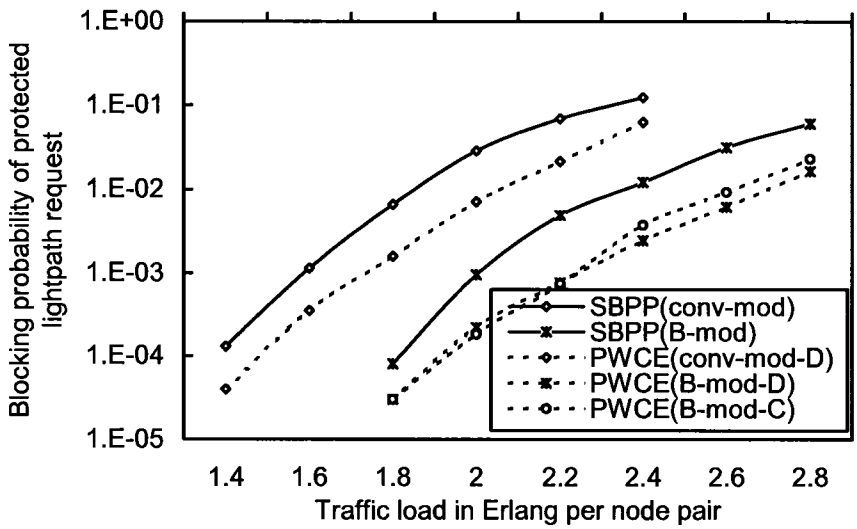
B) Blocking Performance Comparison between SBPP and PWCE

We also evaluated blocking performances for the SBPP and PWCE provisioning approaches under the modular designs. The general finding under the modularization situation is found to be very similar to those found for the previous non-modularization case.

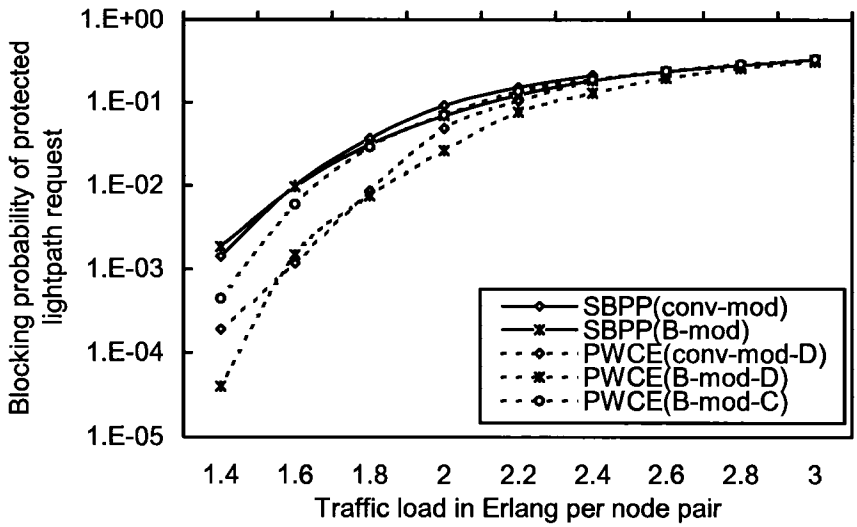
Figure 8.10 plots the blocking probabilities for the three test networks. Similar to the results of the designs without modularity, it is found that in a network with sparse connectivity, SBPP displays better blocking performances than PWCE. However, with the increase of network nodal degree, the blocking performance of PWCE is improved and eventually outperforms SBPP. Now let us discuss the results of the three test networks in detail. The average nodal degree of NSFNET is 3.0, which belongs to a type of network with sparse connectivity. As shown in Figure 8.10(a), SBPP outperforms PWCE (albeit their difference seems slight). In contrast, SmallNet has a relatively higher nodal degree, i.e., 4.4. Its blocking performance of PWCE as shown in Figure 8.10(b) is found to outperform SBPP. Moreover, the performance difference is significant, as we see that the curves of PWCE and SBPP self-group clearly. Finally, COST239 possesses the highest network connectivity, whose average nodal degree is 4.7. Thus, more pronounced performance differences between PWCE and SBPP are observed, as shown in Figure 8.10(c). It is surprising that all the PWCE curves, including the one of conventional modular design, outperform all the curves of SBPP. It should be noted that, for the result of conventional modular design, such a good performance is achieved only when its total network capacity on each span is smaller than that of the model B modular design.



(a) NSFNET



(b) SmallNet



(c) COST239

Figure 8.10. Blocking Performances of PWCE and SBPP Provisioning Approaches Based on the Hop-Based Shortest Path Routing Algorithm and the FF Algorithm in the (a) NSFNET, (b) SmallNet, and (c) COST239 Modular-Capacity Networks. Simulation Rounds=10^5.

To identify how the performance of PWCE is related to the average network nodal degree, we conducted extensive experiments on the derived topologies of a 10-node ring. The results show that at a low nodal degree, PWCE cannot perform as well as SBPP. However, with the increase of network connectivity, PWCE catches up and eventually outpaces SBPP when the average nodal degree grows to about 4.0. This threshold nodal degree is higher than the corresponding one (i.e., 3.2) for the designs without modularity. We attribute this to the additional constraint of bounded total modularized capacity. Without this constraint, under model B PWCE can exploit its full potential, at which the total capacity on each span can be arbitrarily large as long as there is sufficient spare capacity to offer the protection. Consequently, in the designs the spare capacity holds the most efficient protection relationship with the protected working capacity. However, when the total capacity on each span is bounded by a modularized capacity and further split based on some strategies such as models D and C, the protection relationship between working and spare can no longer be maintained as efficiently as in model B (in which no total capacity is bounded on any span). Therefore, under the modularized-resplitting designs, the PWCE method can fall into some sub-optimal spare capacity protection situations. Consequently, the advantage of spare capacity efficiency of PWCE

over SBPP is somewhat weakened, which thereby requires that PWCE have a higher nodal degree to catch up with SBPP in the blocking performance.

The purpose of using both models D and C to resplit the total modular capacity is to identify how the splitting strategies will affect the performance of PWCE. Similar to the previous finding in the non-modular PWCE designs, the results of modular designs indicate that, although the envelopes designed by model C exhibit larger volumes (in bulk), they display higher networking blocking probabilities in all three test networks, again supporting the significance of structuring effort for PWCE designs.

CHAPTER 9
ADAPTIVE PROTECTED WORKING CAPACITY ENVELOPE (APWCE): CONCEPT, DESIGN, OPERATION, AND PERFORMANCE[13]

In this thesis, we have investigated a new dynamic survivable service-provisioning approach PWCE in the context of *p*-cycles, and compared it to the conventional SBPP-based provisioning approach in terms of control overhead and blocking performance. However, the studies were carried out under the assumption that there is an accurate traffic load forecast and the envelope is stationary in both shape and total capacity volume. In practice, the above assumption for the forecasted traffic load may not be valid. There exists traffic load uncertainty and fluctuation in real network environment, which may make the traffic load pattern or distribution mismatch the envelope capacity distribution originally designed for a forecasted traffic load pattern. Consequently, the network performance may be significantly degraded [LeGr02]. A strong distinction should be made between random dynamic, but statistically stationary, traffic and uncertain traffic[14]. Uncertainty is here defined as the non-stationarity of the Erlang traffic loads, which includes aspects of traffic pattern variation and bulk traffic growth. Few researchers have actually addressed the problem of on-line adaptive transport configuration for uncertain, not just random, traffic. Increasingly, network operators face truly uncertain load patterns, not merely random demands. The issue of how to deal with the fluctuation or uncertainty of traffic load in the context of envelope-based service provisioning should be further investigated. This motivates us to develop an extended scheme, termed Adaptive Protected Working Capacity Envelope (APWCE). APWCE integrates self-optimizing and self-reconfiguring capabilities to continually and automatically adapt an operating envelope to match a real non-stationary traffic load pattern, so that the PWCE-based service provisioning can resist the influence of the

[13] This chapter contains some material previously reported in [ShGr04c] and [ShGr05c].

[14] As with telephone traffic, every call can individually arise and complete at random times, and yet the Erlang traffic intensity is precisely known. Often simulations of such randomness are claimed as also addressing uncertainty in planning, but they do not. True uncertainty is reflected by the Erlang intensities of the random processes themselves being unknown.

traffic load fluctuation or uncertainty and maintain a sound network blocking performance any time.

This research focuses on the aspect of traffic load pattern unpredictability, but with the total volume of traffic loads being constant. We can thereby obtain comparative insights about schemes for adaptive response to shifting demand intensities. Otherwise, if the total demand volume drops, then all blocking may go essentially to zero, making comparisons meaningless. If the total demand volume grows significantly, this inevitably requires additional physical capacity placements. Our interest is in the shorter-term time-scale during which spatial patterns of traffic load may change significantly without significant overall volume change, and furthermore in examining how well an adaptive PWCE scheme can accommodate such truly uncertain (not merely random) patterns of demand arrival, working only within the existing installed capacities. We note, however, that when total growth is faced, the information internal to a system that is adaptive to the evolving demand pattern can also greatly inform and guide the capacity augmentation process.

9.1. Concept of Adaptive Protected Working Capacity Envelope (APWCE)

The key idea of adaptive PWCE is to create a feedback control system, in which fairly simple aggregate measurements of utilization on each span are used to drive an optimization problem to optimally reconfigure the envelope to track the evolving traffic load pattern. The measurements of utilization are taken on the time-scale at which the random demand exhibits non-stationary statistical tendencies. For instance, phenomena, such as a focused overload or a traveling busy hour and week-day to holiday variation, are ways in which the traffic load patterns may evolve away from any initially defined busy hour and associated assumption of statistical stationary of the random arrival/departure processes during that interval. For a backbone transport network, these would be measurements taken perhaps on a several-hourly or daily time-scale, not on the time scale of individual connections, but on the scale where the overall statistics evolve. To "adapt the envelope," we recognize that within the physical total capacities on spans, we can revise the boundaries between working and spare capacity to redimension the

PWCE. The hope is to track variation in an uncertain load pattern as it evolves away from the pattern for which the capacities were initially designed. Success will be measured in terms of retaining low blocking of random protected-service demand arrivals, as the overall pattern of offered traffic load evolves.

Essentially, the above working principle of APWCE is analogous to the traditional time-dependent routing (TDR) in the telephony circuit-switching networks [AsCh04]. TDR methods alter the network routing tables at a fixed point in time during the day or week. TDR routing tables that consider the time variation of traffic load in the network are determined on a pre-planned basis and are implemented consistently over a given time period. To allow for APWCE reconfiguration on either a periodic basis or an event basis, the adaptive PWCE that we present here is more advanced than the TDR methods. For instant, an event of severe performance degradation due to the traffic load uncertainty can trigger a new iteration of envelope reconfiguration. In addition, corresponding to the routing tables alteration, APWCE addresses the alteration of a protected working capacity envelope, which involves the reoptimization of its total capacity volume and distribution.

9.1.1. Control System and Operations

To support envelope reconfiguration and dynamic survivable services, the APWCE control system can be made up of a centralized control system, which is adapted based on the previously-mentioned TDR system, incorporated with today's GMPLS-based distributive control plane. The adapted centralized control system is responsible for the envelope performance monitoring and reconfiguration, while the GMPLS control plane is dedicated to actual dynamic service provisioning. Figure 9.1 illustrates the infrastructure of this type of control system, which involves a network central controller and a host of GMPLS-based local switch controllers. The local switch controllers form a GMPLS control plane in which various GMPLS control protocols, ranging from the routing protocols (e.g., OSPF-TE) to signaling protocols (e.g., RSVP-TE or CR-LDP) and link management protocol (LMP), are being run as described in Section 7.3. To support the envelope reconfiguration, each of the GMPLS local controllers should continually monitor network performance in terms of (1) envelope capacity utilization on each incident span, and (2) blocking ratios of the services starting/ending at the node. The

local controllers are also required to respond to queries from the central controller that collects network-wide performance statistics, and to alarm the central controller on adverse performance if any severe performance degradation, such as congestion on a span or high service blocking on a certain node pair, is noticed. The network central controller collects and monitors network-wide performance statistics by sending queries and accepting alarms from the local GMPLS controllers. Based on the obtained performance statistics and the received alarms, the central controller determines whether or not the current operating envelope should be reconfigured. However, the central controller never participates in any activities related to direct service provisioning such as lightpath establishment or teardown; it only focuses on spontaneously reoptimizing and reconfiguring the operating envelope. Because it always takes some time for the network traffic load to experience a significant shifting or variation, the envelope reoptimization and reconfiguration is anticipated to be far less frequent than individual service provisioning. Moreover, although the central controller is plotted separately from the GMPLS control plane as shown in Figure 9.1, this does not imply that it must be physically independent from the local controllers. Actually, it is possible to "collapse" the whole control system onto the GMPLS control plane by designating one or multiple GMPLS local controllers to simultaneously function as the central controller as well. This can largely improve he availability of the control system, since the central controller is usually a risky point for the whole control system.

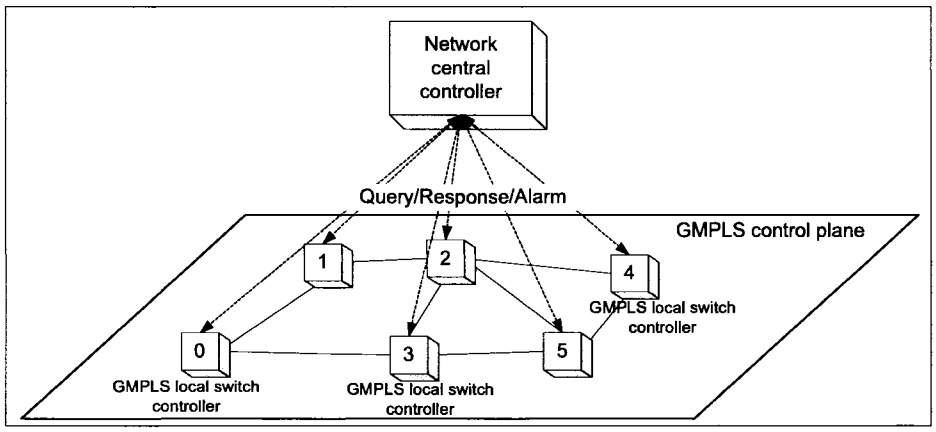


Figure 9.1. Network Control System Architecture Supporting APWCE.

Regarding the practicality of the above control infrastructure, GMPLS is a mature standardized control technique dedicated to provisioning dynamic services within transport networks. Therefore, it would not be a bottleneck for the whole control system. The mechanism of query/response-based centralized network control and periodical reconfiguration has been tested and corroborated in traditional telephony circuiting switching networks [AsCh04]. Viewing far more dynamic nature of the telephone networks, employing a central controller to *spontaneously* reoptimize and reconfigure an operating envelope would not cause a bottleneck to the overall control system either. For the central network controller, an important issue can be the time required for the solution of envelope ILP reoptimization process. Nevertheless, our experiments show that it takes only several minutes or even seconds for PWCE reoptimization under a moderate-scaled network such as NSFNET, which can be anticipated to be much shorter than the time interval between two neighboring APWCE reconfigurations. Moreover, even when the network has such a large size that the time for ILP reoptimization is unacceptably long, it is possible to employ efficient heuristics to redesign a new envelope, which sacrifices the design optimality for a fast APWCE redesign. In addition, for the p-cycle-based survivability technique, limiting the number of eligible cycles can be another effective option to expedite the process of envelope reoptimization.

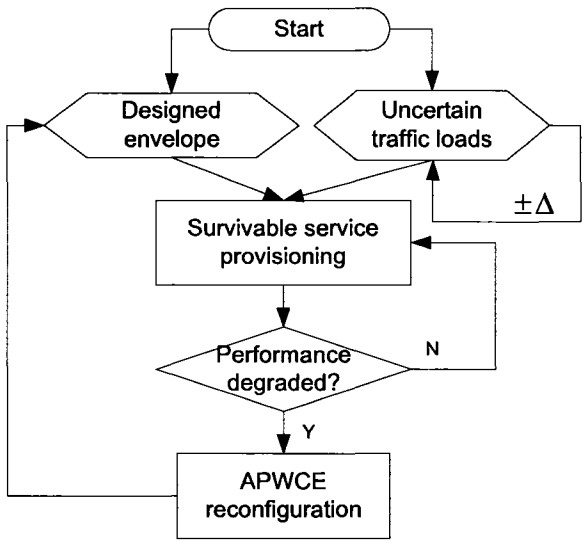


Figure 9.2. Flowchart of APWCE Operation.

Based on the above control architecture, we can further elaborate the APWCE reconfiguration operation as illustrated in Figure 9.2. To begin with an envelope, a

channel volume maximization design problem can be defined to determine the initial capacity and configure it relatively well for low blocking on initial forecast of the long-term average traffic load pattern. This defines initial network conditions and bootstraps the subsequent adaptive tracking process. A reoptimization step occurs whenever warranted, either periodically or based on the onset of congestions or high blocking in certain regions of the envelope. The objective is to restructure the envelope to fit the changed load distribution. The results of the reoptimization are new decisions about the boundary between working and spare channels in the total capacity of each span. This new envelope will be used to provision services for responding to lightpath requests. This reconfiguration operation will be carried out in a recursive manner as illustrated in the figure, such that the envelope can be continually adapted to best fit a real non-stationary traffic load pattern and resist the influence of the traffic load fluctuation to maintain a sound network blocking performance any time.

In shifting to a new configuration, an issue that must be particularly addressed is how to deal with existing services. The existing services should not be interrupted in the course of the reconfiguration. To ensure their continuity, an important constraint that requires the new designed envelope "wrap" all the existing services on each span should be satisfied when designing a new APWCE. We will give a detailed interpretation on this constraint in the subsequent modeling section.

9.1.2. Strategies for Triggering Reoptimization

We consider three possible strategies to trigger APWCE reconfigurations. Nonetheless, nothing prevents other strategies or a multi-criteria reoptimization policy from triggering reoptimization.

A) Strategy I: Periodic

The *periodic* strategy is the simplest to assume that an APWCE reconfiguration is initiated whenever a certain time period has passed. Obviously, under this strategy, the length of reconfiguration period involves a tradeoff between network performance and overall network control burden. Good tracking at low blocking can be expected if the reconfiguration is carried out at a period that is known to be two or more times as rapid as significant changes in the traffic load pattern can evolve. Therefore, it is worthwhile to

find a reasonable reconfiguration period that can benefit from PWCE reconfiguration, but does not bring too much burden to the control system.

B) Strategy II: Link Utilization

Under this strategy, the network controller monitors a sliding-window average measure of the utilization of channels in the envelope on each span. According to this metric, the control system determines whether or not a PWCE reconfiguration is needed. If the average utilization on one or more spans exceeds a certain threshold, a reoptimization will be triggered, aiming to relieve the congestion. The average link utilization is defined as follows:

$$u_i = \frac{\sum_{t \in 1 \ldots T} w_i(t) / w_i^0}{T} \tag{9-1}$$

An alternate that also can be considered is

$$u_i = \frac{\sum_{j \in 1 \ldots N} w_i(j) / w_i^0}{N} \tag{9-2}$$

In equation (9-1), T is the total number of time samples, $w_i(t)$ denotes the number of working channels occupied at the t^{th} sampling, and w_i^0 is the current protected envelope capacity on the span. Alternatively, we may generate a utilization measure as in equation (9-2), which is driven only by arrivals or departures that affect a given span. In this case N is the total number of arrival events during the measurement period, and $w_i(j)$ is the number of working channels occupied at the j^{th} arrival event on span i. Equation (9-2) represents utilization as the average ratio of used capacity units to the current *envelope* capacity on the span. Statistically, averaging on a time-sample basis and an arrival-event basis are equivalent. Therefore, equation (9-1) or (9-2) could each be expressed in either form for the samples averaged. We have employed equation (9-2) to compute the average utilization in our studies.

C) Strategy III: Blocking Performance

The *blocking performance* strategy can be based either on node-pair service blocking performance or network-wide service blocking performance. The service *blocking probability* is defined as the ratio of the total blocked requests to the total number of requests over an averaging period. Under the node-pair subcase, each local controller monitors and records the blocked service requests that start or end at the node. If the blocking performance of any node pair exceeds a certain threshold level within a continuous monitoring period, an alarm will be sent to the network central controller and an APWCE reconfiguration will be triggered. While under the network-wide subcase, the network central controller periodically collects service blocking statistics from the network local controllers and then computes the network-wide blocking probability. When a certain predefined threshold is reached, the APWCE reconfiguration will be triggered. The node-pair blocking criterion is thus a more directly end-user-oriented strategy, which may be helpful to guarantee Quality of Service (QoS) for each customer in terms of blocking probability or availability. In contrast, the network-wide blocking criterion can improve the overall network blocking performance, but may not assure the level of blocking performance for each node pair. However, in this study, without losing generality, we have employed the network-wide blocking strategy to trigger APWCE reconfiguration.

9.1.3. Envelope Template for PWCE Reconfiguration

When a new round of APWCE reconfiguration is initiated, a capacity resplitting model should be applied to design a new envelope. In order to ensure that the new designed envelope matches the current network capacity utilization, a design pattern or template should be provided. We assume that the APWCE reconfiguration process is Markovian. That is, new envelope design and reconfiguration of the current step is only related to the current operating envelope, but not relevant to previous envelopes in history. This assumption brings us much simplicity when determining the PWCE design template. If the N^{th} envelope is the current operating envelope, to design a new envelope, we use a term called *expected envelope capacity* to function as the design template for the $(N+1)^{th}$ envelope. The term e*xpected envelope capacity* is defined as the product of the span average link utilization and the envelope capacity of the current operating (i.e., the

N^{th}) envelope. This product is expected to have an excellent match with the current traffic load pattern to reflect the minimum expected capacity on each span to serve the current traffic load pattern. Thus, it is an efficient template for the $(N+1)^{th}$ envelope design. With this template, it is straightforward to foresee that a span with a larger amount of *expected envelope capacity* will be assigned more envelope capacity.

9.1.4. Total Span Capacity for Reconfiguration

Various assumptions can be made for the total deployed capacity on each span. In a network without modularity, the total capacity can be simply the sum of working capacity and spare capacity obtained based on the volume-maximization design[15], i.e., the exact sum of w_i^0 and s_i. However, it is also possible that some redundant capacity is deployed on the span. A common example is a network with modularized capacity, where the deployed capacity is modularized, and usually over-provisioned compared to the capacity really required. For the over-provisioned capacities, one option is to use the resplitting models to partition them into working and spare units as in Chapter 8, in order to further expand envelope volumes. Alternatively, we may leave the over-provisioned part unattended or use it to carry best-effort traffic, SBPP traffic, etc. If the system is operated in the latter fashion, more opportunities can be exploited for the APWCE reconfiguration. The over-provisioned capacity can be used to compensate for the capacity shortage on some span if a new designed envelope requires more capacity than the old one. In addition to modularized capacity, we may study a network with uniform capacity. That is, there are an equal number of capacity units (say 16 units) deployed on each span. As mentioned, although such a network may not be as practical as the network with non-uniform capacity, it is worthy of investigation owing to the wide interests of researchers. Moreover, this study can help us to verify the results obtained for the network with modular capacity.

[15] However, such a span capacity will bring few benefits to implement APWCE reconfiguration. This holds true because the capacity has been deployed so stringently to *perfectly* accommodate the current envelope that little room is left to allow for efficient new envelope redesigns, which have close envelope volumes, but absolutely different capacity distributions from the current one.

9.2. ADAPTIVE PWCE DESIGNS: PROCEDURE AND MODELS

Before beginning the APWCE-based real operation, the network first needs to have an initial envelope, which is designed to optimally fit the long-term traffic load forecast. Therefore, we need optimal design models to specifically design such a starting-point envelope. We employ the initial designed envelope to provision dynamic survivable services. The uncertainty of real traffic load may affect the network performance to be lower than a certain threshold level, which triggers a new iteration of PWCE reconfiguration. To reoptimize the envelope based on the current network traffic load, a reoptimization model is required. Specifically, we use the current network performance or status (e.g., span link capacity utilization) to guide the new envelope design. We employ the term *expected envelope capacity* (as described in Section 9.1.3), which is defined as the product of the envelope capacity $w_k^0(N)$ and the average link capacity utilization $u_k(N)$ [see equations (9-1) and (9-2)] of the current (i.e., N^{th}) envelope, to function as a weight factor to redesign a new [i.e., the $(N+1)^{th}$] envelope. It is reasonable to apply such a factor to shape the new envelope capacity distribution, since it precisely reflects the expected envelope capacity that is really required by the current network traffic load pattern[16].

The overall modeling procedure for the PWCE reconfiguration is illustrated in Figure 9.3 and is somewhat evocative of training the weights in neural network for minimum error on some recognition task. However, here it is the pattern of spare capacity allocation on the network graph that in effect is being trained to produce a PWCE configuration that best fits the current traffic load distribution. Interestingly, the mapping of changes in spare capacity vector to working envelope capacity vector is through the restorability operator acting through the particular network graph. In Figure 9.3, the parameter of *expected envelope capacity* on each span of the N^{th} round envelope $u_i(N) \cdot w_i^0(N)$ is the primary factor to assess the envelope capacity of the $(N+1)^{th}$ iteration, but every change on one span implies possible corresponding changes on others,

[16] We must avoid a common (intuitive) misunderstanding that a span with a higher capacity utilization $u_k(N)$ would always be assigned with more envelope capacity in the new design. This is true only when the N^{th} envelope has the same envelope capacity on each span; otherwise, the weighting factor should also consider the current N^{th} envelope capacity.

all coupled through the network structure and the logic of how a distribution $\langle s_i \rangle$ determines a corresponding $\langle w_i^0 \rangle$. This is represented by the dotted lines and weighted by a general function $f(u_k(N))$.

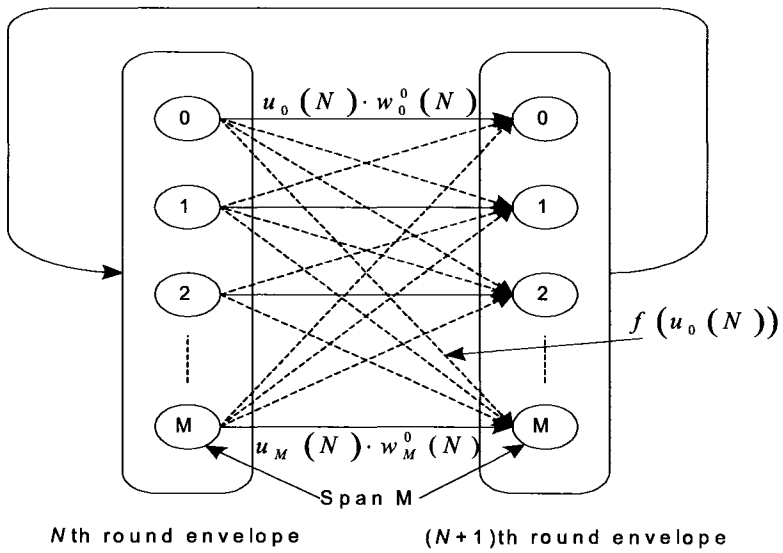


Figure 9.3. Model of APWCE Reconfiguration.

9.2.1. APWCE Reoptimization in Networks with Non-Uniform Span Capacity

In a network allowing for different installed capacity on each span, the APWCE optimization process begins with an initial PWCE design. Given a forecasted demand pattern or matrix, we employ the conventional *p*-cycle design model [GrSt98a] to design a *p*-cycle network. Then we assume the obtained protection capacity as the *reserve network* to run a volume-maximization process. Model B in Chapter 7, which attempts to identify the maximum volume of working capacity, but meanwhile retains the shape of the envelope to be akin to the initial working capacity distribution based on the forecasted pattern, can be used for this purpose. After the volume-maximization, the step of capacity modularization as described in Chapter 8 can be followed up. Given a set of modules and their individual costs, the modularization process aims to minimize the total network cost under the constraint that all the volume-maximization capacities obtained in the previous step should be fully accommodated. As indicated, the modularization process can often bring about over-provisioned capacities on some spans in addition to the pre-modularized. Essentially, it is these over-provisioned capacities that provide extra space

for PWCE to efficiently carry out adaptive reconfigurations. These over-provisioned capacities allow the new envelope design to assign more capacity on some span than the one before reconfiguration. As long as the resultant total capacity never exceeds the total deployed modular capacity on the span, the envelope capacity is allowed to be set arbitrarily. As such, in the reconfiguration model, besides the constraints to fully protect working capacities, two other constraints, of which one is to ensure the total capacity on each span does not exceed the total modularized capacity, and the other to ensure the total capacity of each envelope design is constant, should be satisfied. Note that the second constraint is added for the purpose of fairness when comparing the performances among different envelope reconfiguration designs.

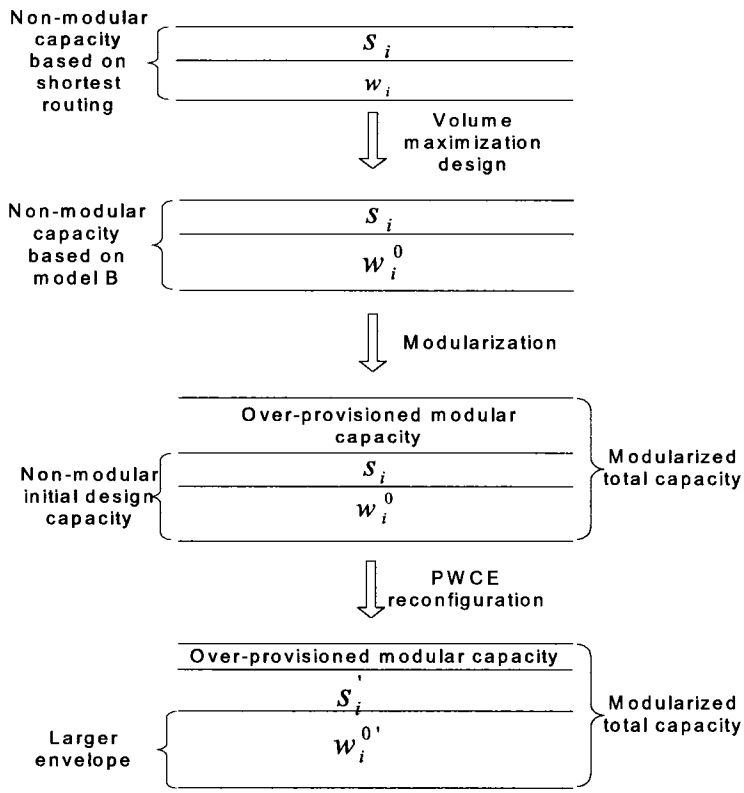


Figure 9.4. Procedure of APWCE Reoptimization, Which in Sequence Involves the Steps of Initial PWCE Design, Capacity Modularization, and Modularized Capacity Resplitting.

Figure 9.4 illustrates the module-based PWCE design and reconfiguration procedure, where a capacity modularization process is carried out to modularize the capacity with some portion over-provisioned, and based on the modularized capacity, the capacity

resplitting step is undertaken to redimension the envelope boundary from w_i^0 to $w_i^{0'}$ and correspondingly the protection capacity from s_i to s_i'.

To summarize, in the above design procedure, following design models are involved: (1) conventional p-cycle design, (2) envelope capacity volume-maximization design (see model B in Chapter 7), (3) capacity modularization (see the modularization model in Chapter 8), and (4) capacity resplitting models. For the capacity resplitting model, given the modularized capacity, it is somewhat different from the generic resplitting model, i.e., model D, in Chapter 7. We present the new resplitting model below.

In addition to the sets, parameters, and variables presented earlier, new terms that are specific for PWCE reconfiguration are listed as follows:

- T_k^m represents the total number of modularized capacity units on span k. (parameter)
- u_k represents the average link utilization of span k in an envelope. $u_k(N)$ denotes the average link utilization of span k in the N^{th} operating envelope. We have employed equation (9-2) to compute it. (parameter)
- w_k represents the instantaneous working capacity units on span k when a new iteration of APWCE reoptimization is triggered. Specifically, $w_k(N)$ denotes the instantaneous working capacity units on span k when designing the $(N+1)^{\text{th}}$ envelope. (parameter)
- w_k^0 is the number of protected working channels (i.e., envelope capacity) on span k. $w_k^0(N)$ denotes the number of such channels on span k in the N^{th} operating envelope. Note that when carrying out the $(N+1)^{\text{th}}$ reoptimization, $w_k^0(N)$ becomes a parameter while a new variable $w_k^0(N+1)$ is brought in, which is the new envelope capacity on span k for the next period. (variable)
- s_k is a general term representing the number of assigned spare capacity unit on span k. $s_k(N)$ denotes the number of spare channels on span k in the N^{th} envelope reoptimization. (variable)

- n_j is the number of unit-capacity (i.e., single-channel) copies of cycle j preconfigured to offer span failure protection. $n_j(N)$ is n_j of the N^{th} envelope reoptimization. (variable)

The N^{th} envelope reoptimization generates the $(N+1)^{th}$ round envelope, whose model is presented as follows:

$$\textbf{Re-opt: maximize}\left\{\lambda+\alpha\cdot\sum_{k\in S}u_k(N)\cdot w_k^0(N+1)\right\} \tag{9-3}$$

Constraints:

$$s_j^0(N+1)\geq\sum_{i\in P}P_j^i\cdot n_i(N+1)\quad\forall j\in S \tag{9-4}$$

$$w_k^0(N+1)\leq\sum_{i\in P}X_k^i\cdot n_i(N+1)\quad\forall k\in S \tag{9-5}$$

$$w_k^0(N+1)\geq\lambda\cdot u_k(N)\cdot w_k^0(N)\quad\forall k\in S \tag{9-6}$$

$$w_k^0(N+1)+s_k^0(N+1)\leq T_k^m\quad\forall k\in S \tag{9-7}$$

$$\sum_{k\in S}\left(w_k^0(N+1)+s_k^0(N+1)\right)\leq\sum_{k\in S}\left(w_k^0(0)+s_k^0(0)\right) \tag{9-8}$$

$$w_k^0(N+1)\geq w_k(N)\quad\forall k\in S \tag{9-9}$$

The objective is to maximize the envelope volume and balance this against retention of a "shape" that is scaled similar to the actual expected working capacity profile of the previous iteration. When applied iteratively for APWCE, the shaping target will be specifically based on the actual average working channel counts on spans (i.e., *expected envelope capacity*) in the prior iteration:

$$u_k(N)\cdot w_k^0(N)=\frac{1}{A}\cdot\sum_{a\in 1\ldots A}w_k(N,a) \tag{9-10}$$

The idea is to shape the envelope to be similar to the recent experience of what is actually used (as being at least indicative of what is probably needed), but beyond that simply maximizing the PWCE channel volume. The shape factor λ works through constraint (9-6) to put utility on similarity between the reoptimized PWCE and actual

average working channel counts in the prior period. In constraint (9-6), N denotes that it is the utilization and envelope values before reoptimization that are used in the current reoptimization as a shaping target. In the test results, however, the tradeoff factor α is set to be a small value, which guarantees maximization of the shape factor λ as the primary objective. As a result we tend to see λ maximized until the spare capacity cannot guarantee restorability of the envelope if the factor is further increased. The secondary objective is to maximize the sheer envelope volume after it cannot be further enlarged under the exact shape of the envelope template. Constraint (9-7) ensures the total assigned capacities respect the total modular capacity on the span. And constraint (9-8) guarantees the total capacity of a new designed PWCE network, which is the sum of the envelope capacity and the spare capacity used to protect the envelope, should never be more than that of the initial design, i.e., $\sum_{k \in S} \left(w_k^0(0) + s_k^0(0) \right)$. We add such a constraint for the fairness of comparison between different reoptimization designs, although in real-world operations it is possible to release such a constraint, especially under the situation that the volume of total network traffic load has some increment. Finally, constraint (9-9) ensures that a new designed envelope can always "wrap" the instantaneous operating working capacity on each span, so that when reconfiguring the envelope, all the existing working services can retain their operation continuities. This constraint is of importance to the real network operation since customers appreciate continuous services. Moreover, it simplifies network control and operation—there is no need to reroute any existing services.

9.2.2. APWCE Reoptimization in Networks with Uniform Span Capacity

Networks with uniform total capacity on all spans present a simplified special case for APWCE reconfiguration. When each span has the same total number of deployed channels, an intuitive approach to designing an initial PWCE is simply to split between working and protection in a way that maximizes the total PWCE volume. Based on this, the initial PWCE design model is presented as follows:

Objective: maximize $\sum_{k \in S} w_k^0(0)$ (9-11)

Subject to constraints (9-4), (9-5), and (9-7), wherein N+1 is replaced by zero and T_k^m is set to be the uniform total capacity on each span.

This initial PWCE design functions as a starting point of operation. When the traffic load pattern shifts, the steps to reoptimize PWCE are the same as those described in the previous subsection, except that in the reoptimization, constraint (9-8) is not needed, since the total capacity on each span is always a uniform value.

9.3. Test Cases and Methodology

9.3.1. Test Networks, Initial PWCE Design, and Simulation

The NSFNET and SmallNet networks have been studied as our test networks. The designs of initial PWCE were carried out based on the two networks each under conditions of non-uniform (differential) span capacity and uniform span capacity. For the test cases with differential span capacities, we assumed that in NSFNET and SmallNet, each node pair had uniformly *three* units of demand and routed these via shortest paths to generate an initial set of working capacity requirements on each span, $\langle w_i \rangle$. As before, we used the demand-splitting shortest path routing algorithm, which distributes the total flow between node pairs as equally as possible over all paths of the shortest hop-length subject to retaining the integrity of each demand unit. We then employed a conventional p-cycle design model [GrSt98a] to find a set of initial p-cycles and spare capacities $\langle s_i(0) \rangle$. The PWCE volume-maximization model (i.e., model B) was then applied based on $\langle s_i(0) \rangle$ to produce a volume-maximized envelope design as the initial $\langle w_i^0(0) \rangle$ envelope. The process of total span capacity modularization then followed. Again, without losing generality, we assumed modular capacities of 4, 8, 16, 32, and 64-channels following a 4-to-2 economy of scale progression, according to the method in [DoGr00]. For test cases with uniform capacity, we assumed that on both test networks, there were a total of 16 channels equipped on each span. With this total capacity, we employed the simple volume maximization objective (9-11) to design the initial PWCE envelopes. We used AMPL/CPLEX 7.1 on an Ultrasparc Sun Server at 450MHz with 4GB of RAM to solve all our optimization problems.

For the simulation of dynamic service provisioning, as before the hop-based shorting path routing algorithm was applied to search routes based on only available channels of the current envelope for APWCE. It is possible that there exists more than one identical hop-length shortest route between a node pair. If so, we will find all of them and then randomly select one of them. For SBPP, the FF routing algorithm was used. Finally, except where noted below, all blocking probabilities were based on 10^5 arrivals on the network as a whole.

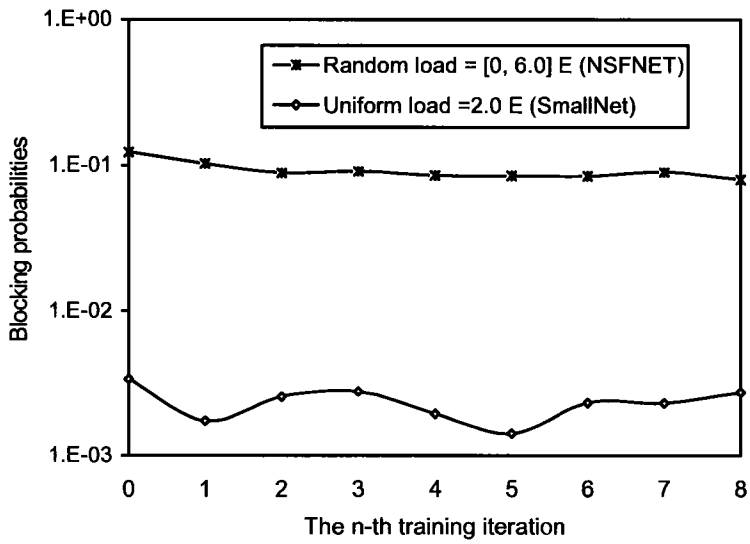
9.3.2. Traffic Load Pattern Generation

To both demonstrate and evaluate the adaptive PWCE process, we need to generate sequences of *non*-stationary traffic load patterns. In other words, we need to model random arrival/departure processes over all node pairs in which the Erlang traffic loads of the individual node-pairs themselves evolve with time. Three types of traffic load patterns have been generated. One is a uniform load distribution where each node pair sees the same Erlang intensity of random arrivals/departures. The second is based on random assignment of the Erlang load for each node pair. For experimental control, the sum of all the traffic loads on each node pair is kept constant through scaling. The third type of load pattern is generated by a process to evolve the traffic load in a way that does not instantly de-correlate over space or time, but assume that it changes under a simple randomly-modulated intensity model. Given an initial traffic load pattern, a relatively small random step increase or decrease in load is taken for each node pair to generate a new load pattern. The range of step change is limited, and negative step changes set the traffic loads on node pairs to be zero if the resultant load on some node pair would become negative. We term the traffic load pattern generated based on the above process *evolving random pattern*. Although we cannot guarantee that traffic load pattern changes in real networks exactly follow the way we have described, the above evolving process can generate a sequence of traffic load patterns that effectively reflect the uncertainty of network traffic loads. The topic of how the real traffic loads change is beyond the scope of this study.

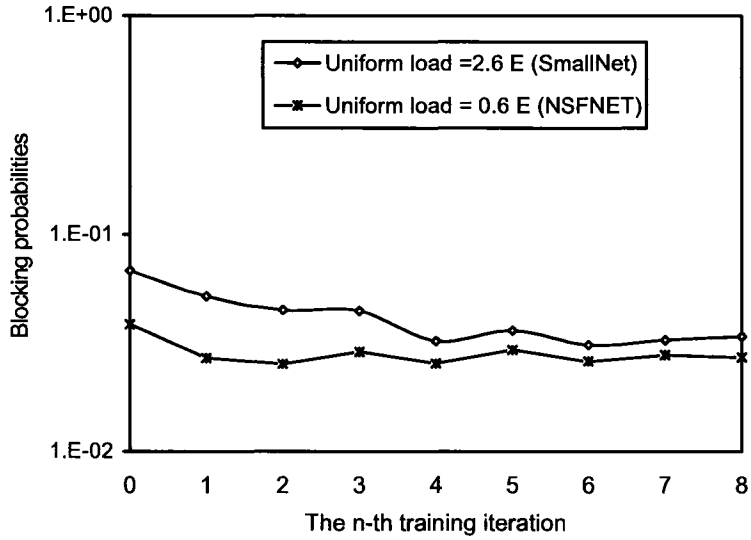
9.4. Results and Discussion

9.4.1. Initial Experiments: APWCE Convergence on a Stationary Random Demand Pattern

This set of initial experiments is used to test the stability and convergence of the proposed reconfiguration process under a stationary demand pattern. The purpose of APWCE reconfiguration is to maintain the network blocking performance always within a certain acceptable level by reconfiguring the envelope when the blocking performance is severely degraded. This implies that each reconfiguration process is expected to improve or at least sustain the blocking performance at a similar level. To ensure this, the reconfiguration process must possess an important characteristic—convergence. Convergence means that the reconfiguration process tends to improve the blocking performance and eventually converges at a certain optimal point or within a certain optimal region, where the blocking probability is the lowest. In other words, it is impossible that the reconfiguration process would sometimes improve the performance, while at other times it would severely degrade the performance. Essentially, convergence is the theoretical basis to justify why the reconfiguration is effective to the performance improvement.



(a) Network with Differential Span Capacity



(b) Network with Uniform Span Capacity

Figure 9.5. Demonstration of APWCE Convergence and Stability ("Self-Training") on Stationary Traffic Load Patterns.

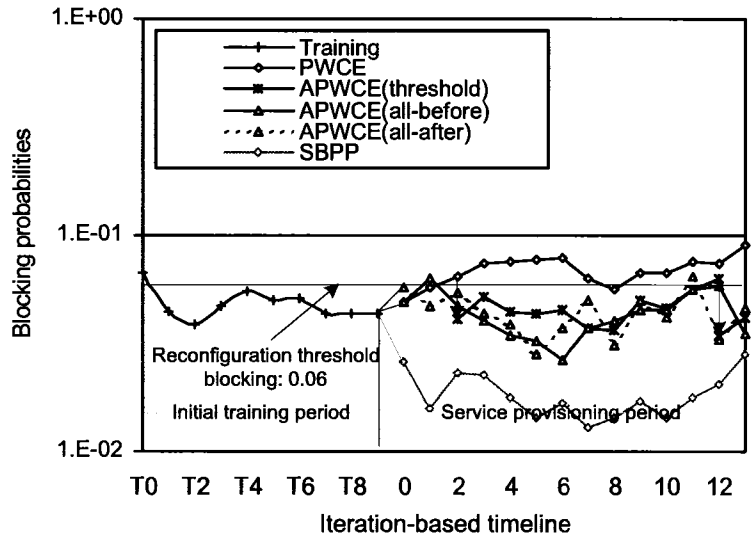
The initial experiments were dedicated to determining whether the proposed APWCE reconfiguration method can guarantee such a convergence. We iterated the APWCE reconfiguration model given a stationary traffic load pattern to in effect let it reconfigure (train) itself on the load pattern. Figure 9.5 provides the results of permitting APWCE iteration for initial envelope configurations for the NSFNET and SmallNet networks. Figure 9.5(a) shows the results for the networks with differential span capacities, whereas Figure 9.5(b) plots the results for the networks with uniform span capacity—there are a total of 16 deployed channels on each span. Specifically for the networks with differential span capacities, we served a uniform pattern for SmallNet, with each node pair having 2.0 Erlangs, and a stationary random pattern for NSFNET generated by assigning a random intensity between [0, 6.0] to each node pair and scaling all loads for a total of 273 Erlangs (on 91 node pairs, this corresponds to 3 Erlangs on average per node pair). For the networks with uniform span capacity, two uniform traffic load patterns were assumed respectively for the two test networks with NSFNET having 0.6 Erlangs per node pair and SmallNet having 2.6 Erlangs per node pair. The results indicate that the convergence really exists for all four cases. Through the convergence process, we observe that overall network blocking performance is at first improved, after which it becomes stable within a certain range. Within that range, there can be some oscillation in blocking probability, but

it never goes too far to escape from the stable locations. This demonstrates at least experimentally on these test cases that the APWCE reconfiguration model is a stable and convergent system on a stationary traffic load pattern. This thereby provides a theoretical basis for arguing that the proposed reconfiguration process can always be effective to the performance improvement. Let us now go on to challenge it with non-stationary load evolutions, and see how it tracks its configuration in response and how its blocking performance compares to SBPP.

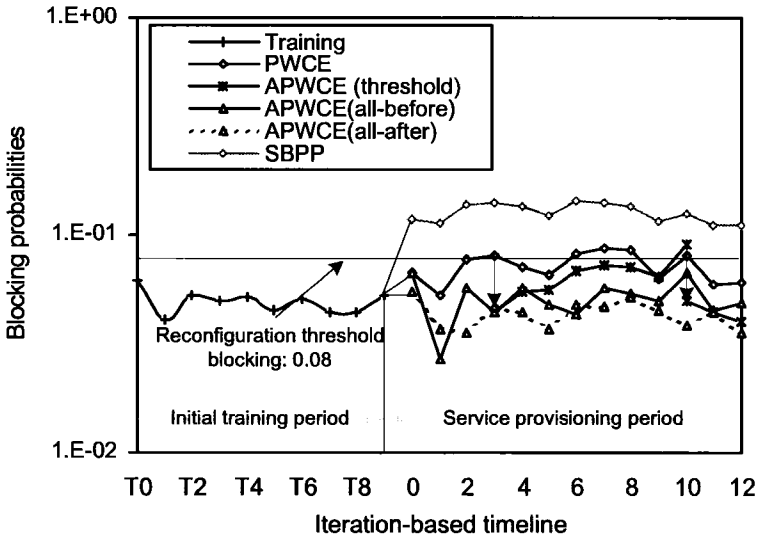
9.4.2. Performance under Evolving Traffic Load Patterns

A) Network with Non-Uniform Span Capacity

In the next experiments, we challenged the APWCE reconfiguration process with evolving (non-stationary) traffic load patterns generated by the above evolving method in the networks with differential span capacities. We first generated an initial traffic load pattern for each of the networks. For SmallNet this was a random (stationary) load pattern, from a random number generator in [0, 4.0]. The resultant loads were all scaled for a total load of 108 Erlangs (45 node pairs at 2.4 Erlangs average on each node pair). NSFNET was given a uniform load pattern with each node pair having 2.8 Erlangs. Starting from these two initial load patterns, we proceeded to evolve the load patterns with the evolution method. On SmallNet each step change was within the range of [-2.16, 2.16], which permits up to 90% change in traffic intensity on a node-pair relative to the 2.4 Erlang average value. Most step changes were smaller than this of course, but intuitively this amounts to representing a fairly volatile, rapidly evolving, traffic load environment. Scaling was also applied at each step to keep the total load at 108 Erlangs. The purpose of normalizing the total load is to separate blocking that arises from changes in the *pattern* of the traffic load (it is this pattern to which we must adapt) from blocking increases or decreases that could result not from good or bad adaptation but purely from total load changes. We did not consider the situation where the volume of total traffic loads varied as well. However, that can be a future research topic. For NSFNET each step change on each node pair was in the range of [-1.4, 1.4] corresponding to up to 50% change per adaptation step with 2.8 Erlang as average load. Total volume was similarly scaled to stay constant at the value of the initial load pattern.



(a) NSFNET



(b) SmallNet

Figure 9.6. Blocking Performance Comparison of Various Survivable Service Provisioning Schemes in a Network with Non-Uniform Span Capacity.

Before starting to experience the evolution of the traffic load patterns, we allowed ten iterations on the initial traffic load patterns to ensure that at least initially the PWCE dimensions were perfectly matched to the initial load conditions. In Figure 9.6, these are the time steps marked "T." For comparison, SBPP is run in parallel through the same ten periods of random stationary arrivals and departures. Then for the 11^{th} and subsequent time-steps, evolving-modulation of the load patterns begins. This is the real test of importance and relevance for a scheme of this type: Can it track the non-stationary random traffic load pattern, retaining low blocking no matter how the pattern evolves (as

long as the total load is feasible within the physically present capacity)? In both cases during the training period we note quick convergence to a steady or steady-average blocking level somewhat below 10% network-wide.

Following the initial convergence phase, Figure 9.6 shows five different blocking curves. The curve "PWCE" shows the corresponding static PWCE using the envelope arising from convergence on the initial condition, with no further changes. This serves as a check that adaptation does in fact matter. This static PWCE is a good configuration for the first load pattern. However, without ongoing reconfiguration, blocking rises. The curve called "threshold" APWCE corresponds to using the overall network blocking probability as the reconfiguration trigger. In NSFNET we experimentally set the threshold blocking to be 6% and in SmallNet, 8%. We see that whenever the blocking probabilities of PWCE exceed the predefined thresholds, the reconfiguration processes are triggered to generate new envelopes, which then greatly improve the blocking compared to the static PWCE. In these trials with both NSFNET and SmallNet, this triggering strategy actually results in only two reconfigurations respectively for a total of 13 iterations. The reconfigurations are marked with the pointing-down arrows. For each iteration, a total of 10^5 arrival lightpath requests were simulated for the blocking performance evaluation. Consequently, 13 iterations correspond to a total of more than 10^6 arrival events. If we also consider the associated departure events during the simulation, then the total number of arrival and departure events would be more than two million. Based on these data, we can see how spontaneous the APWCE reconfigurations are—for more than two million arrival and departure events, the APWCE approach needs only two reconfigurations. This thereby confirms our previous expectation that the APWCE reconfigurations will be spontaneous and not cause a critical burden to the network control system. In addition, it is quite clear that after these reconfigurations, the blocking is dragged back below the threshold. Therefore, from these observations and the principles about the blocking-triggered scheme, the characteristic behavior we could expect would be oscillation around the threshold blocking level.

The curves called "APWCE (all-before)" and "APWCE (all-after)" correspond to continual re-optimization at every time-step of the traffic load evolution, which can be

regarded as an APWCE approach based on the periodic reconfiguration strategy. "Before" refers to the realistic case where the reconfigured envelope is based on utilization measures from the prior period, but used to serve the actual traffic in the current operating period. "After" is a strictly artificial case, included for research purposes, that shows the performance if the utilizations from the current period are first precisely operated, then the corresponding envelop used to serve during that interval. Although strictly artificial (not feasible in practice) it can be thought of as showing the performance to be expected in the limit of slowly-changing load patterns. As might be expected, APWCE (all) curves do show the best overall performance of the PWCE schemes. In addition, although APWCE (all) curves have more frequent APWCE reconfigurations, compared to the more than two million total arrival and departure events, the 13 reconfigurations are still spontaneous and thus trivial to the network control system.

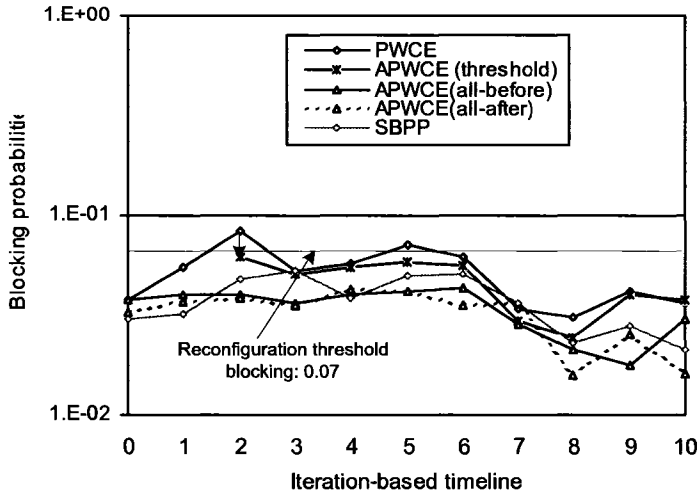
The last curve is for SBPP using the same total network capacity as the PWCE test cases (but PWCE uses the envelope capacity to provision survivable services). In NSFNET, SBPP achieves better blocking performance than PWCE and APWCE schemes, although the absolute differences in blocking are trivial. Conversely, in SmallNet, which is more highly connected, the blocking of SBPP is worse than any of the PWCE schemes. This is consistent with results in our prior study of comparing a fixed PWCE and SBPP under random but non-evolving traffic loads, where we found that with average nodal degree over 3.0 to 3.2 the PWCE scheme can achieved a better blocking performance than SBPP. Here NSFNET has an average nodal degree of 3.0, while SmallNet's node degree is more than 4.0. That is why we obtain the above converse results. This general tendency is thought to be attributable to the use of *p*-cycles as the underlying APWCE protection mechanism. It is already well-known that span-oriented survivability technologies in general rise in redundancy faster than path-oriented schemes as graph connectivity decreases. Therefore, in these tests it is not so much that SBPP or APWCE are differing in "adaptiveness," but rather that APWCE's protection technique is requiring relatively more of the protection capacity at lower degree, thereby leaving fewer (within equal total capacities) channels to be incorporated in the protected working envelope. Finally, we comment that our interest in APWCE as a practical option for real-

world networks does not hinge critically on whether APWCE always has lower blocking than SBPP. As long as the blocking is essentially similar between schemes, then the case for APWCE is made in all the other operational simplifications that it provides. This will be especially true in networks that in fact are not typically experiencing significant blocking at all, but are facing rapid arrival/departure rates. It is in these circumstances that APWCE provides the simplest end-user provisioning paradigm and yet also has the adaptive nature needed to avoid evolution towards a possible high-blocking configuration.

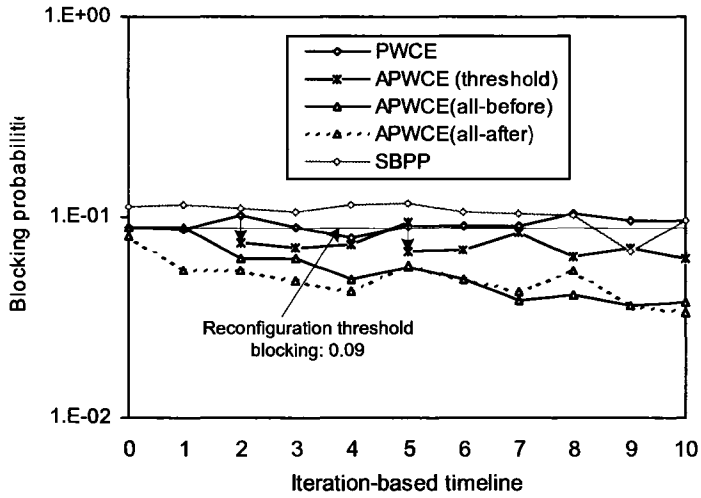
B) Network with Uniform Span Capacity

The purpose of the test under uniform span capacity is to make a further evaluation on whether or not the proposed reconfiguration process can generally bring performance improvement for other network capacity scenarios such as a uniform span capacity, which is often studied by researchers. Similar observations were found under the uniform capacity case to those of the previous non-uniform capacity case.

In this set of test cases, the two networks had exactly 16 channels in total on each span. For the PWCE cases we first used pure volume-maximization [objective (9-11)] to generate initial working envelopes. We then repeated the above experimental trials, but with revised loads to define a meaningful set of experiments given the specific capacity. However, for these tests we let the APWCE process proceed without the benefit of any initial convergence period. For NSFNET, we started from a uniform traffic load pattern with each node pair having 0.6 Erlang load, followed by evolving step changes in [-0.6, 0.6] which is up to 100% of the initial 0.6 Erlang uniform load. We again scaled for total traffic load to be held constant at that of the initial load pattern (54.6 Erlangs). For SmallNet the initial uniform load pattern has 2.6 Erlang on each node pair and the step change is uniform random in [-1.3, 1.3] with scaling to 117 Erlangs in total. To save our simulation time, in the following results we used only 10^4 arrivals per simulation period.



(a) NSFNET



(b) SmallNet

Figure 9.7. Blocking Performance Comparison of Various Survivable Service Provisioning Schemes in a Network with Uniform Span Capacity.

Figure 9.7 shows the experimental results. We find that they are very similar to those we obtained in the previous section, although now it can be argued that the comparison against SBPP is even more rigorous because the capacity design of the initial network was totally arbitrary, unrelated to either scheme (whereas above, recall that the initial total capacities had their origins in a modularized p-cycle network design). In the results of SmallNet, almost all the PWCE-related curves outperform the SBPP curve to demonstrate better blocking, which can be attributed to the high connectivity of SmallNet.

Nonetheless, in NSFNET, any SBPP and APWCE blocking differences are basically "awash"—there is no absolute claim to always achieving the lowest blocking for either scheme, although APWCE (all) curves are most often strictly the lowest in blocking. Absolute differences are however still only several percent or so, and thus not of great significance. Perhaps of greater relevance would be an interpretation that the average nodal degree of NSFNET (=3.0) lies in the transiting region where PWCE is outperforming SBPP in terms of blocking probability.

9.5. SUMMARY

Based on the concept of fixed PWCE, we extended to implement and evaluate the concept of Adaptive PWCE (APWCE). The purpose of this is to enable the PWCE framework capable of more practically supporting real-life dynamic survivable service provisioning. To thoroughly understand APWCE, we analyzed a wide range of critical aspects related to the scheme and proposed various possible strategies for each of the aspects. It is found that compared to SBPP, APWCE continues keeping the feature of simplicity in control and operation. Moreover, based on extensive experimental tests, we also proved the stability of the developed PWCE self-reconfiguration process, and meanwhile proposed various traffic load generation ways to evaluate the performance of APWCE in comparison with fixed PWCE and SBPP. It is found that APWCE demonstrates important advantages over the other schemes in resisting the blocking performance degradation due to the traffic load uncertainty. Without the PWCE reconfiguration, the traffic load uncertainty can significantly degrade the blocking performances of a fixed PWCE. In contrast, with the support of PWCE reconfiguration, the degraded blocking performance can always be dragged back below a certain acceptable level through a simple but infrequent reconfiguration process. The experiments were also conducted for a network with uniform capacity, and the same conclusion was reached that APWCE performs best in resisting the blocking performance degradation due to the traffic load uncertainty. Therefore, along with the previous findings for a network with fixed PWCE, we can finally conclude that APWCE provides a complete paradigm for operation of a dynamic protected transport network as a simple, fast-restoration, and capacity-efficient system.

CHAPTER 10
CONCLUDING DISCUSSION AND FUTURE WORK

10.1. SUMMARY OF THESIS

Network survivability and dynamic service provisioning are two of the most important aspects for the future transport networks. SBPP-based dynamic survivable service provisioning is currently the *de facto* provisioning approach receiving much attention from both academia and industry. Nevertheless, this method suffers from the drawback of operational complexity. Specifically, it has a complicated network status database, time-consuming working and protection route searching process, and complex signaling process. Moreover, SBPP is a path-oriented survivability method, which thereby inherently demonstrates a slower restoration speed compared to span-oriented methods such as span-protecting *p*-cycles.

To overcome the above difficulties, we proposed an alternative provisioning method, termed *p*-cycle-based PWCE, to provision dynamic survivable services, which has advantages over the conventional SBPP method in terms of simplicity, faster restoration speed, and comparable or even better blocking performance. We combined the concept of PWCE with the *p*-cycle technique, owing to the extraordinarily simple operation of PWCE and the feature of *ring-like switching speed and mesh-like spare capacity efficiency* of *p*-cycles. The GMPLS-based control system was developed to operate PWCE-based network. The routing and signaling components were developed and compared with those under SBPP. It was found that the PWCE method is advantageous, with much lower control overhead and one fewer session in the signaling process. Also, the network status database comparison indicated that PWCE has a much simpler database to record only the information on working capacity, while SBPP needs to record the information on not only working and spare capacities but also the spare capacity sharing relationship.

To maximize the utilization of spare capacity, based on the forcer theory we developed a wide range of volume-maximization models for envelope design. A total of eight design scenarios were specifically considered. Based on the envelopes designed by

these volume-maximization models, the blocking performance of PWCE was further evaluated in comparison with SBPP. It was found that PWCE and SBPP achieve comparable performances, and with the increase of network nodal degree, PWCE performs better and better, and can finally outperform SBPP to demonstrate a lower blocking probability. The investigation on how the network nodal degree can affect the blocking performance of PWCE also indicated that when the average nodal degree of a network grows to be more than 3.0 or 3.2, PWCE can achieve a comparable or even better blocking performance than SBPP. Therefore, it is more efficient to implement PWCE in a network with denser connectivity.

Furthermore, to thoroughly understand the technique, we also investigated PWCE under the assumptions of uniform capacity and modular capacity. The findings are analogous to those discovered earlier, namely that PWCE can achieve a comparable performance to that of SBPP. Thus, this further ascertains that the good performance of PWCE is not specific to certain network design scenarios, but general enough to fit all kinds of networks with uniform and non-uniform capacities.

Finally, in view of traffic load or demand uncertainty in real network operations, the PWCE approach was further extended to support the capability of reoptimization and reconfiguration in order to adapt the envelopes to match the evolving traffic loads in all time. For this, the concept called adaptive PWCE (APWCE) was specifically proposed to allow the envelope to be reconfigured, conforming to the real network load distribution. Specifically, the aspects of the triggering mechanism, the envelope template, the capacity budget for the reconfiguration, and the envelope reoptimizing models, were discussed in detail. The experimental results indicate that the APWCE approach can continually and automatically adapt an operating envelope to match a real non-stationary traffic load pattern. This ensures that the PWCE-based service provisioning can resist the influence of traffic load fluctuation or uncertainty and maintain a sound network blocking performance any time.

10.2. Future Work

Further research can be explored on the following topics:

10.2.1. Concept of Protected Working Capacity Tunnel (PWCT)

Extended from the concept of PWCE, we can further propose a more user-oriented scheme termed Protected Working Capacity Tunnel (PWCT). The new scheme provisions dynamic survivable services on the node-pair basis and guarantees the customer's Quality of Service (QoS) in terms of blocking probability. Conventional path-based survivability schemes such as SBPP, path restoration, flow *p*-cycles, etc., are eligible to apply this provisioning scheme. Compared to PWCE, although the overall network blocking performance may not be as good as that of PWCE, PWCT is capable of even simpler operation in maintaining protected tunnels between node pairs. Typically, the concept of PWCT can be efficiently applied under some network situations like highway express links or paths, on which the data traffic is so voluminous that dedicated and fast protection tunnels are necessary. It can also be applied to some cross-domain situations, where the channels that interconnect the domains require dedicated protection and fast restoration. Nonetheless, PWCT should be distinguished from the conventional 1+1 or 1:1 dedicated path protection. PWCT allows spare capacity sharing, while 1+1 and 1:1 do not. Thus, a better spare capacity efficiency can be expected from PWCT than from the 1+1 or 1:1 dedicated protection.

10.2.2. PWCE Network Design Supporting Multi-QoP

With the development of PWCE-based service provisioning scheme, we now can classify network services according to their QoP levels into five types: preemptible, best effort, SBPP, PWCE, and 1+1 or 1:1. To provision all these types of services, a network with multiple QoP-levels should be designed. Two possible strategies can be used to instruct the design, namely *independent strategy* and *integrated strategy*. In the independent strategy, each of the networks dedicated to supporting a certain type of QoP is first independently designed, and then these networks are placed onto the same physical network to form a network supporting multi-QoP services. In contrast, the integrated design strategy requires that all the networks be designed in an aggregate or mixed fashion. The advantage of this is that the strategy allows for spare capacity sharing among the networks that support different types of QoPs. For example, the networks supporting SBPP services and PWCE services can be designed together to allow them to

share a common pool of spare capacity. This can greatly improve the efficiency of spare capacity sharing. However, the penalty of this is of course more complicated design and operation. The advantages and disadvantages of the two strategies should be further evaluated and compared.

10.3. Other Contributions of This Doctoral Work

Besides the contributions presented and discussed herein, the overall doctoral project associated with this thesis made several other notable contributions, which involve 4 journal papers, 6 peer-reviewed conference papers, two book chapters, and two patent applications. In addition, I have developed a Visual C++ based Optical Network Simulator (ONS) for my doctoral research, and co-developed a Java-based emulator to demonstrate the concept of Protected Working Capacity Envelope (PWCE).

10.3.1. Journal Papers

[A1] **Gangxiang Shen**, Wayne D. Grover, "Design and Performance of Protected Working Capacity Envelopes Based on *p*-Cycles for Dynamic Provisioning of Survivable Services," *OSA Journal of Optical Networking*, vol. 4, no. 7, April 2005, pp. 361-390.

[A2] **Gangxiang Shen**, Wayne D. Grover, "Segment-based Approaches to Survivable Translucent Network Design under Various Ultra Long Haul System Reach Capabilities **(Invited)**," *OSA Journal of Optical Networking special issue (SON)*, vol. 3, no. 1, January 2004, pp. 1-24.

[A3] **Gangxiang Shen**, Wayne D. Grover, "Extending the *p*-Cycle Concept to Path-Segment Protection for Span and Node Failure Recovery," *IEEE Journal on Selected Area of Communications (JSAC) special issue (Optical communications and networking series 2003)*, vol. 21, no. 8, October 2003, pp. 1306-1319.

[A4] **Gangxiang Shen**, Wayne D. Grover, Tee Hiang Cheng, and Sanjay K. Bose, "Sparsely Placement of Electronic Switching Nodes for Low Blocking in Translucent Optical Networks," *OSA Journal of Optical Networking*, vol. 1, no. 12, December 2002, pp. 424-441.

10.3.2. PEER-REVIEWED CONFERENCE PAPERS

[B1] **Gangxiang Shen**, Wayne D. Grover, "Survey and Performance Comparison of Dynamic Provisioning Methods for Optical Shared Backup Path Protection," in the proceedings of the *1st IEEE International Workshop on Guaranteed Optical Service Provisioning (GOSP)*, Boston, Oct. 2005, pp. 387-396.

[B2] **Gangxiang Shen**, Wayne D. Grover, "Automatic Lightpath Service Provisioning with an Adaptive Protected Working Capacity Envelope based on *p*-Cycles," in the proceedings of the *5th International Workshop on Design of Reliable Communication Networks (DRCN)*, Italy, 2005, pp. 375-383.

[B3] **Gangxiang Shen**, Wayne D. Grover, "Design and Performance of Protected Working Capacity Envelopes based on *p*-Cycles: a Fast, Simple, and Scalable Framework for Dynamic Provisioning of Survivable Services (**Invited**)," in the proceedings of *SPIE Asia-Pacific Optical Communications (APOC)*, Beijing China on Nov. 7-11, 2004, pp. 519-533.

[B4] **Gangxiang Shen**, Wayne D. Grover, "Exploiting Forcer Structure to Serve Uncertain Demands and Minimize Redundancy of *p*-Cycle Networks," in the proceedings of the *4th Annual Optical Networking and Communications Conference (OptiComm)*, Dallas, Oct. 2003, pp. 59-70.

[B5] Wayne D. Grover, **Gangxiang Shen**, "Extending the *p*-cycle concept to path-segment protection," in the proceedings of *IEEE International Conference on Communications (ICC)*, Anchorage, May 2003, pp. 1314-1319.

[B6] **Gangxiang Shen**, Wayne D. Grover, "Capacity Requirements for Network Recovery from Node Failure with Dynamic Path Restoration," in the proceedings of *Optical Fiber Communications (OFC)*, Atlanta, March 2003, vol. 2, pp. 775-777.

10.3.3. BOOK CHAPTERS

[C1] Wayne D. Grover, John Doucette, Dion Leung, Adil Kodian, Anthony Sack, Matthieu Clouquer, **Gangxiang Shen**, Wong Sukcharoenkana, "Design of Survivable Networks Based on *p*-Cycles," in the *Handbook of Optimization in Telecommunications*, P. M. Pardalos, M. G. C. Resende (editors), Kluwer Academic Publishers, 2005.

[C2] **Gangxiang Shen**, Wayne D. Grover, "Design of Protected Working Capacity Envelopes Based on *p*-Cycles: An Alternative Framework for Survivable Automated Lightpath Provisioning **(Invited)**," in *Performance Evaluation and Planning Methods for the Next Generation Internet*, A. Girard, B. Sansò, F. Vazquez-Abad (editors), Kluwer Academic Publishers, 2005.

10.3.4. PATENTS PENDING

[D1] Wayne D. Grover, **Gangxiang Shen**, "Protecting a Network Using Protected Working Capacity Envelopes," US Patent Application 20050195739, Feb. 2005.

[D2] **Gangxiang Shen**, Wayne D. Grover, "Path Segment Protecting *p*-cycles," Canadian Patent Application 2435193, July 14, 2003.

[D3] Wayne D. Grover, **Gangxiang Shen**, "Path Segment Protecting *p*-cycles," US Patent Application 20040109407, July 14, 2003.

10.3.5. SOFTWARE DEVELOPMENT

A) Tool of Optical Network Simulator (ONS)

Since I joined Network Systems Group, TRLabs, I have written more than ten thousands of lines of C++ codes for my research, and developed a simple and user-friendly software tool—Optical Network Simulation (ONS)—as shown in Figure 10.1. The tool is used to obtain data files input to an optimization software AMPL/CPLEX and run blocking performance simulations for dynamic lightpath service provisioning, such as PWCE and SBPP. A wide range of network survivability techniques have been incorporated in the tool, including span restoration, span-based *p*-cycles, path restoration (without or with stub release), flow *p*-cycles, shared backup path protection (SBPP), path-segment restoration, shared backup segment protection (SBSP), node failure recovery, etc. Also, to study the performance of dynamic service provisioning, various routing algorithms were implemented in the tool to simulate the cases with service protection and without protection.

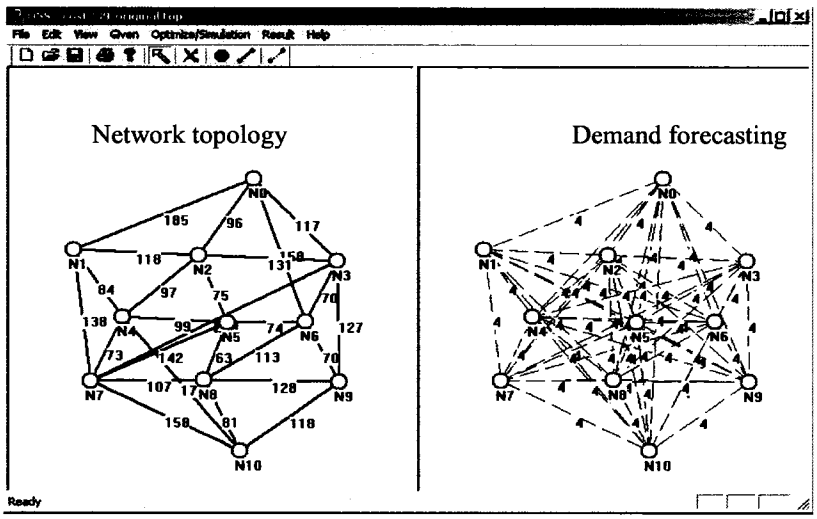


Figure 10.1. Snapshot of Optical Network Simulator (ONS)

B) PWCE Concept Emulator

I also co-developed a Java-based emulator to demonstrate the concept of Protected Working Capacity Envelope (PWCE). I was one of the key members of PWCE emulator developing team. My major role was to help a summer-internship undergraduate student, Dimitri Baloukov, understand the detailed concept and design process of PWCE, and guide him to model dynamic lightpath service provisioning and define an object-oriented Java class hierarchy. Figure 10.2 shows the snapshot of this emulator.

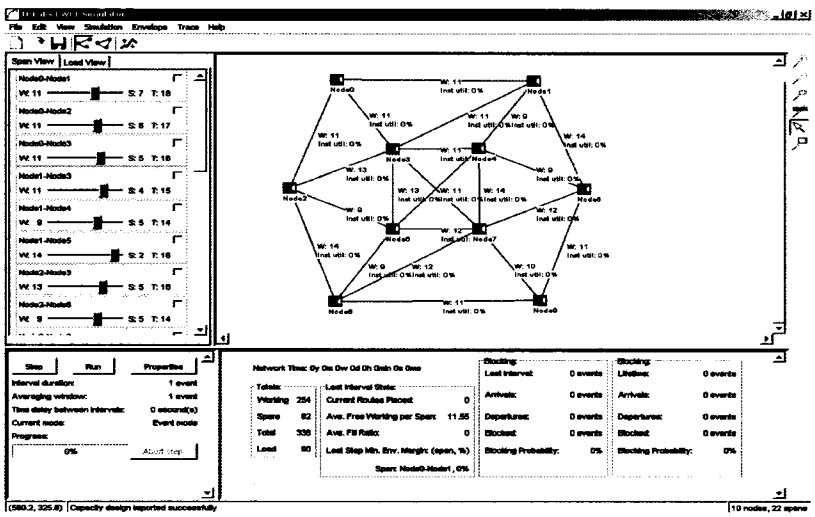


Figure 10.2. Snapshot of PWCE Concept Emulator

REFERENCES

[Amst89] S. R. Amstutz, "Burst Switching-An Update," *IEEE Communications Magazine*, vol. 27, no. 9, Sep. 1989, pp. 50-57.

[AsCh04] G. R. Ash, P. Chemeouil, "20 Years of Dynamic Routing in Circuit-Switched Networks: Looking Backward to the Future," *IEEE Global Communications Newsletter*, Oct. 2004. Available: http://www.comsoc.org/pubs/gcn/gcn1004.html, April 4, 2005 [date accessed].

[AwBe01] D. Awduche, et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels," *RFC 3209*, (Standards Track), Dec. 2001.

[BaRo02] I. Baldine, G. N. Rouskas, H. G. Perros, D. Stevenson, "JumpStart: A Just-In-Time Signaling Architecture for WDM Burst-Switched Networks," *IEEE Communications Magazine*, vol. 40, no. 2, Feb. 2002, pp. 82-89.

[Berg03] Berger, L. (Editor), "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description," *RFC 3471*, (Standard track), Jan. 2003.

[BoLa02] E. Bouillet, J. F. Labourdette, G. Ellinas, R. Ramamurthy, S. Chaudhuri, "Stochastic Approaches to Compute Shared Mesh Restored Lightpaths in Optical Network Architectures," in the proceedings of *INFOCOM'02*, June 2002, pp. 801-807.

[ChGa92] I. Chlamtac, A. Ganz, G. Karmi, "Lightpath communication: An Approach to High Bandwidth Optical WAN's," *IEEE Transaction on Communications*, vol. 40, no. 7, July 1992, pp. 1171-1182.

[ChSt03] A. L. Chiu, J. Strand, "An Agile Optical Layer Restoration Method for Router Failures," *IEEE Network*, vol. 17, no. 2, March-April 2003, pp. 38-42.

[ClGr01] M. Clouqueur, W. D. Grover, D. Leung, O. Shai, "Mining the Rings: Strategies for Ring-to-Mesh Evolution," in the proceedings of the *3rd International Workshop on Design of Reliable Communication Networks (DRCN)*, Oct. 2001, pp.113-120.

[ClGr02] M. Clouqueur, W. D. Grover, "Dual Failure Availability Analysis of Span-Restorable Mesh Networks," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, May 2002, pp. 810-821.

[DaHa98] S. L. Danielsen, P. B. Hansen, K. E. Stubkjaer, "Wavelength Conversion in Optical Packet Switching," *IEEE Journal of Lightwave Technology*, vol. 16, no. 12, Dec. 1998, pp. 2095-2108.

[DaRe00] B. Davie, Y. Rekhter, *MPLS Technology and Applications*, Morgan Kaufmann Publishers, San Francisco, 2000.

[DeGr99] P. Demeester, M. Gryseels, et al., "Resilience in Multilayer Networks," *IEEE Communications Magazine*, vol. 37, no. 8, Aug. 1999, pp. 70-75.

[DoFa02] P. Dobbelaere, K. Falta, S. Gloeckner, S. Patra, "Digital MEMS for Optical Switching," *IEEE Communications Magazine*, vol. 40, no. 3, March 2002, pp. 88-95.

[DoGr00] J. Doucette, W. D. Grover, "Influence of Modularity and Economy-of-scale Effects on Design of Mesh-restorable DWDM Networks," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, Oct. 2000, pp. 1912-1923.

[DoGr02] J. Doucette, W. D. Grover, "Capacity Design Studies of Span-Restorable Mesh Networks with Shared-Risk Link Group (SRLG) Effects," in the proceedings of the *3rd Annual Optical Networking and Communications Conference (OptiComm)*, July-Aug. 2002, pp.25-38.

[DoGr03] J. Doucette, W. D. Grover, "Node-Inclusive Span Survivability in an Optical Mesh Transport Network," in the proceedings of *National Fiber Optic Engineers Conference (NFOEC)*, Sep. 2003.

[DuGr94] D. A. Dunn, W. D. Grover, M. H. MacGregor, "A Comparison of k-Shortest Paths and Maximum Flow Methods for Network Facility Restoration," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 1, Jan. 1994, pp. 88-99.

[ElHa00] G. Ellinas, A. G. Hailemariam, and T. E. Stern, "Protection Cycles in Mesh WDM Networks," *IEEE Journal on Selected Area in Communications*, vol. 18, no. 10, Oct. 2000, pp. 1924–1937.

[FaDa04] M. Fawaz, B. Daheb, O. Audouin, M. Du-Pond, G. Pujolle, "Service Level Agreement and Provisioning in Optical Networks," *IEEE Communications Magazine*, vol. 42, no. 1, Jan. 2004, pp. 36-43.

[FiWa94] W. Fischer, E. Wallmeier, T. Worster, S. P. Davis, A. Hayter, "Data Communications Using ATM: Architectures, Protocols, and Resource Management," *IEEE Communications Magazine*, vol. 32, no. 8, Aug. 1994, pp. 24-33.

[GaBl98] R. Gaudino, D. J. Blumenthal, "WDM Channel Equalization Based on Subcarrier Signal Monitoring," in the proceedings of *Optical Fiber Communications (OFC)*, Feb. 1998, pp. 167 – 168.

[GaRe98] P. Gambini, M. Renaud, et al., "Transparent Optical Packet Switching: Network Architecture and Demonstrators in the KEOPS Project," *IEEE Journal on Selected Area in Communications*, vol. 16, no. 7, Sep. 1998, pp. 1245-1259.

[GeRa00] O. Gerstel, R. Ramaswami, "Optical Layer Survivability: A Services Perspective," *IEEE Communications Magazines*, vol. 38, no. 3, Mar. 2000, pp.104-113.

[Gile97] C. R. Giles, "Tutorial on Fiber Gratings in Lightwave Network Applications," in the proceedings of *Optical Fiber Communications (OFC)*, Feb. 1997, pp. 64.

[GrDo01] W. D. Grover, J. Doucette, "Topological Design of Survivable Mesh-based Transport Networks," *Annals of Operations Research: Topological Network Design in Telecommunication Systems*, vol. 106, no. 9, Sep. 2001, pp. 79-125.

[GrDo02] W. D. Grover, J. Doucette, "Advances in Optical Network Design with *p*-Cycles: Joint Optimization and Pre-selection of Candidate *p*-Cycles," in the proceedings of *IEEE-LEOS Topical Meetings*, July 2002.

[GrLi99] W. D. Grover, D. Y. Li, "The Forcer Concept and Express Route Planning in Mesh-Survivable Networks," *Journal of Network and Systems Management*, vol. 7, no. 2, Jun. 1999, pp. 199-223.

[GrMa00] W. D. Grover, R. G. Martens, "Optimized Design of Ring-Mesh Hybrid Networks," in the proceedings of *International Workshop on Design of Reliable Communication Networks (DRCN)*, April 2000, pp. 291-297.

[Grov97] W. D. Grover, "Self-Organizing Broadband Transport Networks," *Proceedings of the IEEE*, vol. 85, no.10, Oct. 1997, pp. 1582-1611.

[Grov02] W. D. Grover, "Understanding *p*-Cycles, Enhanced Rings, and Oriented Cycle Covers," in the proceedings of *International Conference on Optical Communications and Networks (ICOCN)*, Nov. 2002, pp. 305-308.

[Grov04a] W. D. Grover, "The Protected Working Capacity Envelope Concept: An Alternate Paradigm for Automated Service Provisioning," *IEEE Communications Magazine*, vol. 42, no. 1, Jan. 2004, pp. 62-69.

[Grov04b] W. D. Grover, Mesh-based Survivable Networks, *Options and Strategies for Optical, MPLS, SONET, and ATM Networking*, Prentice Hall, 2004.

[GrSt98a] W. D. Grover, D. Stamatelakis, "Cycle-Oriented Distributed Preconfiguration: Ring-like Speed with Mesh-like Capacity for Self-planning Network Restoration," in the proceedings of *IEEE International Conference on Communications (ICC)*, 1998, pp. 537–543.

[GrSt98b] W. D. Grover, D. Stamatelakis, "Scalable ADM-like Nodal Device for *p*-Cycle Restorable Networks," patent filed in USA, Oct. 1998.

[GrSt00] W. D. Grover, D. Stamatelakis, "Bridging the Ring-mesh Dichotomy with *p*-Cycles," in the proceedings of *International Workshop on Design of Reliable Communication Networks (DRCN)*, Apr. 2000, pp. 92-104.

[HaKo02] O. Hauser, M. Kodialam, T. V. Lakshman, "Capacity Design of Fast Path Restorable Optical Networks," in the proceedings of *INFOCOM'02*, vol. 2, June 2002, pp. 817-826.

[HeBy95] M. Herzberg, S. Bye, A. Utano, "The Hop-Limit Approach for Spare-Capacity Assignment in Survivable Networks, *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, June 1995, pp. 775-784.

[HeRe00] P. Heywood, M. Reardon, "Alcatel Backs the Bubble," *Lightreading News Analysis*, http://www.lightreading.com/document.asp?doc_id=1315, July 2000.

[HoHu02] P. Ho, T. M. Hussein, "A Framework for Service-guaranteed Shared Protection in WDM Mesh Networks," *IEEE Communications Magazine*, vol. 40, no. 2, Feb. 2002, pp. 97-103.

[IrMa98] R. R. Iraschko, M. MacGregor, W. D. Grover, "Optimal Capacity Placement for Path Restoration in STM or ATM Mesh Survivable Networks," *IEEE/ACM Transactions on Networking*, vol.6, no.3, June 1998, pp. 325-336.

[IrGr00] R. R. Iraschko, W. D. Grover, "A Highly Efficient Path-Restoration Protocol for Management of Optical Network Transport Integrity," *IEEE Journal on Selected Areas in Communications*, vol.18, no.5, May 2000, pp. 779-793.

[Isla02] M. N. Islam, "Raman Amplifiers for Telecommunications," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 8, no. 3, May-June 2002, pp. 548-559.

[Jamo02] B. Jamoussi (editor), et al., "Constraint-Based LSP Setup using LDP," *RFC 3212*, (Standards Track), Jan. 2002.

[KaSa94] R. Kawamura, K. Sato, I. Tokizawa, "Self-Healing ATM Networks based on Virtual Path Concept," *IEEE Journal on selected Areas in Communications*, vol. 12, no. 1, Jan. 1994, pp. 120-127.

[KiKo00] S. Kini, M. Kodialam, T. V. Laksham, C. Villamizar, "Shared Backup Label Switched Path Restoration," *Internet Draft*, draft-kini-restoration-shared-backup-00.txt, Aug. 2000.

[KoAc96] M. Kovacevic, A. S. Acampora, "Electronic Wavelength Translation in Optical Networks," *IEEE Journal on Lightwave Technology*, vol. 4, no. 6, June 1996, pp. 1161-1169.

[KoLa00] M. Kodialam, T. V. Lakshman, "Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration," in the proceedings of *INFOCOM'00*, 2000, pp. 902-911.

[KoRe03a] K. Kompella, Y. Rekhter (editors), "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching," *Internet draft*, draft-ietf-ccamp-ospf-gmpls-extensions-12.txt, (Standard Track), Oct. 2003.

[KoRe03b] K. Kompella, Y. Rekhter (editors), "Routing Extensions in Support of Generalized Multi-Protocol Label Switching," *Internet draft*, draft-ietf-ccamp-gmpls-routing-09.txt, (Standard Track), Oct. 2003.

[Lang03] J. Lang (editor), "Link Management Protocol (LMP)," *Internet Draft*, draft-ietf-ccamp-lmp-10.txt, (Standard Track), Oct. 2003.

[LeGr02] D. Leung, W. D. Grover, "Comparative Ability of Span Restorable and Path Protected Network Designs to Withstand Uncertainty in the Demand Forecast," in the proceedings of *National Fiber Optic Engineers Conference (NFOEC)*, Sep. 2002, pp.1450-1461.

[LeLi93] K. Lee, Victor O. K. Li, " A Wavelength-Convertible Optical Network," *IEEE Journal of Lightwave Technology*, vol. 11, no. 5/6, May/June 1993, pp. 962-970.

[LiWa02] G. Li, D. Wang, C. Kalmanek, R. Doverspike, "Efficient Distributed Path Selection for Shared Restoration Connections," in the proceedings of *INFOCOM'02*, June 2002, pp. 140-149.

[LiYa02] G. Li, J. Yates, D. Wang, C. Kalmanek, "Control Plane Design for Reliable Optical Networks," *IEEE Communications Magazine*, vol. 40, no. 2, Feb. 2002, pp. 90-96.

[MaAn98] J. Manchester, J. Anderson, B. Doshi, S. Dravida, "IP over SONET," *IEEE Communications Magazine*, vol. 36, no. 5, May 1998, pp. 136-142.

[MaKu03] X. Ma, G. S. Kou, "Optical Switching Technology Comparison: Optical MEMS vs. Other Technologies," *IEEE Communications Magazine*, vol. 41, no. 11, Nov. 2003, pp. s16-s23.

[Mann03] E. Mannie (editor), "Generalized Multi-Protocol Label Switching Architecture," *Internet Draft*, draft-ietf-ccamp-gmpls-architecture-07.txt, (Standard Track), May 2003.

[MéBa02] M. Médard, R. A. Barry, S. G. Finn, W. He, S. Lumetta, "Generalized Loop-back Recovery in Optical Mesh Networks," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, Feb. 2002, pp. 153-164.

[MeRe87] R. J. Mears, L. Reekie, I. M. Jauncey, D. N. Payne, "Low Noise Erbium-Doped Fiber Amplifier at 1.54μm," *IEE Electronic Letters*, vol. 23, no. 19, 1987, pp. 1026-1028.

[MoAz98] A. Mokhtar, M. Azizoglu, "Adaptive Wavelength Routing in All-optical Networks," *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, Apr.1998, pp. 197-206.

[MoGr01] D. Morley, W.D. Grover, "Tabu Search Optimization of Optical Ring Transport Networks," in the proceedings of *IEEE Globecom 2001*, Nov. 2001, vol. 4, pp. 2160-2164.

[MoMu01] G. Mohan, C. S. Murthy, A. K. Somani, "Efficient Algorithm for Routing Dependable Connections in WDM Optical Networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 5, Oct. 2001, pp. 553-566.

[NeSt95] L. Nederlof, K. Struyve, et al., "End-to-End Survivable Broadband Networks," *IEEE Communications Magazine*, vol. 33, no. 9, Sep. 1995, pp. 63-70.

[Omah95] M. J. O'Mahony, "Optical Multiplexing in Fiber Networks: Progress in WDM and OTDM," *IEEE Communications Magazine*, vol. 33, no. 12, Dec. 1995, pp. 82-88.

[OuZh03] C. Ou, J. Zhang, H. Zang, L. Sahasrabuddhe, B. Mukherjee, "Near-optimal Approaches for Shared-path Protection in WDM Mesh Networks," in the proceedings of *International Conference on Communications (ICC)*, May 2003, pp. 1320-1324

[QiXu02] C. Qiao, D. Xu, "Distributed Partial Information Management (DPIM) Schemes for Survivable Networks - Part I", in the proceedings of *INFOCOM'02*, June 2002, pp. 302-311.

[RaDa99a] B. Ramamurthy, D. Datta, H. Feng, J. P. Heritage, B. Mukherjee, "Transparent vs. Opaque vs. Translucent Wavelength-Routed Optical Networks," in the proceedings of *Optical Fiber Communications (OFC)*, vol. 1, 1999, pp. 59-61.

[RaDa99b] B. Ramamurthy, D. Datta, H. Feng, J. P. Heritage, B. Mukherjee, "Impact of Transmission Impairments on the Teletraffic Performance of Wavelength-Routed Optical Networks," *IEEE Journal on Lightwave Technology*, vol. 17, no. 10, Oct. 1999, pp. 1713-1723.

[RaLu04] B. Rajagopalan, J. Luciani, D. Awduche, "IP over Optical Networks: A Framework," *RFC 3717*, (Informational), Mar. 2004.

[Rear02] M. Reardon, "Juniper Goes Terabit With the T640," *Lightreading News Analysis*, http://www.lightreading.com/document.asp?doc_id=14335&site=lightreading, April 2002.

[RoVi01] E. Rosen, et al., "Multiprotocol Label Switching Architecture," *RFC 3031*, (Standards Track), Jan. 2001.

[Sart01] B. Sartorius,"3R All-Optical Signal Regeneration," in the proceedings of the *27th European Conference on Optical Communication (ECOC)*, vol. 5, 2001, pp. 98-125.

[ScGr02] D. A. Schupke, C. G. Gruber, A. Autenrieth, "Optimal Configuration of p-Cycles in WDM Networks," in the proceedings of *International Conference on Communications (ICC)*, vol. 5, Apr./May 2002, pp. 2761-2765.

[SeYa02] P. Sebos, J. Yates, D. Rubenstein, A. Greenberg, "Effectiveness of Shared Risk Link Group Auto-Discovery in Optical Networks," in the proceedings of *Optical Fiber Communications (OFC)*, Mar. 2002, pp. 493-495.

[ShBo01a] G. Shen, S. K. Bose, T. H. Cheng, C. Lu, T. Y. Chai, "Performance Study on a WDM Packet Switch with Limited-Range Wavelength Converters," *IEEE Communications Letters*, vol. 5, no. 10, Sep. 2001, pp. 432-434.

[ShBo01b] G. Shen, S. K. Bose, T. H. Cheng, et al., "Heuristic Algorithm for Efficient Light-Path Routing and Wavelength Assignment in WDM Networks under Dynamically Varying Loads," *Computer Communications*, vol. 24, no. 3-4, Feb. 2001, pp. 364-373.

[Shen03] G. Shen, "A Framework of *p*-Cycle-Based Protected Working Capacity Envelope (PWCE) for Dynamic Protected Lightpath Service Provisioning," *Interim Technique Report for Ph.D. Candidacy Examination*, Nov. 2003.

[ShGr02] G. Shen, W. D. Grover, T. H. Cheng, S. K. Bose, "Sparsely Placement of Electronic Switching Nodes for Low Blocking in Translucent Optical Networks," *OSA Journal of Optical Networking*, vol. 1, no. 12, Dec. 2002, pp. 424-441.

[ShGr03a] G. Shen, 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, Oct. 2003, pp. 1306-1319.

[ShGr03b] G. Shen, W. D. Grover, "Exploiting Forcer Structure to Serve Uncertain Demands and Minimize Redundancy of *p*-Cycle Networks," in the proceedings of the *4th Annual Optical Networking and Communications Conference (OptiComm)*, October 2003, pp. 59-70.

[ShGr04a] G. Shen, W. D. Grover, "Segment-Based Approaches to Survivable Translucent Network Design under Various Ultra Long Haul System Reach," *OSA Journal on Optical Networking*, vol. 3, no. 1, Jan. 2004, pp. 1-24.

[ShGr04b] G. Shen, W. D. Grover, "Design and Performance of Protected Working Capacity Envelopes based on *p*-Cycles: a Fast, Simple, and Scalable Framework for Dynamic Provisioning of Survivable Services (Invited)," in the proceedings of *SPIE Asia-Pacific Optical Communications (APOC)*, Beijing, China, Nov. 2004, pp. 519-533.

[ShGr04c] G. Shen and W. D. Grover, "A Framework for Dynamic Survivable Service Provisioning Based on *p*-Cycles and the Protected Working Capacity Envelope (PWCE) Concept," *TRLabs Technical Report*, TR-04-06, Oct. 2004.

[ShGr05a] G. Shen, W. D. Grover, "Design of Protected Working Capacity Envelopes Based on *p*-Cycles: An Alternative Framework for Survivable Automated Lightpath

Provisioning," a book chapter in *Performance Evaluation and Planning Methods for the Next Generation Internet*, Andre Girard, Brunilde Sanso, Felisa Vazquez-Abad (editors), Kluwer Academic Publishers, Feb. 2005.

[ShGr05b] G. Shen, W. D. Grover, "Design and Performance of Protected Working Capacity Envelopes based on *p*-Cycles for Dynamic Provisioning of Survivable Services," *OSA Journal of Optical Networking*, vol. 4, no. 7, April 2005, pp. 361-390.

[ShGr05c] G. Shen, W. D. Grover, "Automatic Lightpath Service Provisioning with an Adaptive Protected Working Capacity Envelope based on *p*-Cycles," in the proceedings of the *5th International Workshop on Design of Reliable Communication Networks (DRCN)*, Italy, 2005, pp. 375-383.

[ShGr05d] G. Shen, W. D. Grover, "Survey and Performance Comparison of Dynamic Provisioning Methods for Optical Shared Backup Path Protection," in the Proceedings of the *1st IEEE International Workshop on Guaranteed Optical Service Provisioning (GOSP)*, Boston, Oct. 2005, pp. 387-396.

[ShMu00] Y. Shun, B. Mukherjee, S. Dixit, "Advances in Photonic Packet Switching: An Overview," *IEEE Communications Magazine*, vol. 38, no. 2, Feb. 2000, pp. 84-94.

[SiGo03] K. M. Sivalingam, S. Gowda, R. Shenai, H. Cankaya, "Performance Evaluation of TCP over Optical Burst-Switched (OBS) WDM Networks," in the proceedings of *IEEE International Conference on Communications (ICC)*, Anchorage, Alaska, vol. 2, May 2003, pp. 1433-1437.

[Stam97] D. Stamatelakis, Theory & Algorithms for Preconfiguration of Spare Capacity in Mesh Restorable Networks, M.Sc. Thesis, Univ. Alberta, Canada, 1997.

[StGr98] D. Stamatelakis, W. D. Grover, "OPNET Simulation of Self-organizing Restorable SONET Mesh Transport Networks," in the proceedings of *OPNETWORKS*, Apr. 1998, paper 04.

[StGr00a] D. Stamatelakis, W. D. Grover, "Theoretical Underpinnings for the Efficiency of Restorable Networks using Pre-configured Cycles ("*p*-Cycles")," *IEEE Transaction on Communications*, vol. 48, no.8, Aug. 2000, pp. 1262-65.

[StGr00b] D. Stamatelakis, W. D. Grover, "IP layer restoration and network planning based on virtual protection cycles," *IEEE Journal on Selected Areas in Communications*, vol.18, no.10, Oct. 2000, pp. 1938-1949.

[SuAz98] R. Subramaniam, M. Azizoglu and A. K. Somani, "On the Optimal Placement of Wavelength Converters in Wavelength Routed Networks," in the proceedings of *INFOCOM'98*, vol. 2, 1998, pp. 902-909.

[SuTa84] J. W. Surrballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol.14, no.2, Feb. 1984, pp.325-336.

[Swal99] G. Swallow, "MPLS Advantages for Traffic Engineering," *IEEE Communications Magazine*, vol. 37, no. 12, Dec. 1999, pp. 54-57.

[Uets04] H. Uetsuka, "AWG Technologies for Dense WDM Applications," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 10, no. 2, Mar.-April 2004, pp. 393-402.

[WuTh92] T. H. Wu, *Fiber Network Service Survivability*, Artech House, 1992.

[YaLa96] J. Yates, J. Lacey, D. Everitt, M. Summerfield, "Limited-Range Wavelength Translation in All-Optical Networks," in the proceedings of *INFOCOM'96*, vol. 3, 1996, pp. 954-961.

[YaSa95] K. Yamagishi, N. Sasaki, K. Morino, "An Implementation of a TMN-based SDH Management System in Japan," *IEEE Communications Magazine*, vol. 33, no. 3, Mar. 1995, pp. 80-85.

[YaZe03] Y. Yang, Q. Zeng, H. Zhao, "Dynamic Survivability in WDM Mesh Networks under Dynamic Traffic," *Photonic Network Communications*, vol. 6, no. 1, 2003, pp. 5-24.

[YeLa01] T. W. Yeow, K.L.E. Law, A. Goldenberg, "MEMS Optical Switches," *IEEE Communications Magazine*, vol. 39, no. 11, Nov. 2001, pp. 158-163.

[YoJe97] M. Yoo, M. Jeong, C. Qiao, "A High Speed Protocol for Bursty Traffic in Optical Networks," in the proceedings of *SPIE's All-Optical Communication Systems: Architecture, Control and Protocol Issues*, vol. 3230, Nov. 1997, pp. 79-90.

[ZaMu01] H. Zang, B. Mukherjee, "Connection Management for Survivable Wavelength-routed WDM Mesh Networks," *SPIE Optical Network Magazine*, vol. 2, no. 4, July/Aug. 2001, pp. 17-28.