2007

Integrated mechanisms for QoS and restoration in mesh transport networks

Gong, Ming

Integrated Mechanisms for QoS and Restoration in Mesh Transport Networks

By

Ming Gong

A Thesis Presented to Lakehead University

in Partial Fulfillment of the Requirement for the Degree of

Master of Science in Control Engineering

Thunder Bay, Ontario, Canada, 2007

Your file Votre référence
ISBN: 978-0-494-31855-3
Our file Notre référence
ISBN: 978-0-494-31855-3

ABSTRACT

Survivable networks have the capability to survive from the events of network components failures. The resilience mechanisms in these networks protect and restore the impaired communication paths by using spare capacity. On the other hand, Quality of Service (QoS) mechanisms focus on network capabilities that provide the facilities to differentiate network traffic and offer different levels of service to each class of traffic. Traditionally the survivability algorithms were applied at the physical (optical) layer, whereas the QoS mechanisms mainly applied at packet-forwarding level. Recent technological breakthroughs can now facilitate novel forwarding techniques for optical data bursts that make it possible to capture packets at the optical layer. A major challenge in the transfer of these ultrahigh-speed data bursts is to allocate resources according to QoS specifications and to provide spare capacity required to address link failures. Consequently, development of novel integrated strategies that facilitate implementation of QoS and survivability algorithms is of significant practical interest and is the primary focus of this study.

We present three novel mesh restoration techniques aimed at minimizing simultaneously the packet delay time and the restoration capacity in transport networks. These algorithms are: Two-step Delay-Constrained Pool Sharing (TDPS), Hybrid Pool Sharing (HPS), and One-Step Delay-Constrained Pool Sharing (ODPS). We show that how these schemes can be used to yield low end-to-end delay paths for demands in the network while still minimizing the spare capacity. Using simulation

methodology, we evaluate the performance of all of these algorithms and compare them with representative existing restoration/QoS algorithms.

We also present three novel integrated routing techniques aimed at minimizing the use of restoration capacity and enhancing the traffic load balancing in mesh transport networks. First, we present a Load Balancing Pool Sharing (LBPS) scheme and show how this scheme can be used to balance the loads on the network links while still minimizing the restoration capacity in the network. In order to eliminate the so called *trap-topology* problem, we introduce two new heuristic algorithms, called Iterative Simple Pool Sharing (ISPS) and Iterative Load Balancing Pool Sharing (ILBPS). We compare the capacity-usage, load balancing, and computation complexity performances of the LBPS and ILBPS algorithms with some representative algorithms, and we show that the proposed schemes can similarly or more evenly distribute the network traffic among network links than the other schemes at lower computation cost.

ACKNOWLEDGEMENTS

I gratefully acknowledge my supervisor Dr. Hassan Naser for his enthusiastic support and patient guidance. I could not have imagined having a better advisor and mentor for my master study, and without his common-sense, knowledge, perceptiveness and cracking-of-the-whip I would never have finished my thesis. I would like to thank my co-supervisor Dr. Abdelhamid Tayebi for his valuable suggestions. I would also like to acknowledge Professor Krishnamoorthy Natarajan and Professor Xiaoping Liu for giving me many good advices to my study and research.

I am grateful to all my classmates and friends in Lakehead University, for being the surrogate family during the two years I stayed at Thunder Bay and for their continued moral support and technical discussion.

Finally, I am forever indebted to my family in Shanghai, for their understanding and endless encouragement at any time.

CONTENTS

FIGURES INDEX

NOTATION

Table I. Notations for the different algorithms presented in this thesis

Notation	Description
b_r	Amount of bandwidth requested by demand r
J	number of links in the network
N	number of nodes in the network
M_i	total capacity on link i
A_i	available capacity on link i
W_i	total allocated working bandwidth on link i
B_j	total allocated backup bandwidth on link j
T_j	the maximum amount of backup bandwidth required on link j if a link in the working path fails
$C_W(i)$	cost of link i for working path computation
$C_B(j)$	cost of link j for backup path computation
$S_W(r)$	set of working links of demand r
$S_B(r)$	set of backup links of demand r
Ω	matrix of size $J \times J$ with elements k_{ij}, which is the amount of backup bandwidth needed on link j if link i fails
$C_1(i)$	cost of link i for the first shortest path computation
$C_2(j)$	cost of link j for the second shortest path computation
$S_1(r)$	set of links in the first shortest path of demand r
$S_2(r)$	set of links in the second shortest path of demand r
l_i	Load factor of link i
TL	set of trap-links
L_k	load on link k
L'	mean of the sample L_k
S	standard deviation of the sample L_k
V	average reserved capacity per accepted demand
n_d	number of accepted demands
d_i	average delay that the previously transmitted packets experienced on link i
D	average total delay per demand

Chapter 1 INTRODUCTION

1.1. Survivability Mechanisms

Survivability refers to the ability of a network to survive from the events of components failures. One important goal in survivable optical networks is that the network should be capable of providing service in the face of a wide range of possible dynamic events that include cable cuts and network-equipment failure (node failure). In backbone transport networks, node failures are usually avoided by standby devices. Also, the occurrence of multiple-link failures is rare, because of the very low probability of fiber link failure in the network. For these reasons, most research to date in survivable optical network design focuses on single link failures. This thesis considers single link failure scenario.

The resilience mechanisms in the survivable networks protect and restore the impaired communication paths by using spare capacity. Many powerful dynamic protection and restoration algorithms have been developed for networks with different topology configurations, such as ring and mesh [1]-[29].

1.1.1. Ring Network

Ring network topology is a closed path, which consists of consecutive nodes connected by point-to-point links [3][4][5]. Data is transmitted from one node to another node around the ring. There are three classical types of rings in optical networks, shown in Figure 1.1. 1) Two-fiber Unidirectional Path Switched Ring (UPSR) is a dual-fiber ring network. One ring fiber is used as the working ring, and

the second one is used for protection purposes. 2) Two-fiber Bidirectional Line Switched Ring (BLSR) is also a dual-fiber ring network. Unlike UPSR, both rings act as working and protection rings. The bandwidth of each ring is divided into two parts: the first part carries working traffic, and the second part is used for protection traffic. 3) Four-fiber BLSR uses four bidirectional fiber rings to interconnect the nodes in a network. Two rings are working rings; the other two are protection rings in the opposite direction.

Ring topology has been widely used for optical backbone networks, because it is the easiest way to interconnect every two nodes in a network and to provide protection for them. However, whenever a node is to be added into a ring, transmission links have to be installed between this node and its topologically adjacent nodes. Therefore it is difficult to add new nodes in ring networks. Susceptibility to failure is another problem of ring. When just one link fails, almost the whole ring suffers from this failure. For example, in Figure 1.2, the fiber between nodes A and D is broken. The working traffic on the working fiber between nodes A and D will be restored on the protection fiber (A-B-C-D). In order to restore the traffic, all the nodes A, B, C, and D, and all the links AB, BC, CD will be involved in re-routing the traffic. The third problem of ring topology is the delay problem. Because of the nature of the ring, when there are a large number of nodes in the ring, a node is difficult to reach through all other nodes with a short delay.

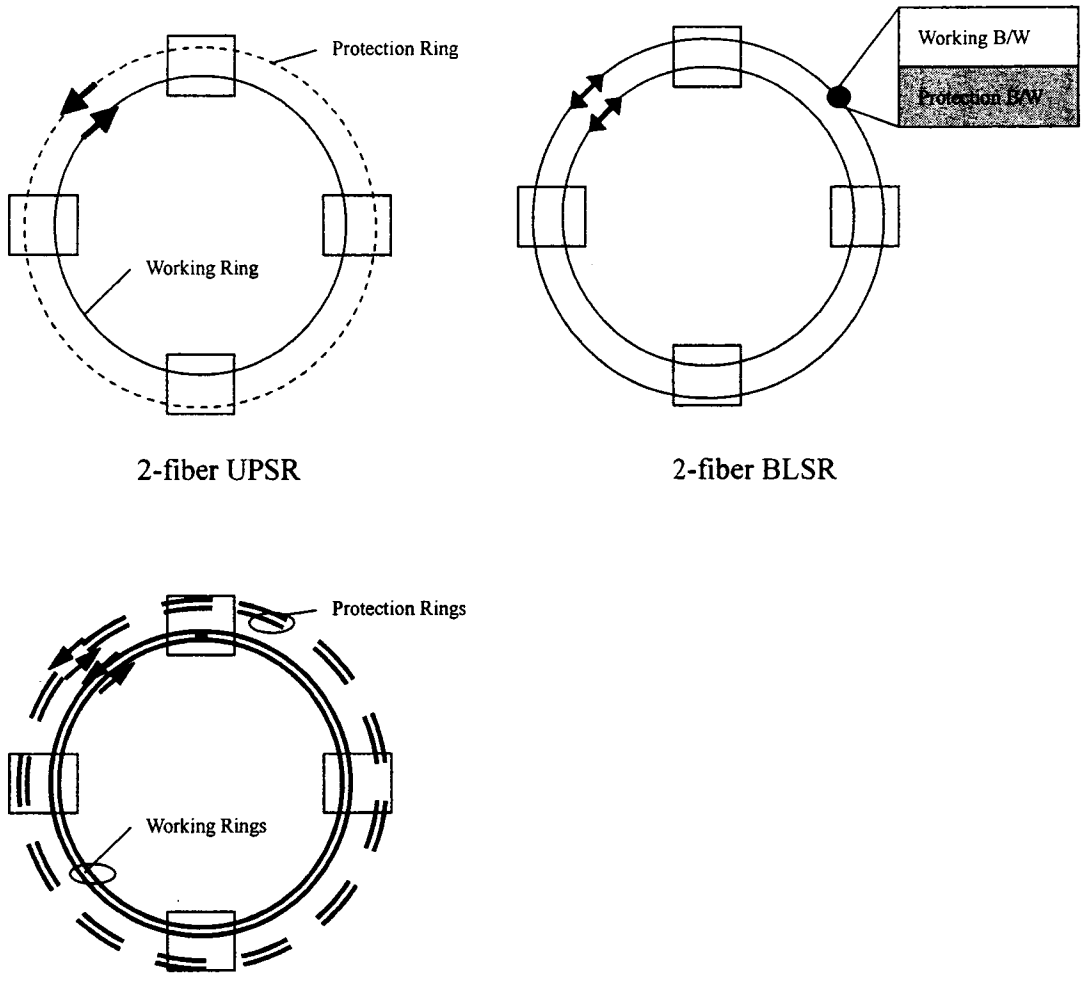


Figure 1.1 Three Classical Rings

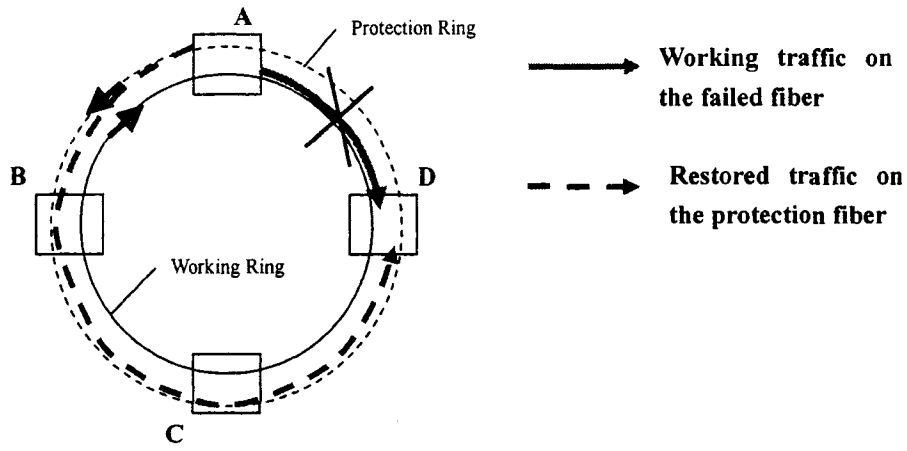


Figure 1.2 Fiber Broken in a Ring

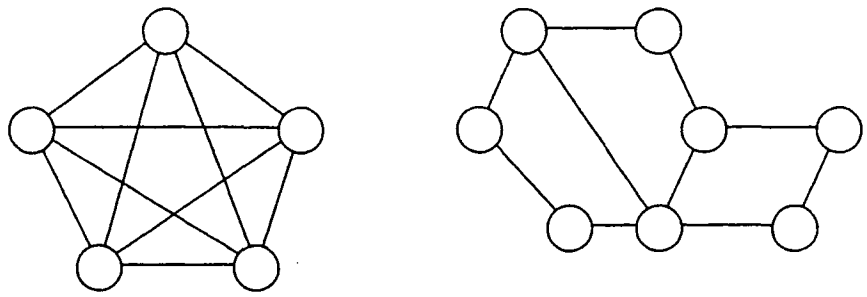


Figure 1.3 Mesh Networks

1.1.2. Mesh Networks

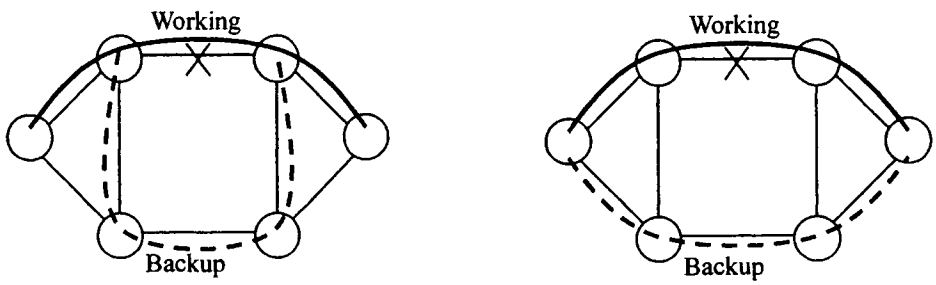
Mesh network is a communications network in which each node has at least two links to other nodes. The improvements in optical switching and routing technology have made the mesh topology more useful than the ring topology in backbone transmission networks [5]. A mesh topology can provide several advantages to overcome most of the ring problems, such as flexibility to network adjustment and robustness to failure. Furthermore, delays can be reduced easily by adding links in the

network [5]. Unlike those in a ring network, in a mesh network a node may have more than two links to interconnect with other nodes. A mesh network can employ one of the two connection arrangements, full mesh topology and partial mesh topology, as shown in Figure 1.3. In the full mesh topology, each node is connected directly to all the other nodes. In the partial mesh topology, each node is connected to a subset of nodes (but not all) in the network. These nodes are typically the ones that the node exchanges the most data. Backbone transmission networks mainly use the partial mesh topology, because of their large number of nodes.

1.1.3. Protection and Restoration in Mesh Networks

Mesh networks have two major types of survivability mechanisms: protection and restoration. Protection mechanisms are proactive, in which backup paths have been established and backup capacities have been reserved in advance. Restoration mechanisms can be totally-reactive or semi-proactive. In totally-reactive restoration mechanisms, backup paths are identified and backup capacities are allocated after the failure occurs. In semi-proactive restoration mechanisms, backup paths are computed and signaled before failure, but backup capacities are allocated only after the failure occurs. Proactive protection is typically inefficient because the demands (connections) do not share the backup capacities of the network. On the other hand, totally-reactive restoration schemes cannot always prevent single link failure, because it is not guaranteed to find a backup path for the failed demand in the network. Due to these reasons, we mainly focus on the semi-proactive restoration mechanisms in this thesis.

There are two protection and restoration paradigms in mesh networks: (1) link protection/restoration and (2) path protection/restoration. In link protection/restoration, traffic restoration is handled by the two end-nodes of the failed link. The traffic along the failed link will be rerouted to a path between these two nodes [illustrated in Figure 1.4 (a)]. In path protection/restoration, traffic restoration is handled by the source and destination nodes of the connections, which are traversing the failed link. For each connection, an end-to-end backup path is used to restore the traffic. The backup path is link-disjointed with the working path of the connection [illustrated in Figure 1.4 (b)]. In this sense, link protection/restoration mechanisms are considered "local", whereas path protection/restoration mechanisms are called "end-to-end".



(a) Link (b) Path

Figure 1.4 Link and Path Protection/Restoration

1.1.4. Dedicated Path Protection and Shared Path Restoration

There are two approaches to backup capacity allocation in path protection/restoration in mesh networks: (1) dedicated protection and (2) shared restoration. In dedicated protection, a pair of link-disjoint working and backup paths is established from the source node to the destination for every protected demand (or connection). The capacities on links along the backup path are exclusively reserved for protecting the working path. An example shown in Figure 1.5 (a) illustrates the working paths and their corresponding backup paths (dashed arcs) for two demands A-E and A-F. The backup path of demand A-E traverses links AB and BE, whereas the backup path of demand A-F traverses links AB, BE, and EF. One unit of the bandwidth is exclusively reserved for each demand on links along their backup paths. And the total reserved capacities on links AB and BE are two units.

Shared mesh restoration refers to a class of mesh restoration techniques in which the spare (backup) capacity is shared among different connections. The primary paths of these connections must be failure disjoint, so that no single failure can put out of service more than one connection at one time. Previous research studies have shown that shared mesh restoration is the most promising technique for saving the spare capacity while still achieving full restoration for any single network component failure, such as a link failure [2].

In shared restoration, a pair of working and backup paths is computed from the source node to the destination for a demand. However, the reserved backup capacity on links along the backup path is not exclusively allocated to the demand. More than

one demand can share the backup capacity on the links along their backup paths; as long as their working paths are failure disjoint. Figure 1.5 (b) illustrates the working paths and their corresponding backup paths (dashed arcs) for demands A-E and A-F. Both backup paths traverse links AB and BE. The working paths of these demands are link-disjoint, and, thus the backup paths share reserved capacities in their mutual links. So, the capacity reserved on each link AB and BE is only one unit, which is one unit less than the reserved capacity on the same links when dedicated protection is used. If a link along one of the working paths fails, the reserved bandwidth on the mutual links is allotted for restoring the failed demand. However, if both of the working paths have links failed, the reserved bandwidth is allotted on a first-come-first-serve basis. So, one demand is restored, and the other is blocked.

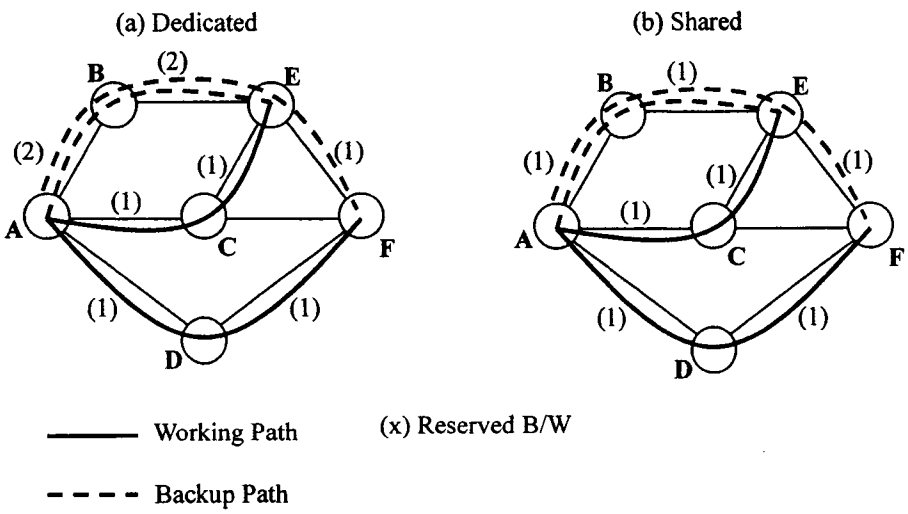


Figure 1.5 Dedicated and Shared Protection/Restoration

1.2. QoS Routing

While survivability mechanisms focus on the network capability to survive from events of physical failures; Quality of Service (QoS) mechanisms focus on network capabilities that provide the facilities to differentiate network traffic and offer different levels of services to each class of traffic.

The main goal of a routing algorithm is to find a feasible path (a path with enough bandwidth) that achieves efficient resource utilization. In addition, routes selected by using QoS routing algorithms must have sufficient resources for the QoS requirements [6]. There are many types of QoS requirements such as delay performance, data rate performance, synchronization, cost, load balancing, and so on.

Backbone service providers are always concerned with avoiding service interruption (due to network faults), providing bounded delay service to increasingly popular real-time applications and networking systems, and with overall network capacity distribution and load balancing. [6]. Thus, we choose the following two QoS requirements in this thesis: 1) minimizing the end-to-end packet delay in shared mesh restoration networks, and 2) evenly distributing the network traffic (load) amongst network links, by the process of routing.

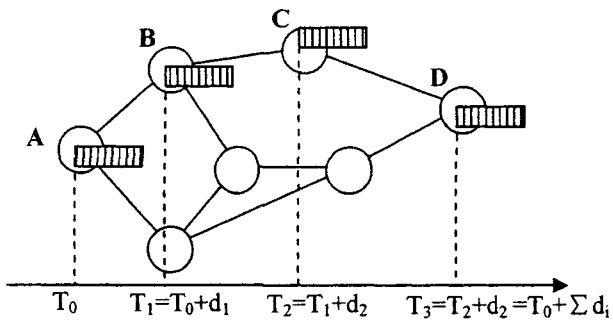


Figure 1.6 End-to-end Packet Delay

1.2.1. End-to-End Packet Delay

One of the objectives of this thesis is to improve the end-to-end packet delay performance in shared mesh restoration networks. Other QoS related parameters (such as packet loss, jitter) will be investigated in the future. End-to-end packet delay is defined as the duration for a packet to be transferred from the source to the destination. In Figure 1.6, a packet from A to D passes through 3 links (AB, BC, and CD) in the path. The end-to-end packet delay is equal to the sum of the individual delay experienced on each link (Σd_i: where d_i is the delay of link i).

1.2.2. Load Balancing

In addition to minimizing the end-to-end packet delay, it should be emphasized that achieving a balanced traffic load is of fundamental importance in communications networks, because ensuring an even workload distribution helps to eliminate congestions on network links [16]. So the second objective of this thesis is to improve the load balancing performance. In this thesis, load balancing refers to the process of distributing the network traffic evenly amongst network links so that no single link is overwhelmed.

An example is given in Figure 1.7 to describe the advantage of routing with load balancing. In Figure 1.7, each link has 3 units of capacity. There are three demands in the network: A-C, A-E, and A-F. In scenario (a), a routing algorithm without load balancing is used. The three paths for the three demands are A-C, A-C-E, and A-C-F respectively. Three units of capacity on link A-C is reserved, so link A-C is overwhelmed. In scenario (b), by using a routing algorithm with load balancing mechanism, the three paths are computed as A-C, A-B-E, and A-D-E. There is no link overwhelmed in the network.

In addition to avoiding link overload, load balancing routing may also help to save capacity in the network. Considering the example in Figure 1.7, suppose that a new demand A-C arrives. In scenario (a), the path found for the new demand (A-C) is A-B-E-C (dashed arc) which costs 3 units of bandwidth. However, in scenario (b), the path is A-C (dashed arc) which only costs 1 unit of bandwidth.

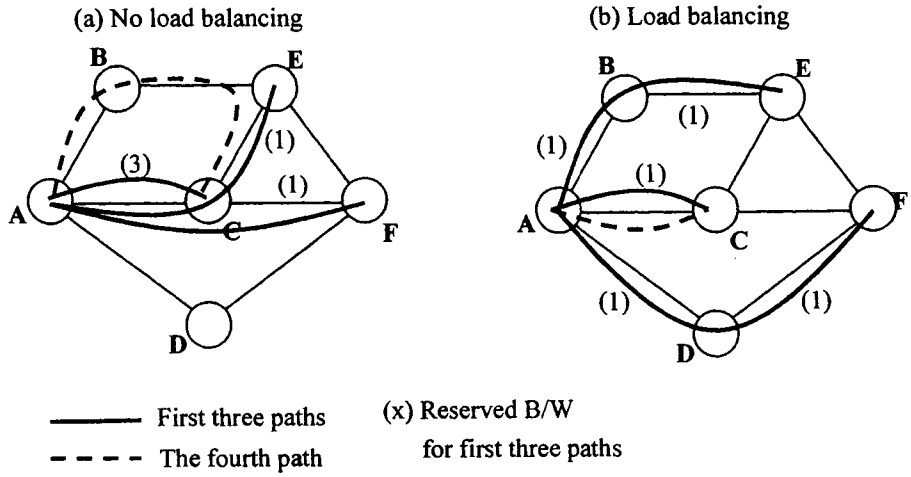


Figure 1.7 Load Balancing

1.3. Integrated QoS and Restoration Algorithms

Shared mesh restoration attempts to minimize the spare capacity that is used to fully recover connections from any single link failure in the network. On the other hand, QoS routing attempts to meet the QoS requirements of the applications and to improve the network performance. QoS mechanisms have been mainly applied at layer 2, e.g. *Asynchronous Transfer Mode* (ATM) and *Multi Protocol Label Switching* (MPLS), or layer 3 (IP) where packet forwarding and switching take place. In ATM networks, *Private Network-to-Network Interface* (PNNI) routing protocol is used to dynamically establish, maintain and clear ATM connections between two nodes. Because PNNI is based on the QoS routing algorithms, it can address needs of applications with real-time QoS requirements, such as guaranteed bandwidth and bounded delay [7]-[10]. In MPLS networks, *Differentiated Services* (DiffServ) architecture is used to provide QoS routing [11]-[13]. In IP networks, by using the resource *ReSerVation Protocol* (RSVP), a demand can request routers to provide a path which satisfies its QoS requirements [11][14][15].

Recent technological breakthroughs can now facilitate novel forwarding techniques for optical data bursts that make it possible to capture data bursts (or packets) at the optical layer. A major challenge in the transfer of the ultrahigh-speed data bursts is to allocate resources according to QoS specifications during the lifetime of data bursts (flows or connections) and to provide spare capacity required to address link failures in order to route these data bursts according to survivability requirements.

Consequently, development of novel integrated strategies that facilitate implementation of QoS and survivability algorithms is of significant practical interest and is the primary focus of this work. Reducing the end-to-end packet delay and improving the load balancing performance in shared mesh restoration networks are the two main objectives of this thesis.

1.3.1. Integrated Delay-Constrained and Restoration

This thesis is mainly concerned with minimizing end-to-end packet delay as one of the QoS requirements in shared mesh restoration networks. In Chapter 4, we first propose a novel integrated approach for packet delay and resiliency, referred to as Two-Step Delay-Constrained Pool Sharing (TDPS). The TDPS scheme aims to achieve the following two goals simultaneously: (a) reducing the total end-to-end packet delay along the working and backup paths, and (b) minimizing the total reserved working and backup capacities in the network.

The TDPS algorithm belongs to a class of mesh restoration algorithms known as two-step heuristic routing algorithms. With these algorithms, the working and backup paths for every demand are computed independently in two steps. In step 1, the working path is computed, whereas in step 2 the selected working path is used to compute a link-disjoint backup path. We will compare the TDPS algorithm with a classical two-step routing algorithm called Simple Pool Sharing (SPS) algorithm [20][22][29] which is reviewed in Chapter 3. The structure of both algorithms is based on a spare capacity pool sharing approach where the spare capacity in each link is

placed into a common pool. Connections will share the resource pool in each backup link as long as their primary paths are failure disjoint. The main difference between the TDPS and SPS algorithms is that the TDPS scheme is intended to achieve goals (a) and (b) listed above, whereas the SPS scheme only achieves goal (b).

The second new integrated delay-constrained shared mesh restoration algorithm introduced in this thesis is called Hybrid Pool Sharing (HPS). Like the TDPS scheme, the HPS scheme takes into account the delay constraint when the working path is computed. However, this constraint is relaxed (not considered) when the backup path is computed, in order to maximize the reusability (sharing) of the backup bandwidth. Therefore, in terms of the capacity performance, we will show that the HPS scheme performs in the middle among these three algorithms (SPS, TDPS, and HPS). In terms of the delay along the working paths, it performs similarly to the TDPS scheme. In terms of the delay along the backup paths, it performs similarly the SPS scheme.

1.3.2. Integrated Load Balancing and Restoration

In recent years, a plethora of shared mesh restoration algorithms have been proposed to provide low capacity cost and readily available restoration in the transmission networks [17][18]. However, one problem that has not been adequately addressed, but which is considered to be a contributory factor by various researchers is that when the sole objective is to maximize backup capacity sharing, some backup links may be shared by many primary paths while other links may not. This uneven distribution of load contradicts the principle of load balancing whose ultimate goal is to distribute the network load evenly amongst network links for congestion control.

Therefore, the major challenge here is to allocate capacity according to load balancing specifications and to provide spare capacity required to address link failures according to survivability requirements. Consequently, development of novel integrated strategies that facilitate load balancing and survivability algorithms is of significant practical interest and is one of the two objectives of this thesis.

In Chapter 5, we present a novel integrated algorithm for load balancing and shared restoration, called Load Balancing Pool Sharing (LBPS). The LBPS scheme is a two-step heuristic routing algorithm that aims to achieve the following two goals simultaneously: (a) minimizing the total reserved capacity in the network by allowing backup paths for multiple connections (demands) to share common spare capacities on backup links, and (b) distributing the network traffic evenly amongst network links. We will compare the performance of the LBPS algorithm with the Simple Pool Sharing (SPS) algorithm.

1.4. Trap-topology

Solving a major drawback of all two-step shared restoration algorithms is one of the contributions of this thesis. The drawback of all two-step routing algorithms (such as LBPS, TDPS, HPS, and SPS) is that these algorithms suffer from a problem known as trap-topology, where the algorithm cannot find a pair of link-disjoint paths between two nodes even though diverse paths between these nodes actually exist on the topology [31][32]. For example, in the network topology shown in Figure 1.8, a two-step scheme can potentially compute the working path A-B-C-Z for a newly

arrived demand between nodes A and Z. This pre-selected working path will not have a diverse backup path even though two diverse paths A-D-E-C-Z and A-B-F-G-Z exist between nodes A and Z in the network.

To solve the trap-topology problem of the TDPS algorithm, we will introduce a one-step algorithm, called One-Step Delay-Constrained Pool Sharing (ODPS) which is based on Delay-Constrained Suurballe's Algorithm (DSA) presented in [1] for generating the shortest pair of link-disjoint paths between a given pair of nodes. The DSA scheme is a dedicated mesh restoration scheme where the bandwidth on each link along the backup path is exclusively reserved for each demand. Since spare capacities are not shared among working paths, this scheme obviously consumes more backup bandwidth than the shared mesh restoration schemes described in this thesis.

Both the DSA and ODPS schemes proposed in this thesis incorporate a one-step path computation process (where the working and backup paths are computed in one step) in order to avoid the trap-topology problem. However, the ODPS scheme consumes less backup capacity than the DSA scheme, because it computes backup paths with shared backup mechanism.

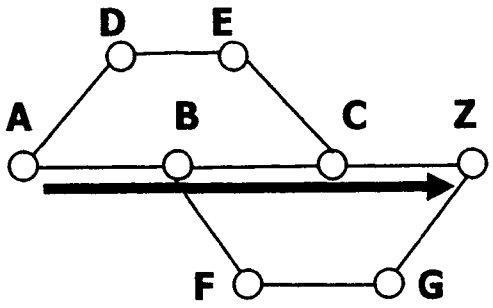


Figure 1.8 Example of trap-topology

In this thesis, we also introduce another technique for eliminating trap topology. This technique is referred to as "iterative simultaneous diverse-paths computation", which identifies the links causing trap topology, and avoids using them during the working path computation, in each iteration. This technique is then incorporated into the SPS and LBPS algorithms to produce two new trap-topology-free shared mesh restoration algorithms. These new algorithms are referred to as: Iterative Simple Pool Sharing (ISPS) and Iterative Load Balancing Pool Sharing (ILBPS).

1.5. Performance Evaluations

Using simulations, which are written in C++, we study the proposed TDPS, HPS, and ODPS algorithms in the existing North-American transport networks (these are: National Science Foundation network (NSF), Global Crossing network (GCN), and MCI network) and compare their end-to-end delay and capacity usage performances with the SPS algorithm. We will show that the TDPS and ODPS schemes achieve much lower end-to-end delay performance than the SPS schemes, at the cost of a minor increase in the reserved capacity in the network. We will also show that the HPS algorithm outperforms the SPS algorithm in terms of the delay performance and outperforms the TDPS and ODPS algorithms in terms of capacity performance.

We also study the proposed LBPS, ISPS, and ILBPS algorithms in the existing North-American transport networks and compare their capacity, load-balancing, and computation complexity performances with the SPS algorithm as well as with two other representative shared mesh restoration algorithms. These are: Routing with Load

Balancing Heuristics (RLBH) which is a load-balancing shared mesh restoration algorithm [33]; and Iterative Two-Step Approach (ITSA) which has been reported as an "optimal" algorithm in the literature [23]. We will show that while the LBPS and ILBPS schemes achieve the optimal capacity performance of the SPS and ISPS schemes, they do however yield a more even workload distribution amongst network links than the SPS and ISPS schemes, which ultimately leads to lower congestion. We will also show that the LBPS and ILBPS algorithms outperform the RLBH algorithm in terms of the capacity and load-balancing performances. Finally we compare the LBPS and ILBPS schemes with the ITSA scheme, and show that they achieve the same capacity performance as the ITSA scheme at much lower computational cost.

1.6. Outline

The rest of the thesis is structured as follows. Chapter 2 provides a survey of related research on shared mesh restoration schemes and QoS routing techniques. Chapter 3 briefly reviews RLBH, ITSA, SPS, and DSA algorithms, used as benchmarks to evaluate the algorithms proposed in this thesis. Chapter 4 introduces the TDPS, HPS, and ODPS algorithms, and then presents the simulation results by implementing the three shared restoration algorithms in representative North American backbone networks. Chapter 5 introduces the LBPS, ISPS and ILBPS algorithms, and then presents the simulation results by implementing these three algorithms in representative North American backbone networks. The last section is the conclusion of this thesis.

Chapter 2 RELATED RESEARCH

2.1. Linear Programming

Linear programming studies optimization problems with a linear objective function, subject to linear equality and inequality constraints [35]. Many shared mesh restoration algorithms have been developed to address optimal path computation problem. Some of these studies are based on linear programming that computes link-disjoint paths for all demands simultaneously [24][25][34].

Minimizing total working and backup capacity usage is normally used as the objective function in these studies. The link-disjoint and shared spare capacity mechanisms are employed as constraint functions [24][25][36].

In reference [37], some Linear Programming QoS routing algorithms are introduced; in which each network link has two integer weights, cost and delay. The algorithms compute a minimum cost path from a source node to a destination node such that the delay of the path is bounded by a specified integer value. They have done the efficiency comparison between all the algorithms. An algorithm based on the dual of the Linear Programming Relaxation was shown to be the most efficient.

Reference [38] presents a general frame work for routing of QoS flows in multi-service optical networks. The capability of surviving against single or multiple node and/or link failure(s) have also been considered in their work; and a QoS routing method based on Linear Programming is provided to compute the primary and backup paths which satisfy the QoS requirements.

Reference [39] presents a Linear Programming algorithm whose objective is to

minimize the overall network capacity needed to carry and protect the traffic. A load balancing function is used as a constrained function in their Linear Programming algorithm. In addition, the complexity of the Linear Programming is investigated both theoretically and empirically.

The Linear Programming based schemes have two primary advantages over the two-step heuristic restoration schemes: they yield more optimal results, and they avoid trap-topology. However, their primary disadvantages are their complexities and the need for the whole network's demand matrix, which preclude their use in dynamic network environment with varying traffic demand.

2.2. Existing Protection and Restoration Algorithms

Many studies have explored the capacity saving of various shared mesh restoration schemes over the alternative protection and restoration schemes. Less studies have however been carried out on the evaluation of the Quality of Service (QoS) performance of the shared mesh restoration schemes.

Some notable examples of existing shared mesh restoration algorithms without QoS mechanism are given below, besides the SPS algorithm which will be reviewed in more detail in the next chapter. In [25], a formulation of a linear programming model for computing least-cost link-disjoint paths for all demands in shared path restoration networks is given. Reference [26] introduces a heuristic shared restoration algorithm modified from the OSPF algorithm to find a pair of risk-disjoint paths. Reference [28] presents an iterative network-flow heuristic algorithm which has been

reported to be optimal in terms of capacity usage. However, it is computationally complex and therefore not suitable for dynamic (real-time) path computation where a large set of alternate paths must be examined between a given source-destination nodes.

In all of the papers referenced above, "blocking probability" and "capacity usage" have been used as the criteria for path selection. None of these schemes have taken QoS related parameters into consideration. In the following, some examples of restoration algorithms which incorporated QoS mechanisms are given.

One example is a restoration scheme proposed in [30] which attempts to quantify QoS in *Wavelength Division Multiplexing* (WDM) shared mesh restoration networks. However, in their paper, QoS represented the amount of time that a connection is down, rather than the packet-level QoS parameters (such as delay, jitter, or loss).

Reference [32] gives a tutorial on QoS routing and reviews techniques and algorithms introduced in the literature for satisfying QoS requirements for every admitted connection. One such algorithm is presented in [45] which proposes a heuristic algorithm for an NP-complete delay-constrained least-cost routing problem. The goal of this algorithm is to find a path that has the highest probability to satisfy a given end-to-end packet delay bound. Another QoS routing algorithm is the one in [46] which studies a bandwidth-constrained and delay-constrained routing problem with imprecise network states. Yet, another algorithm is the one presented in [47] which finds a bandwidth-delay-constrained path by Dijkstra shortest path algorithm. The above three algorithms are only the most notable examples of many other QoS routing

algorithms developed in recent years. However, almost none of these algorithms have taken protection and restoration parameters into consideration.

2.3. Existing Load Balancing Algorithms

Setting the QoS requirement aside, however, there is the further problem of load balancing in shared mesh restoration networks. Some representative examples of algorithms for load balancing are those presented in [40]-[42]. These algorithms can be used to compute a single path between any given pair of nodes in the network that will traverse through links with the least amount of traffic load. These algorithms cannot however be used in mesh restoration networks where a pair of link-disjoint paths must be computed between any two nodes. In [43], a load balancing routing algorithm has been introduced which computes a pair of link-disjoint paths between a given pair of nodes. However, the algorithm does not incorporate backup bandwidth sharing, and therefore is suitable for dedicated mesh restoration only.

A few algorithms have been developed that incorporate explicitly the load balancing function in their mesh restoration formulations. The RLBH algorithm is one such algorithm that incorporates both functions into the algorithm [33]. This algorithm is reviewed in the next chapter.

Chapter 3 BENCHMARK MESH RESTORATION ALGORITHMS

In this chapter, we review five algorithms and use them as the benchmarks for evaluating the proposed algorithms in this thesis.

3.1. Simple Pool Sharing (SPS)

The SPS algorithm aims to compute a pair of link-disjoint working and backup paths between a given source and destination nodes [20] [22] [29]. The objective is to minimize the working and the backup capacities required for complete recovery of traffic from any single link failure. This is done by placing the spare capacity in each link into a common pool. Different demands can share the backup bandwidth in the pool if their working paths are link-disjoint. This ensures that failed connections from any single link failure in the network can be fully restored on the backup link.

The SPS scheme computes the pair of paths in two steps. In step 1, the working path is computed, whereas in step 2 a link-disjoint backup path is computed. Following the notations and definitions in Table I on page viii of this thesis, let us denote b_r the amount of bandwidth requested by a newly arrived demand r, A_i the available bandwidth on link i, and $C_W(i)$ the cost of link i for the purpose of working path computation. $C_W(i)$ is defined as of the following:

$$C_W(i) = \begin{cases} \infty & b_r > A_i \\ 1 & b_r \le A_i \end{cases} \qquad (2.1)$$

$C_W(i)$ is set to 1 if link i has enough available capacity for demand r, otherwise, it is set to infinity. A least-cost algorithm (e.g. Dijkstra's algorithm) is used with the above cost assignment in order to compute the least-capacity working path for demand r.

In step 2 of the SPS algorithm, a link-disjoint shared backup path is computed by using the backup bandwidth pool sharing technique described below. The backup bandwidth reserved on each link is recorded in the following matrix [20]:

$$\Omega = \begin{bmatrix} 0 & k_{12} & k_{13} & \dots & k_{1J} \\ k_{21} & 0 & k_{23} & \dots & k_{2J} \\ k_{31} & k_{32} & 0 & \dots & k_{3J} \\ \dots & & & & \\ k_{J1} & k_{J2} & k_{J3} & \dots & 0 \end{bmatrix} \tag{2.2}$$

Element k_{ij} is the amount of backup bandwidth needed on link j if link i fails ($1 \leq i, j \leq J$). J is the number of links in the network. In pool sharing, the total amount of backup bandwidth needed on link j (B_j) is the maximum of all elements in column j, which is:

$$B_j = \max_{\forall i \leq J}[k_{ij}] \tag{2.3}$$

We denote $S_W(r)$ as the set of links along the working path of demand r, T_j as the maximum amount of backup bandwidth required on link j if a link in $S_W(r)$ fails, and $C_B(j)$ as the cost of link j for backup path computation. T_j and $C_B(j)$ can be computed from the following formulas:

$$T_j = b_r + \max_{\forall i \in S_W(r)} [k_{ij}] \tag{2.4}$$

$$C_B(j) = \begin{cases} \infty & j \in S_W(r) \\ \varepsilon & T_j \le B_j \\ \dfrac{T_j - B_j}{b_r} & 0 < T_j - B_j \le A_j \\ \infty & \text{otherwise} \end{cases} \tag{2.5}$$

Condition 1 in (2.5) ensures that the backup path will be link-disjoint from the corresponding working path computed in step 1. With condition 2, the cost of link j is set to a very small number ε if on this link adequate shared backup bandwidth had already been reserved to restore demand r (i.e. $T_j \le B_j$). With condition 3, the cost of link j is set to $(T_j - B_j) / b_r$, where $(T_j - B_j)$ is the amount of additional backup capacity that must be reserved to restore demand r, and if this additional bandwidth is available (i.e. if $0 < T_j - B_j \le A_j$). If the additional bandwidth is not available, the cost of link j is set to infinity. The cost assignment (2.5) is used by Dijkstra's algorithm to compute a link-disjoint backup path for the working path computed in step 1.

Once the backup path is computed the total reserved shared backup bandwidth on links along the backup path must be updated. This is performed via updating the elements of matrix Ω: for every link i along the computed working path and every link j along the computed backup path, bandwidth b_r will be added to element k_{ij}.

$$\forall i \in S_W(r), \forall j \in S_B(r) : k_{ij} \leftarrow k_{ij} + b_r \tag{2.6}$$

$S_B(r)$ is the set of links along the backup path of demand r. After the elements in matrix Ω are updated, the new total shared backup bandwidth reserved on the backup links in $S_B(r)$ can be obtained from (2.3).

3.2. Routing with Load Balancing Heuristics (RLBH)

The RLBH algorithm is a two-step shared mesh restoration algorithm presented in [33]. The algorithm employs a threshold-based load balancing mechanism to select candidate links and to avoid using heavily loaded links during the working and backup path computations. For working path computation, if the amount of free bandwidth in a link is less than a threshold (called *critical index*), the link becomes critical and the cost of the link is set to a large number. For backup path computation, the cost of a link is set to a large number if the link is heavily loaded or if it does not have high chance of containing a sharable backup bandwidth to restore the new demand. To measure the latter quantity, the number of working paths that are already supported by the shared backup bandwidth on this link is compared against a threshold (called *venture index*). A backup link with supported ratio larger than the venture index is considered not able to support any more working paths; therefore such a link is not considered as a backup link candidate for the new demand.

The primary difference between the RLBH algorithm and our load balancing algorithms presented in this thesis is that we do not employ threshold-based mechanism to select candidate links. Because, network links can generally operate at different capacities, it is unclear to us how these thresholds must be specified so that the fair

usage of capacity on every link can be enforced.

3.3. Iterative Two-Step Approach (ITSA)

The ITSA algorithm [23] is an iterative algorithm, which implements an iterative rule to compute a pair of link-disjoint working and backup paths for a given demand between a given source and destination nodes. In each iteration, a two-step path computation approach is used to compute a new pair of working and backup paths. The sum of the costs of the two paths is used as the criterion to select the optimal pair of paths. When the final iteration is completed, the algorithm selects a pair of paths with the smallest accumulative required capacity among all iterations.

The ITSA algorithm executes in K iterations. It uses *Yen's algorithm* [49] to compute K shortest paths between the source and destination nodes of the given demand. Yen's algorithm is a classical algorithm for ranking the K shortest loopless paths between a pair of nodes in a network. In iteration k, the ITSA algorithm uses the kth shortest path as the potential working path, and it computes a link-disjoint backup path for the working path. The sum of the costs of the two paths is then compared with the sum of the costs of the two paths found in the previous iteration. If the new sum is less than the old, the new pair of paths is accepted and the old pair is discarded. At termination, the ITSA algorithm yields the pair of link-disjoint paths with the least total cost. Due to its large number of iterations, it has been shown that the ITSA scheme is optimal and can achieve a very low call-blocking performance [23].

3.4. Suurballe's Algorithm

Suurballe's algorithm [1] adopts a different strategy than that taken by either the LP-based or the two-step algorithms. With Suurballe's algorithm, the shortest pair of link-disjoint paths is computed for every demand in one step. The general configuration for the construction of these paths consists of a shortest path and a second path, with the difference, however, that the second path has overlaps with the first path under the following constraints: i) the cost of the links of the second path that interlacing with the first is made negative, and ii) the costs of all other links in the network are not changed, and remain the same as they were during the first path. Provided that these constraints are properly imposed, reference [1] has hypothesized that the shortest pair of link-disjoint paths is obtainable from a suitable shortest path algorithm (such as a Dijkstra algorithm) by applying it twice such that: a) the first run of the shortest path algorithm produces the shortest path from a source node to a destination node, and b) the second run produces a second path which can possibly have overlaps with the first. The link-disjoint shortest pair would then be obtained by erasing the interlacing links.

Although the Suurballe's algorithm avoids trap topology, it is not suitable for computing a working path and a link-disjoint shared backup path. The reason for this is that instead of using two different link cost functions (one for working path computation and one for shared backup path computation), the Suurballe's algorithm uses one link cost function to compute the first and second paths. Therefore, either of the two paths computed by the algorithm can be used as a working or as a backup

path; these two paths are essentially no different from each other in terms of bandwidth allocation. For this reason, the Suurballe's algorithm is suitable to be used in dedicated mesh restoration, where the bandwidth on backup links is exclusively reserved for each demand, rather than being shared with other demands.

3.5. Delay-Constrained Suurballe's algorithm (DSA)

The Delay-Constrained Suurballe's algorithm (DSA) reviewed in this thesis is an application of the Suurballe's algorithm. The DSA algorithm uses a link delay time to be the cost of a link. As a result, it computes a pair of working and backup paths with the lowest total delay.

Let d_i represent the average delay that the previously transmitted packets experienced on link i. In general, d_i can have three components: queuing delay, transmission delay, and propagation delay. In DSA, the working and backup paths will be found in three phases described below.

[Phase 1]: compute the lowest delay path using the Dijkstra algorithm. Let $S_1(r)$ be the set of links along this path (to be found). The DSA scheme uses the following cost function $C_1(i)$ for every link i in order to compute this path:

$$C_1(i) = \begin{cases} \infty & b_r > A_i \\ d_i & b_r \le A_i \end{cases} \tag{2.7}$$

[Phase 2]: compute the second lowest delay path from the source node to the destination node. Let $S_2(r)$ be the set of links along the second path to be found. The following cost function $C_2(j)$ is used for every link j in order to compute the second lowest delay path:

$$C_2(j) = \begin{cases} Z_j & j \in S_1(r) \\ d_j & b_r \le A_j \\ \infty & b_r > A_j \end{cases} \tag{2.8}$$

Where

$$Z_j = \begin{cases} \infty & \text{To\_Destination} \\ -d_j & \text{To\_Source} \end{cases} \tag{2.9}$$

All links are assumed to be bidirectional. In Phase 2, the DSA scheme replaces each bidirectional link in the set $S_1(r)$ by two unidirectional links (arcs); one directed toward the destination (To_Destination), and the other toward the source node (To_Source). The cost of the arc toward the destination is set to infinity ($Z_j = \infty$), whereas the cost of the arc toward the source is set to $-d_j$. An example is shown in Figure 3.1. The costs of all other links not in $S_1(r)$ are set to either d_j or ∞ depending on how much bandwidth is available on these links. Using the cost function $C_2(j)$, the Modified Dijkstra algorithm [1] is executed in order to find the path $S_2(r)$. The Modified Dijkstra algorithm is a simple modification to the standard Dijkstra algorithm that computes the shortest path in the network in the presence of links with negative costs [1].

[Phase 3]: during this phase, any interlacing links of the two paths $S_1(r)$ and $S_2(r)$ are found and erased. The remaining links are then re-grouped in order to obtain the pair of link-disjoint working and backup paths. The lower delay path will be selected as the working path and the higher delay path will be the backup path. Reference [1] has shown that the above interlacing and re-grouping process always yields a pair of link-disjoint paths.

From the cost functions (2.7) and (2.8), we can see that the DSA algorithm is a dedicated protection algorithm. It can potentially reserve (allocate) more backup capacity than the shared restoration schemes.

Bi-direction link *i*:
C(AB) = C(BA) = d_i

Negative link j:
C(EF) = ∞
C(FE) = - d_j

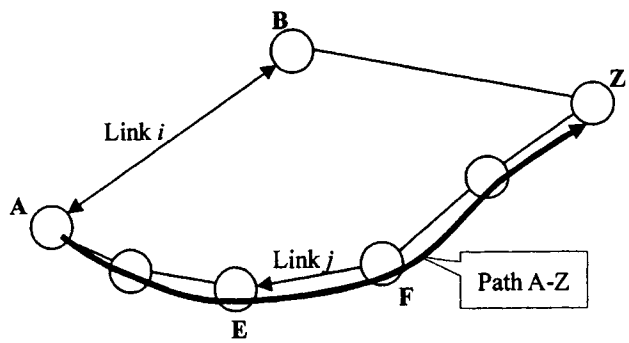


Figure 3.1 Negative Cost Link

Chapter 4 PROPOSED DELAY-CONSTRAINED RESTORATION ALGORITHMS

This chapter presents three novel restoration techniques aimed to minimize simultaneously: (1) the total end-to-end delay time along the primary and backup paths, and (2) the reserved capacity, in mesh transport networks. First, we present a Two-step Delay-Constrained Pool Sharing (TDPS) scheme and show how this scheme can be used to yield low end-to-end delay paths for demands in the network. We also introduce a hybrid two-step algorithm which relaxes the delay constraint on the backup path in favor of maximizing the reusability (sharing) of the backup bandwidth. We compare the performances of these two algorithms with the representative existing algorithm, referred to as Simple Pool Sharing (SPS) algorithm. Like any other two-step schemes, the TDPS, HPS, and SPS schemes suffer from a problem known as trap-topology. In order to eliminate this problem, we introduce a new heuristic algorithm, called One-Step Delay-Constrained Pool Sharing (ODPS). Using simulations, we study all of these proposed algorithms on the existing North-American transport networks, and compare their total end-to-end delay and capacity usage performances with the representative algorithm.

4.1. Two-Step Delay-Constrained Pool Sharing (TDPS)

The TDPS scheme computes a pair of working and backup paths for a demand r in two steps. In step 1, the working path is computed, whereas in step 2 a link-disjoint

backup path is computed by using a pool sharing process which is the similar to the SPS scheme. However, the TDPS scheme uses different link cost assignments than those used by the SPS scheme to determine the paths.

In the first step of the algorithm, the TDPS algorithm computes a working path for demand r that (a) satisfies the capacity requirement of this demand and, (b) achieves a minimum end-to-end packet delay. Like before, let d_i represent the average delay that the previously transmitted packets experienced on link i. The cost of link i is then defined as the following:

$$C_W(i) = \begin{cases} \infty & b_r > A_i \\ d_i & b_r \le A_i \end{cases} \tag{3.1}$$

The cost of link i is set to d_i if the link has enough available capacity for demand r. Otherwise, the cost is set to infinity. The above cost assignment is used by Dijkstra algorithm to find a least-delay working path for demand r.

In the second step, the TDPS scheme computes a least-delay link-disjoint shared backup path for the working path. Demand r shares the backup bandwidth on links along the backup path with other demands. Following the notations used in the previous chapter, we denote $S_W(r)$ to be the set of links along the working path of demand r, and T_j to be the maximum amount of backup bandwidth required on link j if a link in S_W fails. T_j is computed according to (2.4). $C_B(j)$ is defined as the following:

$$C_B(j) = \begin{cases} \infty & j \in S_W(r) \\ d_j & T_j \le B_j \\ d_j + \delta & 0 < T_j - B_j \le A_j \\ \infty & \text{otherwise.} \end{cases} \tag{3.2}$$

The above cost assignment has two main differences from the cost assignment (2.5). It sets the cost of link j to d_j (rather than to ε) if on this link adequate shared backup bandwidth had already been reserved to restore demand r (i.e. $T_j \le B_j$). Therefore, if more than one backup links meet the above bandwidth condition, the one with the lowest delay is favored. The cost function (3.2) also differs from (2.5) in that it sets the cost of link j to $d_j + \delta$ (rather than to $(T_j - B_j) / b_r$) if this link satisfies the third condition. A link j meets this condition if $T_j - B_j$ units of additional backup bandwidth must be reserved on this link in order to restore demand r, and if this bandwidth can be honored (if $0 < T_j - B_j \le A_j$). Parameter δ is a nonnegative number which is added to allow the path computation algorithm to favor a link that meets condition 2 over a link that meets condition 3, if the delays on both links are equal. This ensures that the TDPS scheme achieves the least-delay backup path while still maximizing backup bandwidth sharing. In order not to affect the delay constraint, parameter δ should be set to a value much smaller than the value of the link cost (d_j). We recommend setting δ to less than 1% of d_{min}, where d_{min} is the minimum of d_j over all links j in the network.

4.2. Hybrid Pool Sharing (HPS)

The SPS scheme can be used to find the least-capacity shared restoration path for many internet applications that are not sensitive to network delay latency time. In contrast, the TDPS scheme can be used to find the least-delay shared restoration path for delay-sensitive internet applications. However, there may exist some internet applications (e.g. interactive applications such as Web call-center) that want to receive low delay service when the network is normal, but can tolerate increased delay when the network is experiencing failure. During this time, connectivity is all that these applications want from the network no matter how much delay they would experience. For these applications, the SPS scheme may not provide adequately low working delay, and the TDPS scheme may be costly because of its relatively high backup capacity usage. In this chapter, we introduce an alternative scheme, called Hybrid Pool Sharing (HPS), which can yield a low working path delay and at the same time can save some backup capacity.

The HPS scheme computes a pair of working and backup paths for a demand r in two steps. In the first step, HPS uses the TDPS scheme's working path cost function (3.1) to compute a low-delay working path. But in the second step, it uses the SPS scheme's backup path cost function (2.5) to compute a least-capacity shared backup path.

4.3. One-Step Delay-Constrained Pool Sharing (ODPS)

The One-step Delay-Constrained Pool Sharing (ODPS) scheme incorporates the pool sharing process into the DSA scheme. Like the DSA scheme, it avoids the trap topology, and yields a pair of link-disjoint paths with the lowest total end-to-end delay (this is shown later in this chapter). However, a main difference between the ODPS algorithm and the DSA algorithm is that ODPS is a shared restoration algorithm, whereas DSA is a dedicated protection algorithm.

Because DSA is a dedicated protection algorithm, the cost of links in phase 2 of the algorithm is determined using the same cost (d_j) as for the phase 1 of the algorithm, except for those links in $S_1(r)$ where their cost is set to Z_j. This can be observed by comparing equations (2.7) and (2.8). For this reason, in DSA, *negative cycles* (closed loops) do not exist, and the simple Modified Dijkstra algorithm is sufficient to find the second path $S_2(r)$. Indeed, reference [1] has shown that negative cycles never exist with the DSA scheme.

However, the ODPS algorithm is a pool sharing restoration algorithm. As a result, the cost of links in phase 2 of the algorithm (to be introduced shortly) will be determined using the pool sharing process, rather than the same cost as for the phase 1 of the algorithm. This pool sharing process may often cause the negative cycle problem as well as two new problems, hereafter referred to as *insufficient capacity links* and *dead-end*, during phase 2 of the path computation. These problems cannot be dealt with by using the classical Modified Dijkstra algorithm alone. To this end, a new algorithm called: the *Iterative Restoration Dijkstra* (IRD) is introduced in this

thesis, which is designed to eliminate these problems during phase 2 of the ODPS algorithm.

Before we describe the above three problems in further detail, let us first present the ODPS algorithm and its three phases. The link costs used during these phases and their differences with those of the DSA algorithm will help one to understand the nature of the above three problems, for which the IRD algorithm was designed. The pseudo-code of IRD is presented in Appendix A.

The ODPS scheme computes the working path and the shared restoration path for a demand r according to the following three phases:

[Phase 1]: this phase is the same as in the DSA scheme. The ODPS scheme computes the first lowest delay path denoted by $S_1(r)$ using the Dijkstra algorithm and the cost function (2.7) repeated for convenience here:

$$C_1(i) = \begin{cases} \infty & b_r > A_i \\ d_i & b_r \le A_i \end{cases} \tag{3.3}$$

[Phase 2]: the ODPS algorithm uses the IRD algorithm to compute the second lowest delay path denoted as $S_2(r)$. The IRD algorithm is different from the Modified Dijkstra algorithm. The IRD is designed to solve the negative cycles, insufficient capacity links, and dead-end problems. These problems will be described shortly.

The IRD algorithm uses the following link cost function to compute the path $S_2(r)$:

$$C_2(j) = \begin{cases} Z_j & j \in S_1(r) \\ d_j & T_j \le B_j \\ d_j + \delta & 0 < T_j - B_j \le A_j \\ \infty & \text{otherwise} \end{cases} \tag{3.4}$$

Where

$$Z_j = \begin{cases} \infty & \text{To\_Destination} \\ -d_j & \text{To\_Source} \end{cases}$$

$$T_j = b + \max_{\forall i \in S_1(r)} [k_{ij}]$$

[Phase 3]: The ODPS algorithm finds and erases any interlacing links of the two paths computed in phases 1 and 2. Then it re-groups the remaining links to obtain the lowest delay pair of link-disjoint working and backup paths. Next, it checks the available working capacity on every link of these two paths. Because of the way that the IRD algorithm operates, one of the following two scenarios can occur as the result of the above check. 1) If both paths pass the above check, the lower delay path is selected as the working path and the higher delay path is selected as the backup path. 2) If some links on one of the two paths can only meet the shared backup bandwidth requirement (but not the working bandwidth requirement), then that path is selected as the backup path and the other path will be the working path.

Negative cycle problem

As shown in (3.4), the ODPS algorithm changes the cost of links (arcs toward the

source node) in $S_1(r)$ to a negative value ($-d_j$) in order to find the second lowest delay path $S_2(r)$. The links with negative cost may generate negative cycles, because of which the Modified Dijkstra algorithm may not find the path $S_2(r)$ successfully. For example, in the network topology shown in Figure 4.1, the shortest pair of paths from A to Z must be found. In phase 2 of the algorithm, the link C-D takes the cost of –4 from C to D, and the cost of infinity from D to C (toward the destination). The shortest path from A to Z actually exists and it is A-B-C-D-E-Z. But there is a negative cycle C-D-E-C with the total cost of –1 in the network. If the Modified Dijkstra algorithm was to be used to compute the second path from A to Z, it would fail to find the path because of the closed-loop. This negative cycle never occurs in phase 2 of the DSA algorithm, because it is a dedicated restoration algorithm, where the link costs are the same in both phases (1 and 2) of the algorithm, except for those links in $S_1(r)$ where their costs are set according to (2.9) during phase 2.

However, the ODPS scheme is a shared restoration algorithm. This means that some links may not have enough working capacity for demand r, and therefore their costs are set to infinity in phase 1. But the same links may have enough shared backup bandwidth and their costs will be finite in phase 2 of the algorithm according to (3.4). Negative cycles can arise in situations like this where the costs of some links in phase 2 are smaller (and finite) than their costs (either infinity or a large finite value) during phase 1 of path computation. The IRD algorithm is designed in this thesis to detect and eliminate these negative cycles during phase 2 of the path computation.

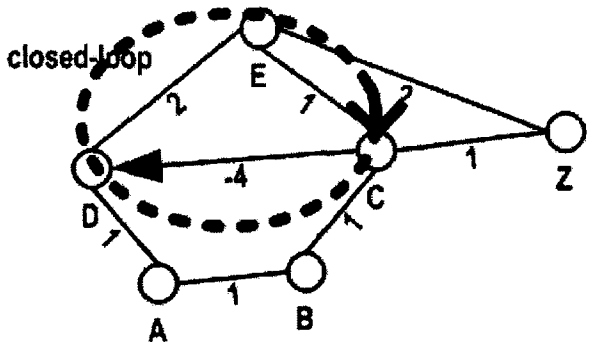


Figure 4.1 Example of negative cycle (closed-loop)

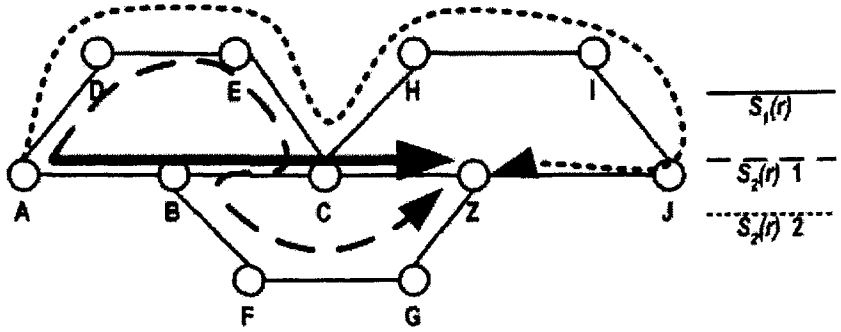


Figure 4.2 Example for IRD

Links with insufficient capacity

Like the DSA algorithm, the ODPS algorithm obtains the working and backup paths by erasing the interlacing links between $S_1(r)$ and $S_2(r)$ in phase 3. For example, in the network topology shown in Figure 4.2, using DSA or ODPS, the $S_1(r)$ and $S_2(r)$ paths might be A-B-C-Z and A-D-E-C-B-F-G-Z. In this example, link B-C is the only interlacing link between these two paths. After erasing this interlacing link, the working path (A-B-F-G-Z) and the backup path (A-D-E-C-Z) are obtained. Because of the erasing process, some links of $S_2(r)$ (links B-F, F-G, and G-Z in the example)

have been converted to as working links. Since DSA is a dedicated protection algorithm, it guarantees that these converted links to have enough working capacity to be used as working links. Indeed with DSA, all the links in $S_1(r)$ and $S_2(r)$ have enough working capacity to be used as working or backup links.

But the ODPS algorithm is a shared mesh restoration algorithm. When ODPS computes $S_2(r)$ using function (3.4), it only guarantees that there is enough capacity for backup path but not for the working path. Therefore if some links of $S_2(r)$ are converted (reselected) as working links, they may not have enough working capacity. The IRD algorithm is designed to avoid selecting these links with insufficient working capacity when $S_2(r)$ is computed.

Dead-end problem

In phase 2, the IRD algorithm starts from the source node and searches through a large set of route possibilities in a systematic way toward the destination node. During this search process, the IRD algorithm may add a link with a negative cost to the partial route found thus far toward the destination node. Because this link has a negative cost, it will be an interlacing link of the two paths $S_1(r)$ and $S_2(r)$. Therefore, the link in $S_2(r)$ that is emanating from the end-node of the above negative-cost link will likely be converted to a working link during phase 3 of the algorithm. Sometimes however the above emanating link does not have enough working capacity, because the cost function (3.4) does not check the working capacity requirement. We call this situation a dead-end, where a classical path computation algorithm (such as the

modified Dijkstra algorithm) would fail to find the path $S_2(r)$.

The IRD algorithm is indeed designed for dealing with the above dead-end problem. It uses a backtracking technique to abandon the negative-cost link described above, and to backtrack to the head-end of that link and from there retry routing toward the destination. For example, in Figure 4.2, when the link C-B with a negative cost is selected for $S_2(r)$, the following links (B-F, F-G, and G-Z) will be changed to working links. So these links must have enough capacity to meet the working path requirement. If they don't, the program fails to find the result. In situation like this, the IRD algorithm abandons link C-B from the partial path A-D-E-C found thus far toward the destination. It backtracks to node C where it selects the link C-H (which happens to have enough working capacity) and appends that link to the partial path A-D-E-C found thus far toward the destination.

4.4. Advantages of ODPS

Like many other two-step mesh restoration algorithms, the TDPS algorithm suffers from the trap-topology problem which arises with generally all two-step path computation algorithms, because with these algorithms, the working and backup paths are computed in two steps. The computed working path in step 1 may block all the possible link-disjoint backup paths. In the above section, we have presented that the ODPS algorithm can avoid trap-topology problem.

Another problem with the TDPS algorithm is that the achieved end-to-end delay for the working path can be significantly smaller than the achieved end-to-end delay

for the corresponding backup path. The reason for this is again due the fact that the algorithm computes the working and backup paths in two steps. In step 1, a working path with the lowest end-to-end delay time (among all eligible paths) is selected, and in step 2, a link-disjoint backup path with the second lowest end-to-end delay time is selected. In general, the difference between the first and the second lowest delay times can be large. This large gap in the delay times can present a problem for delay-sensitive demands, because while the smaller delay time along the working path may be well within the demand's delay bound, the larger delay time along the backup path may not. As a result the demand will be blocked (not accepted) due to unacceptable delay along the backup path. By using ODPS, the gap between working and backup delay times can be reduced, and then less demand will be blocked than using TDPS.

4.5. Simulation Results

We have used simulation technique to test the TDPS, HPS, and ODPS schemes and also to compare their performances with the SPS scheme. The simulation has been carried out to measure the performance metrics such as total end-to-end delay and capacity usage. The simulation program is written in C++.

In this section, we define the total end-to-end delay of a demand to be the sum of the link delay along its working and backup paths. We use the average total delay per demand (denoted by D) as a measure of delay performance, which is the total end-to-end delay along the working and backup paths of all demands averaged over

the number of accepted demands. In terms of the delay performance, a diverse paths computation scheme is said to outperform other candidate schemes if it achieves the smallest D.

We use the average reserved capacity per accepted demand (V) in order to evaluate the capacity performance of the mesh restoration schemes presented in this thesis. We compute V as the sum of $(W_k + B_k)$ over all values of k divided by n_d, where n_d is the number of accepted demands in the network.

We used three representative North American backbone networks (NSF, GCN and MCI) to test the criteria defined above. NSF is based on the National Science Foundation network, GCN is based on the Global Crossing network, and MCI is based on the MCI network. The NSF network has 16 nodes interconnected by 25 bidirectional links, GCN has 27 nodes interconnected by 38 links, and MCI has 38 nodes interconnected by 67 links, all in mesh form. The topologies of these networks are illustrated in Figure 4.3. Some important characteristics of the three networks are shown in Table 4.1.

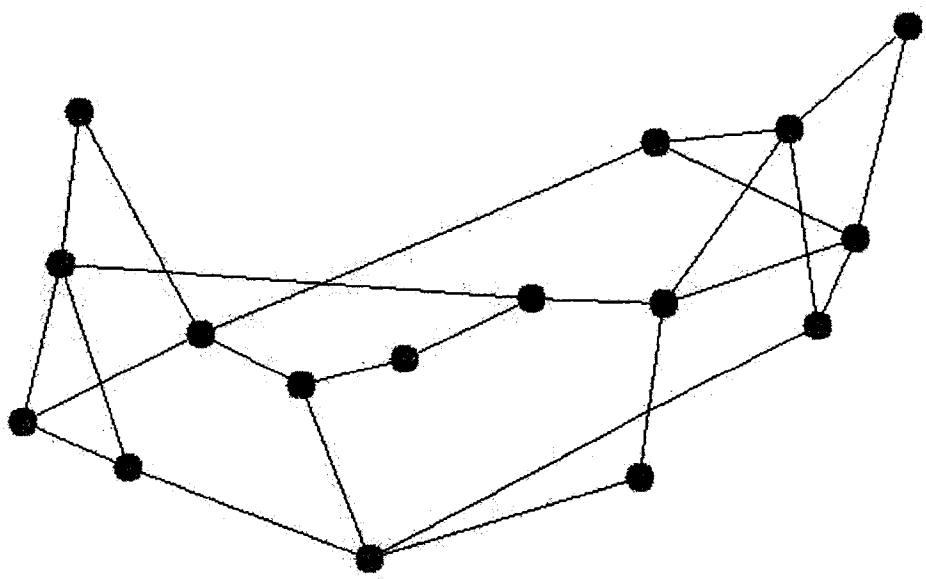
In this chapter, we assume that all links in the network have equal capacity of M_k = 2.4 Gbps which is commonly used in today's optical backbone networks. We generated five demand matrices for each network, with the total number of demands in each matrix was: 30 / 40 / 50 / 60 / 70 for NSF and GCN, and 40 / 50 / 60 / 70 / 80 for MCI. For every demand in these matrices, the source and destination nodes are generated randomly according to uniform distribution between 1 to M, where M is the number of nodes in the network. In our simulations, all demands requested identical

amount of bandwidth b_r = 100 Mbps from the network, which is a standard bandwidth in Ethernet. The link delay d_i (msec) was set to the propagation delay on link i, and parameter ε and δ were set to 0.01. The sensitivity of the TDPS and ODPS algorithms to the variations in parameter δ will be investigated later in this chapter.

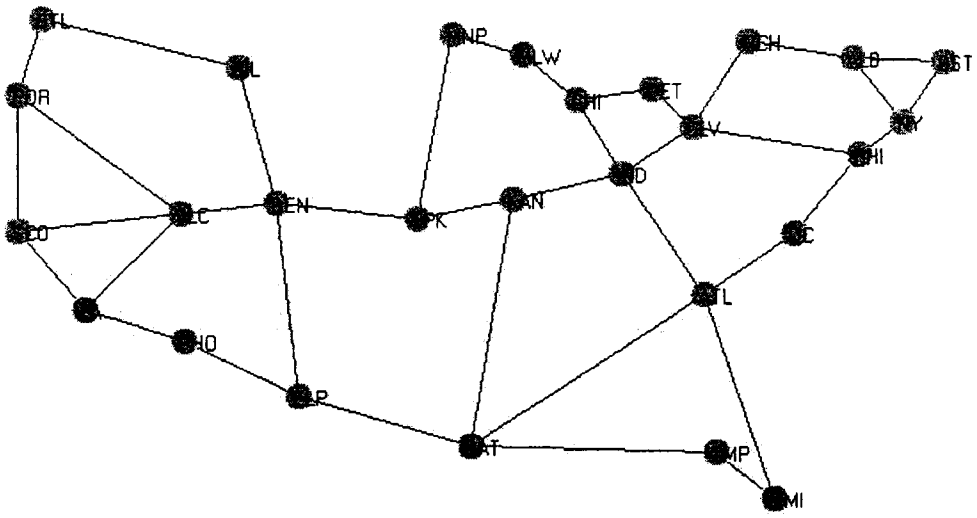
In order to achieve a reasonable confidence interval, we have repeated each simulation scenario 40 times and the final values for D and V metrics were obtained as the average over all 40 simulations.

TABLE 4.1. NETWORK CHARACTERISTICS

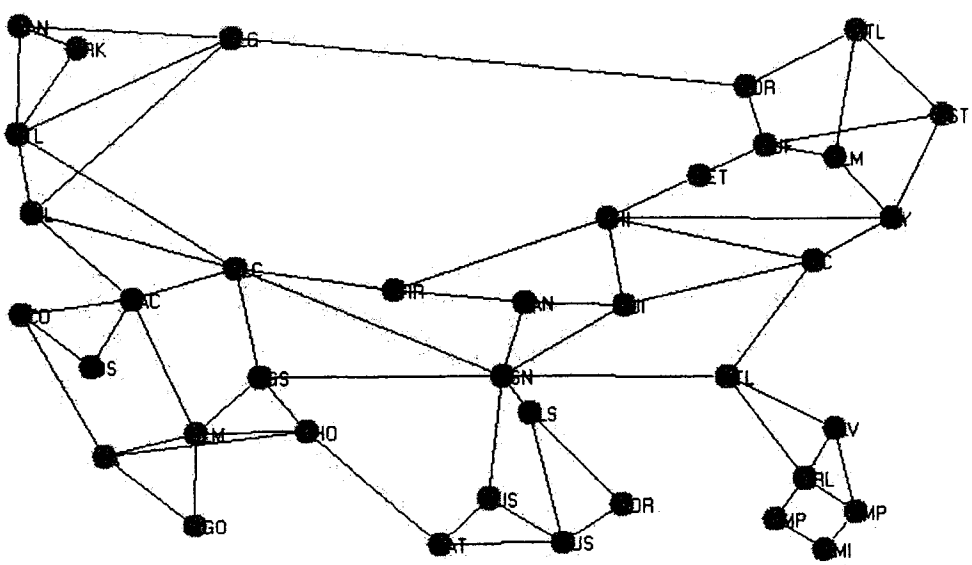
Network	No. of Nodes	No. of Links	Capacity	Min/Avg/Max Link Length (Km)	No. of Demands in Each Traffic Matrix
NSF	16	25	2.4G	600 / 1182 / 3000	30 / 40 / 50 / 60 / 70
GCN	27	38	2.4G	84 / 708 / 1609	30 / 40 / 50 / 60 / 70
MCI	38	67	2.4G	16 / 676 / 3427	40 / 50 / 60 / 70 / 80



NSF



GCN



MCI

Figure 4.3 Topology of the simulated networks

4.5.1. Delay Performance

As we said before, we use the total delay per demand (D) as a measure of delay performance in this thesis. The values of D for the three proposed restoration schemes (TDPS, HPS, ODPS), and the SPS scheme are shown in Figure 4.4 for NSF, Figure 4.5 for GCN, and Figure 4.6 for MCI. The data is shown as a function of the number of accepted demands in the network (n_d). The above figures show that the total end-to-end delay D for the TDPS and ODPS schemes are significantly smaller than the corresponding metric for the SPS and HPS schemes. The reason for this is that the TDPS and ODPS schemes concentrate on minimizing end-to-end delay. Consequently, the delay D with the TDPS and ODPS schemes is lower than the corresponding parameter for the SPS and HPS schemes.

We can also observe that the ODPS scheme outperforms TDPS slightly. This improved performance is yielded by complete elimination of trap topology problem and the reduction in the gap between the delay times on the working and backup paths for every demand. Finally, the HPS scheme takes the middle ground between the SPS scheme and the TDPS/ODPS schemes. The reason for this is that the HPS scheme uses the TDPS scheme's working path cost function (3.1) to compute a low-delay working path, but in the second step it uses the SPS scheme's backup path cost function (2.5) to compute a least-capacity shared backup path.

Thus, we can conclude this section as follows: (1) the delay-constrained schemes TDPS and ODPS yield less end-to-end delay results than the SPS and HPS schemes; (2) the HPS scheme outperforms SPS in terms of the delay performance, because it

considers delay constraint when finding the working path.

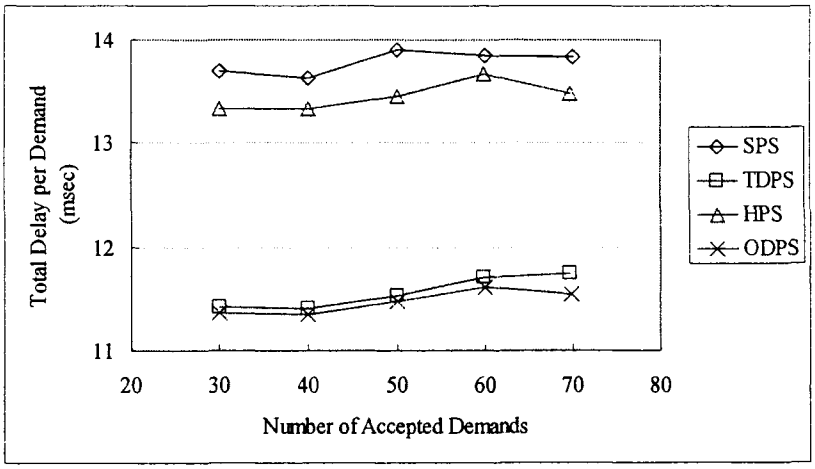


Figure 4.4 Average total delay per demand in NSF

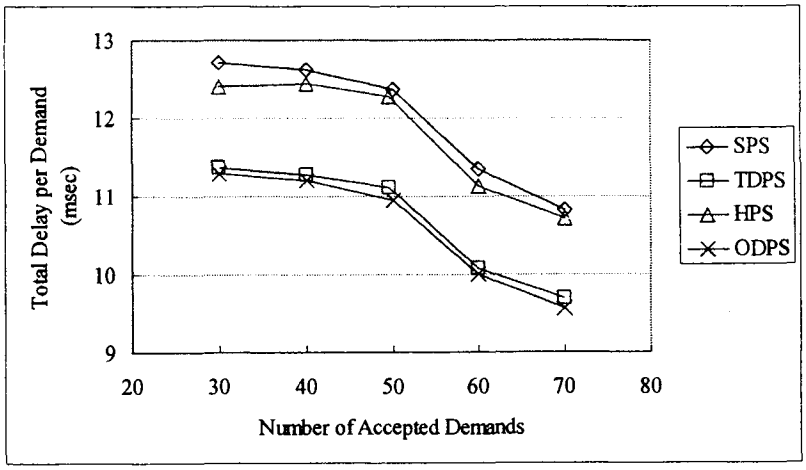


Figure 4.5 Average total delay per demand in GCN

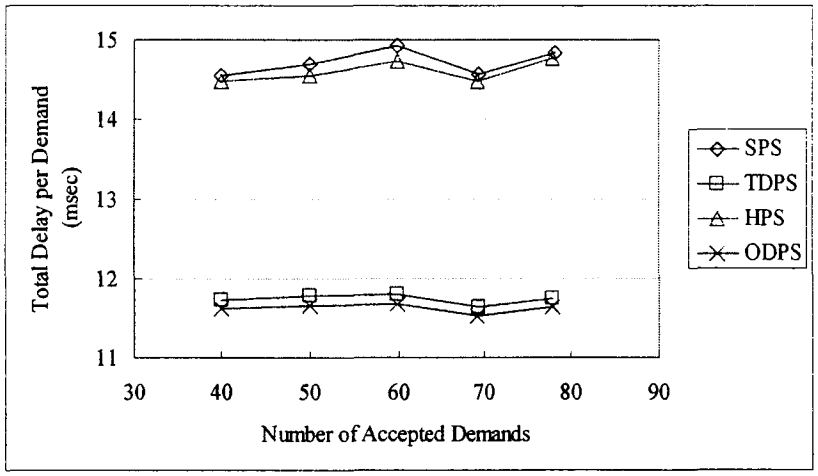


Figure 4.6 Average total delay per demand in MCI

4.5.2. Capacity Usage

The related average reserved capacity per accepted demand (V) of all the four restoration schemes in the three networks are shown in Figure 4.7 through Figure 4.9, each as a function of the number of accepted demands (n_d) in the network. The following important observations can be made from these figures: (1) in all of the three tested networks, the SPS scheme has the best performance in terms of the capacity usage. This is intuitive, because the SPS scheme attempts solely to minimize the capacity usage in the network. Whereas the other three schemes concentrate on minimizing end-to-end delay; the capacity usage is of their secondary importance. (2) The ODPS scheme performs slightly better than TDPS. (3) The HPS scheme always outperforms the TDPS and ODPS schemes, and underperforms the SPS scheme. This observation was particularly to be expected, because the working paths of the HPS scheme are less capacity optimized than the SPS scheme. The HPS scheme consumes less capacity than the TDPS and ODPS schemes, because it

maximizes backup bandwidth sharing among backup paths.

In Figures 4.7 through 4.9, we can see that the average reserved capacity per demand decreases as the number of accepted demands increases. Similar observation can also be made when we consider the delay metric in Figures 4.4 and 4.5. This happens because as the number of accepted demands increases the network load increases. At high network loads, long connections (demands) and connections with large bandwidth are more likely to be rejected (when they arrive) than the short connections or connections with small bandwidth. This is because the former connections will be subject to more capacity constraints at intermediate links than the later connections. As the result, the network tends to favor short connections over the long connections at high network loads, which leads to the overall decrease in reserved capacity per demand (or decrease in the delay performance).

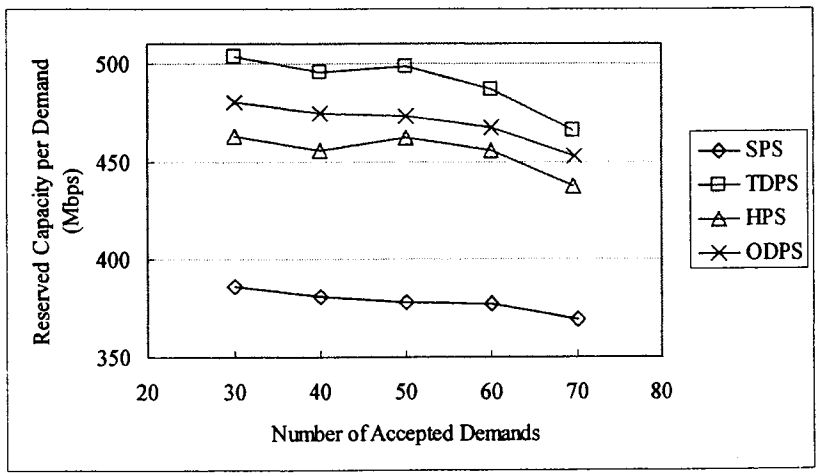


Figure 4.7 Average reserved capacity per demand in NSF

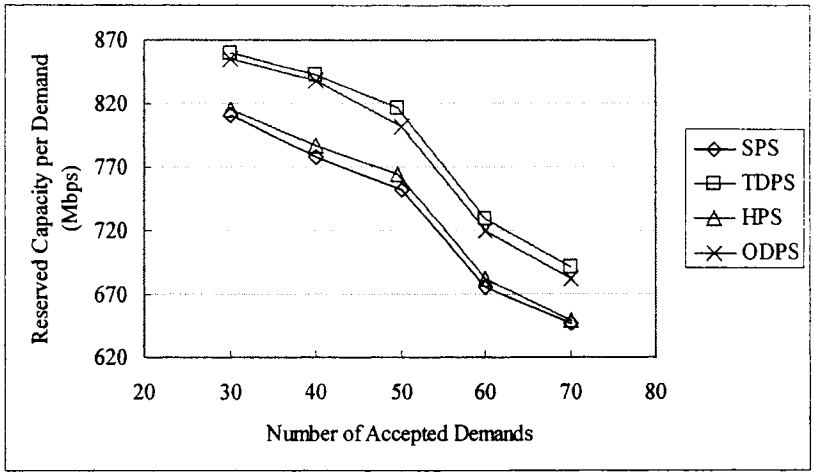


Figure 4.8 Average reserved capacity per demand in GCN

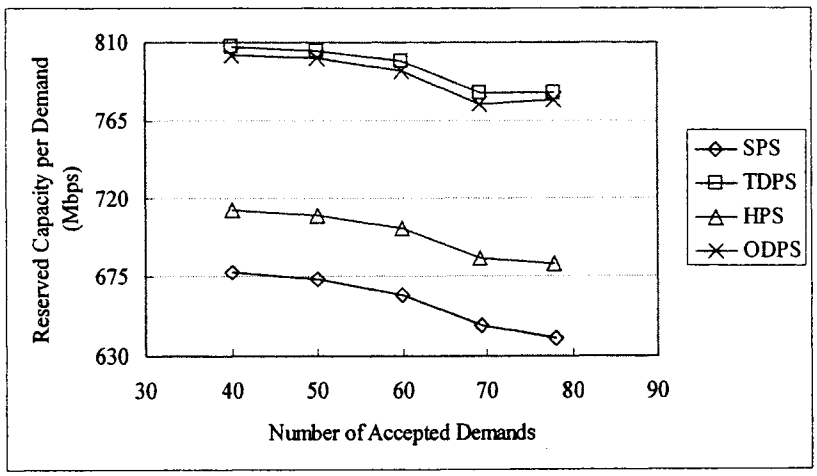


Figure 4.9 Average reserved capacity per demand in MCI

4.5.3. Computation Complexity

The computation complexity of the SPS, TDPS, and HPS algorithms are all $O(2 \cdot N \cdot \log[N])$, since these algorithms execute exactly twice the Dijkstra algorithm, whose complexity is $O(N \cdot \log[N])$ [50]. In phase 1 of ODPS, the Dijkstra algorithm is executed once. In phase 2 of ODPS, the IRD algorithm is executed. Since there is a restoration process in IRD, the complexity of IRD is higher than the Dijkstra

algorithm, but it is still in the same order as the Dijkstra algorithm. Therefore, the computation complexity of the ODPS algorithm is also $O(2 \cdot N \cdot \log[N])$.

4.5.4. Sensitivity Analysis

In this section, we investigate the robustness of the TDPS and ODPS algorithms to the variations in parameter δ. Figure 4.10 and Figure 4.11 show the D and V plots obtained for the TDPS algorithm with four different values for the parameter δ, in the NSF network. In both figures, plot A corresponds to $\delta = 5 \times d_{min}$, plot B corresponds to $\delta = 2 \times d_{min}$, plot C corresponds to $\delta = 1 \times d_{min}$, and plot D corresponds to $\delta = 0.2 \times d_{min}$, where d_{min} is the minimum of d_j's for all links j in the network. These figures show that the results are not very sensitive to the particular setting of the parameter δ, when the value of δ is less than $1 \times d_{min}$. Similar results were obtained when we varied the value of δ for the TDPS algorithm (and also ODPS algorithm) in the GCN and MCI networks. The corresponding results have been omitted for the sake of space.

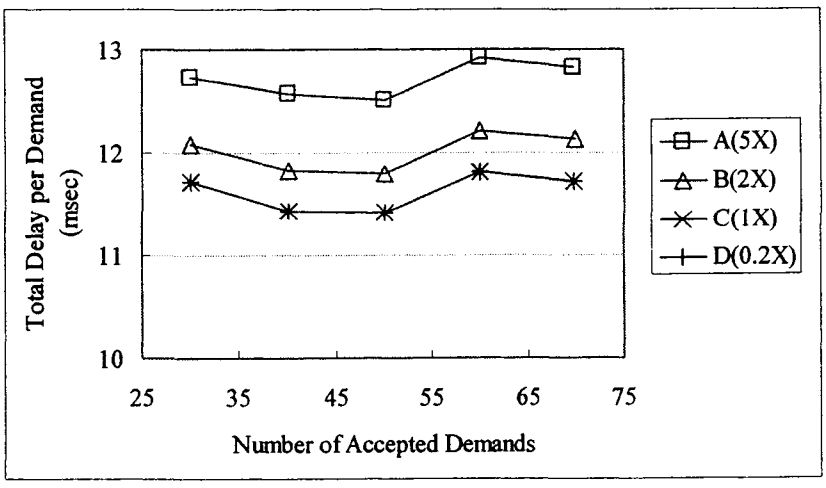


Figure 4.10 Total delay performance of TDPS in NSF for various values of parameter δ

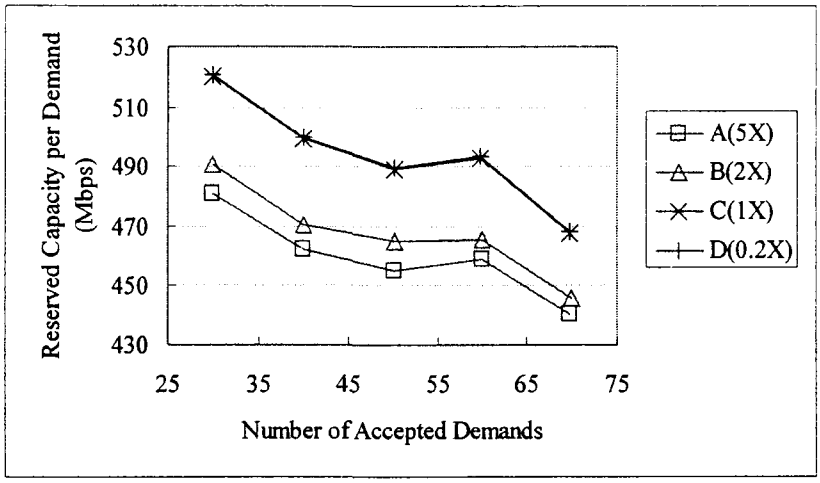


Figure 4.11 Average reserved capacity of TDPS in NSF for various values of parameter δ

4.5.5. Compare ODPS with TDPS

In this section, we compare ODPS with TDPS by studying the packet delay and the demand-blocking performances of the ODPS and TDPS. We have generated three random demand matrices for each network by using the same method as used in the above sections. However, we do not fix the number of generated demands in these matrices. Instead, we generate enough number of demands so that the total used capacity in the network will be 30%, 50%, and 70% respectively, i.e. the three demand matrices will generate a network load of 0.3, 0.5, and 0.7 respectively. All demands requested an end-to-end delay of not more than 20 msec along the working or backup path. If the achieved delay on either of the two paths (working or backup) is more than the requested delay bound, the demand will be rejected.

The related end-to-end delay and demand-blocking performances of the TDPS and ODPS schemes in all three tested networks are shown in Figure 4.12 through Figure 4.17, each as a function of the network load. These figures confirm the

improved performances of the ODPS scheme compared with those of the TDPS. With the ODPS scheme, the delay on the working path (ODPS_w) and the delay on the backup path (ODPS_b) are closer together than those of the TDPS scheme for all load levels shown. But the main advantage of the ODPS scheme is clearly the demand-blocking, being much lower than that of the TDPS scheme for all load levels shown in Figure 4.15 through Figure 4.17. The improved blocking ratio is yielded by complete elimination of trap topology problem and the reduction in the gap between the delay times on the working and backup paths for every demand.

Figure 4.12 through Figure 4.14 show that the average end-to-end delay per demand decreases when the network load increases. This is due to the similar reason explained for the phenomena in Figures 4.7 through 4.9 at the end of section 4.5.2.

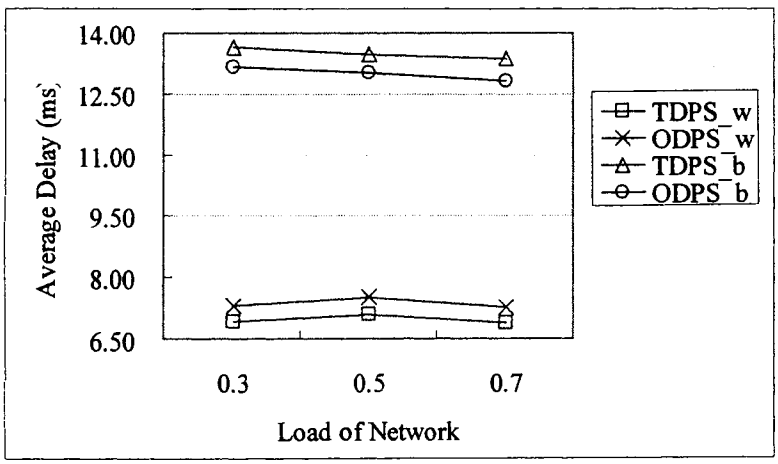


Figure 4.12 End-to-end delay of TDPS and ODPS schemes in NSF network

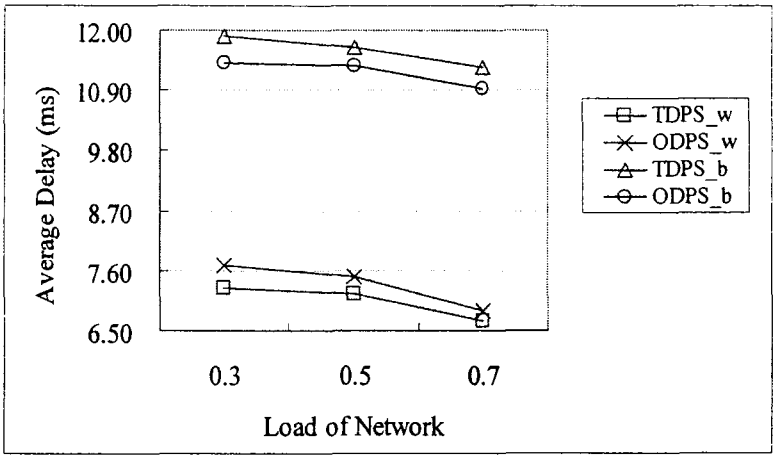


Figure 4.13 End-to-end delay of TDPS and ODPS schemes in GCN network

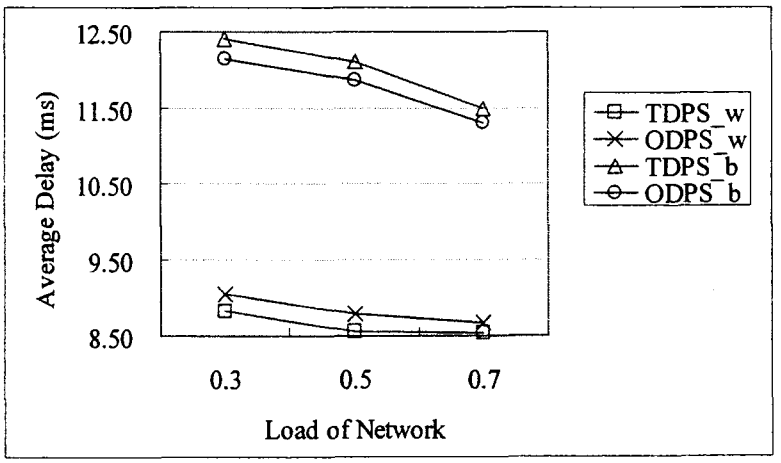


Figure 4.14 End-to-end delay of TDPS and ODPS schemes in MCI network

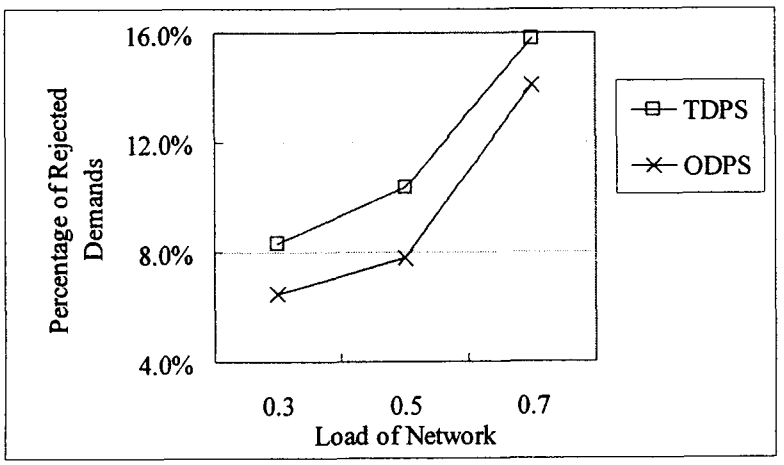


Figure 4.15 Percentage of blocked demands with TDPS and ODPS schemes in NSF network

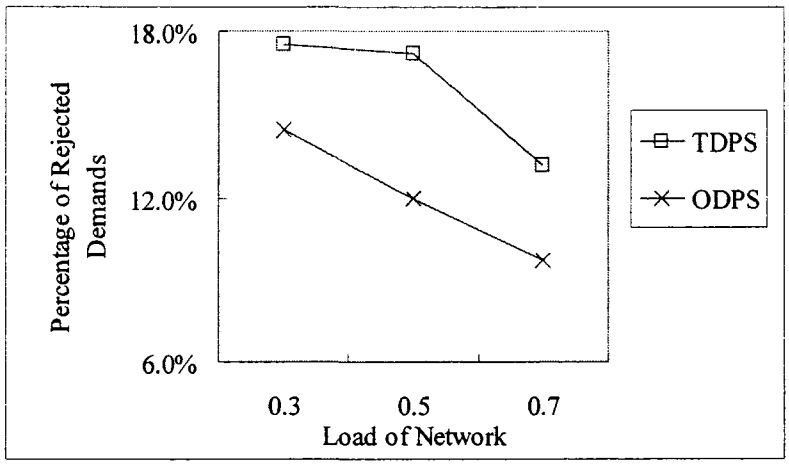


Figure 4.16 Percentage of blocked demands with TDPS and ODPS schemes in GCN network

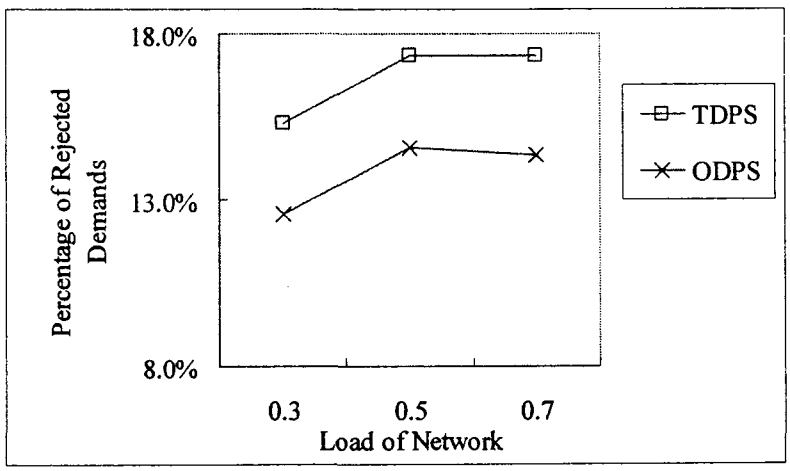


Figure 4.17 Percentage of blocked demands with TDPS and ODPS schemes in MCI network

Chapter 5 PROPOSED LOAD BALANCING AND RESTORATION ALGORITHMS

This chapter presents three novel shared mesh restoration algorithms, two of which incorporate a load balancing technique aimed at enhancing the traffic load balance in the network, while still minimizing the use of capacity. These two algorithms are called the Load Balancing Pool Sharing (LBPS) scheme and the Iterative Load Balancing Pool Sharing (ILBPS) scheme. The third algorithm presented in this chapter is also an iterative shared mesh restoration (but not load balancing) algorithm, referred to as Iterative Simple Pool Sharing (ISPS) which is based on the SPS algorithm.

Like any other two-step schemes, the SPS and LBPS schemes suffer from the trap-topology problem. In order to eliminate the trap-topology problem, we introduce the two new heuristic algorithms, the ISPS and ILBPS algorithms. The ISPS algorithm uses an iterative process to avoid the trap-topology problem. The ILBPS algorithm uses same mechanism as LBPS to evenly assign link load, however, it can avoid the trap-topology problem by using similar iterative process as ISPS. Using simulations, we study these three algorithms on the NSF, GCN and MCI networks, and compare their traffic load balancing, capacity usage, and complexity performances with the representative algorithms (SPS, RLBH and ISTA). We show that while the LBPS and ILBPS schemes are as efficient as (and often better than) the alternative algorithms in terms of the capacity usage, they perform better in terms of traffic load distribution.

5.1. Load Balancing Pool Sharing (LBPS)

The LBPS scheme is a non-iterative shared mesh restoration algorithm that computes a pair of link-disjoint working and backup paths for a demand r in two steps. It adopts a cost assignment strategy based on the load balancing technique described below to assign link-costs for the working and backup path computations. We will use the notations and definitions in Table I (see Notations on page vii) to represent: the pool sharing matrix (Ω), the total available bandwidth on link j (A_j), the total reserved backup bandwidth on link j (B_j), the maximum backup bandwidth required on link j if a link in $S_W(r)$ fails (T_j), the total working bandwidth reserved on link j (W_j), and the total capacity on link j (M_j), where $M_j = W_j + B_j + A_j$.

In step 1, the LBPS algorithm computes a working path for demand r. The link cost function for working path computation is designed to achieve two goals: 1) to seek for the shortest path that traverses through links with sufficient bandwidth to accommodate demand r; and 2) to minimize the likelihood of using links that are heavily loaded. To achieve the second goal, we define a measure of load balance on link i as:

$$l_i = \frac{\alpha_1 W_i + \alpha_2 B_i}{M_i} \tag{4.1}$$

Weights α_1 and α_2 are small non--negative tunable parameters. Essentially, l_i measures the ratio between the weighted sum of the allocated working and backup capacities and the total capacity on link i. Some networks use the backup bandwidth

to carry low priority traffic, or *extra* traffic. The low-priority traffic is preempted when the protection switching occurs. In these networks, different values for α_1 and α_2 can be used to control (or to differentiate) the effect of the allocated working and backup capacities on the load factor l_i. However, when a value of one is used for α_1 and α_2, l_i is simply the fraction of the total capacity used on link i.

With the LBPS scheme, the link cost for working path computation is defined as:

$$C_W(i) = \begin{cases} \infty & b_r > A_i \\ 1 + l_i & b_r \le A_i \end{cases} \tag{4.2}$$

Condition 2 in the above equation ensures that if two candidate links i and j have enough available capacity to accommodate demand r (i.e. if $b_r \le A_i$ and $b_r \le A_j$), the link is favored by the path computation algorithm that has the lower load factor. A working path for demand r is found by running the Dijkstra's algorithm on the above cost function.

In step 2, the LBPS algorithm computes a link-disjoint backup path for the working path found in step 1. In backup path computation, the LBPS scheme pursues the following two goals: 1) avoid using heavily loaded links in working path computation; and 2) use links that have high enough level of sharable backup bandwidth. Both goals can be achieved by using a properly designed link cost function that allows the algorithm to use the sharable backup bandwidth (B_j) and the current load level (l_j) in every link j to guide the backup path computation. The link cost $C_B(j)$ for backup path computation is thus defined as:

$$C_B(j) = \begin{cases} \infty & j \in S_W(r) \\ l_j & T_j \le B_j \\ \dfrac{T_j - B_j}{b_r} + l_j & 0 < T_j - B_j \le A_j \\ \infty & \text{otherwise} \end{cases} \tag{4.3}$$

Condition 2 ensures that if two links j and k have adequate shared backup bandwidth to restore demand r ($T_j \le B_j$ and $T_k \le B_k$), the link with less load balance factor is favored by the path computation algorithm over the other link. With condition 3, the cost of link j is set to $(T_j - B_j) / b_r + l_j$, where $(T_j - B_j)$ is the amount of additional backup bandwidth that must be reserved to restore demand r, and if this additional bandwidth is available (i.e. if $0 < T_j - B_j \le A_j$). The term l_j has been added to this condition to allow a load-based selection of a backup link, for similar reasons given in condition 2 above. If the additional bandwidth is not available, the cost of link j is set to infinity.

5.2. Iterative Simple Pool Sharing (ISPS)

The Iterative Simple Pool Sharing (ISPS) algorithm is intended to minimize the total reserved capacity in the network, as in the SPS scheme. However, it achieves this goal by adopting a different mechanism based on an iterative process of searching for link-disjoint paths for a demand. The iterative process is designed in order to eliminate the trap topology associated with non-iterative two-step path computation algorithms, such as the SPS scheme.

In each iteration, the ISPS scheme attempts to find the shortest pair of link-disjoint working and shared backup paths between the source and destination nodes of the given demand at the current stage of the algorithm. If the pair of the paths is found the algorithm terminates successfully. If the pair is not found, it could be due to a link (or links) causing the trap-topology between the source and destination nodes. We call such a link as "*trap-link*". One example of the trap-link is the link BC for the demand between nodes A and Z in the topology shown in Figure 1.8.

In order to identify trap-links in the current iteration, the ISPS algorithm employs a technique based on the Suurballe's algorithm discussed earlier in this thesis. If a trap-link is found, it will be eliminated from the topology in the next iteration of the optimization process. The formal description of the ISPS algorithm is given below.

Let us denote T_L as the set of trap-links that are currently known. One iteration of the ISPS algorithm adds one or more new trap-links to T_L. The ISPS algorithm has four steps; steps 2 to 4 are repeated until a pair of link-disjoint paths have been found:

1. [Initialization]: $T_L = \emptyset$.

2. [Compute Working Path]: Run Dijkstra's algorithm to compute a working path $S_W(r)$ by using the following cost function for every link i:

$$C_W(i) = \begin{cases} \infty & i \in T_L \\ \infty & b_r > A_i \\ 1 & b_r \le A_i \end{cases} \tag{4.4}$$

If the working path is not found, demand r is blocked; otherwise a backup path is computed by the next step.

3. [Compute Backup Path]: Run Dijkstra's algorithm again to find a link-disjoint shared backup path $S_B(r)$ by using the link cost function (2.5). If the backup path is found, the algorithm terminates successfully and returns the pair of paths. Otherwise, the algorithm proceeds to the next step to detect any trap-link that might have blocked the computation of the backup path.

4. [Find Trap-links]: Run the Dijkstra's algorithm to find a temporary backup path $S'_B(r)$ which can possibly have overlaps with the working path $S_W(r)$, by using the following cost function:

$$C'_B(j) = \begin{cases} Z_j & j \in S_W(r) \\ \varepsilon & T_j \le B_j \\ \dfrac{T_j - B_j}{b_r} & 0 < T_j - B_j \le A_j \\ \infty & \text{Otherwise} \end{cases} \tag{4.5}$$

Where

$$Z_j = \begin{cases} \infty & \text{To\_Destination} \\ -\delta & \text{To\_Source} \end{cases}, \quad \delta << \varepsilon$$

If $S'_B(r)$ is found, the trap-links will be the interlacing links of the two sets $S_W(r)$ and $S'_B(r)$. Add these links to T_L, and start a new iteration. If $S'_B(r)$ is not found, the demand is blocked.

Note that condition 1 ($i \in T_L$) in (4.4) is designed to avoid using the trap-links found in the previous iteration in the working path computation at the current iteration. These links are indeed the links that have met condition 1 of the cost function (4.5), used in the previous iteration. From the second part of (4.5), the ISPS scheme replaces every link along the path $S_W(r)$ (found in Step 1) by two unidirectional links (arcs); one directed toward the destination (To_Destination), and the other toward the source node (To_Source). The cost of the arc toward the destination is set to infinity ($Z_j = \infty$), whereas the cost of the arc toward the source is set to a very small negative number ($Z_j = -\delta$). The costs of all other links not in $S_W(r)$ are set to either ε or $(T_j - B_j) / b_r$ depending on how much shared backup bandwidth already reserved on these links.

5.3. Iterative Load Balancing Pool Sharing (ILBPS)

The Iterative Load Balancing Pool Sharing (ILBPS) algorithm introduced in this section is intended to achieve two goals set earlier for the LBPS scheme. The first goal is to minimize the reserved capacity, and the second goal is to evenly distribute

link load. However, it achieves these goals by adopting an iterative procedure, which is very similar to that of the ISPS scheme, in order to avoid the trap topology associated with the non-iterative LBPS scheme.

The structure of the ILBPS scheme is very similar to the ISPS scheme. It employs the same 4-step iterative procedure used in the ISPS scheme to compute a pair of link-disjoint paths for every demand. Thus the computation complexity of the ILBPS scheme is the same as the ISPS scheme. The difference between the ILBPS and ISPS schemes is that the ILBPS scheme uses the following link-cost functions that are different from those used in the ISPS scheme. In step 2, the ILBPS scheme uses the following link-cost function, instead of (4.4), to compute a working path:

$$C_W(i) = \begin{cases} \infty & i \in T_L \\ \infty & b_r > A_i \\ 1 + l_i & b_r \le A_i \end{cases} \tag{4.6}$$

In step 3, the ILBPS scheme uses the cost function (4.3), instead of (2.5), to find a link-disjoint shared backup path. If this path is not found the algorithm proceeds to step 4 by using the following cost function, instead of (4.7), to determine trap-links:

$$C'_B(j) = \begin{cases} Z_j & j \in S_W(r) \\ l_j & T_j \le B_j \\ \dfrac{T_j - B_j}{b_r} + l_j & 0 < T_j - B_j \le A_j \\ \infty & \text{otherwise} \end{cases} \tag{4.7}$$

Where

$$Z_j = \begin{cases} \infty & \text{To\_Destination} \\ -\delta & \text{To\_Source} \end{cases}$$

5.4. Simulation Results

We have used simulation technique to test the LBPS, ISPS, and ILBPS schemes and also to compare their performances with the representative shared mesh restoration schemes: SPS, ITSA, and RLBH. The simulation has been carried out to measure the performance metrics such as the load balancing, capacity usage, and computation time.

For the purpose of measurement, we have defined the load on link k (denoted by L_k) to be the sum of the allocated working and backup capacities ($W_k + B_k$) divided by the total capacity of the link (M_k). We have used the standard deviation of the sample L_k (denoted by S) as a measure of load distribution (load balancing) amongst network links. The standard deviation S is calculated by taking the square root of the sum of $[(L_k - L')^2 / J]$ over all possible values of k ($1 \ldots J$), where L' is the mean of the sample L_k ($L' = \sum [L_k] \quad / J$). In terms of load balancing, a diverse paths computation scheme is said to outperform other candidate schemes if it achieves the smallest S.

We have used the average reserved capacity per accepted demand (V) in order to evaluate the capacity performance of the mesh restoration schemes presented in this thesis. We compute V as the sum of ($W_k + B_k$) over all values of k divided by n_d, where n_d is the number of accepted demands in the network.

We used the three representative North American backbone networks (NSF, GCN and MCI) which were also used in Chapter 4 to test the criteria defined above. Some important characteristics of these networks are shown in Table 5.1. The only difference between Table 5.1 and Table 4.1 is the number of demands in each demands matrix.

In this section, we also assume that all links in the network have equal capacity of M_k = 2.4 Gbps. We generated six demand matrices for each network, with the total number of demands in each matrix was: 50 / 80 / 100 / 150 / 200 / 250 for NSF, 50 / 100 / 200 / 300 / 500 / 700 for GCN, and 50 / 100 / 200 / 300 / 500 / 1000 for MCI. For every demand, the source and destination nodes were generated randomly. In our simulations, all demands requested identical amount of bandwidth b_r = 100 Mbps from the network.

As a baseline, we set the values of the system parameters (α_1, α_2, δ) to (0.1, 0.01, 0.001), respectively. We will however investigate the impact of different values for these parameters on S and V metrics, later in this chapter. The maximum number of iterations (K) of the ITSA algorithm is set to 50, which has been shown to be an optimal number in [23]. In order to achieve a reasonable confidence interval, we have repeated each simulation scenario 50 times and the final values for S and V metrics were obtained as the average over all 50 simulations.

TABLE 5.1 NETWORK CHARACTERISTICS

Network	No. of Nodes	No. of Links	Capacity	No. of Demands in Each Traffic Matrix
NSF	16	25	2.4G	50 / 80 / 100 / 150 / 200 / 250
GCN	27	38	2.4G	50 / 100 / 200 / 300 / 500 / 700
MCI	38	67	2.4G	50 / 100 / 200 / 300 / 500 / 1000

5.4.1. Load Balancing

As stated earlier, we use the standard deviation of link load (S) as a measure of load balancing in this thesis. The values of S for the three proposed restoration schemes (LBPS, ISPS, ILBPS), and the three representative restoration schemes (SPS, RLBH, ITSA) are shown in Figure 5.1 for the NSF network, Figure 5.2 for the GCN network, and Figure 5.3 for the MCI network. The data is shown as a function of the number of accepted demands in the network (n_d).

The above figures show that the standard deviation S for the LBPS, ILBPS, and ITSA schemes are significantly smaller than the corresponding metric for the SPS, ISPS, and RLBH schemes, at low to medium network load. There is no significant difference in parameter S across these six schemes at heavy load, with the exception that the RLBH scheme slightly outperforms other schemes at this load level. This is expected, because when the network load is high, all links are loaded nearly to their full capacity. Therefore, the standard deviation S of the six schemes are almost the same at high network load.

Thus, we can conclude this section as follows: (1) the load balancing schemes LBPS and ILBPS distribute the network load more evenly than the SPS and ISPS schemes; (2) they outperform the RLBH load balancing scheme at low to medium

load; and (3) they perform comparatively with the ITSA scheme.

Figure 5.1 through Figure 5.3 show that the standard deviation of the link load (S) increases initially and decreases after it reaches a specific value with the increase in the number of accepted demands (or with the increase in network load). When the network load is high, all the network links are nearly loaded to their full capacity. As a result, the variance in the reserved capacity among the links decreases. We can also see from Figure 5.1 through Figure 5.3 that the parameter S of RLBH is lower than the other algorithms when the number of accepted demands is high. It is because; the total reserved capacity per demand of RLBH (shown in Figures 5.4 through 5.6) is higher than the other algorithms. This means that, with the RLBH algorithm, the network links are much more fully loaded to their 100% capacity than with the other algorithms, when the network load is high. Therefore, the parameter S of RLBH decreases faster than the other algorithms.

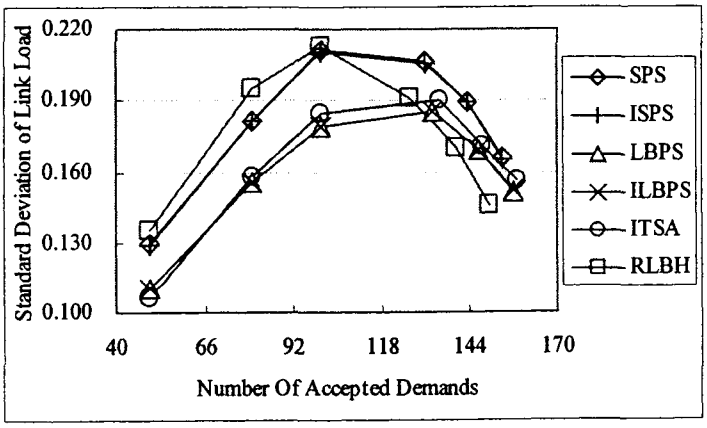


Figure 5.1 Standard deviation of link load in NSF

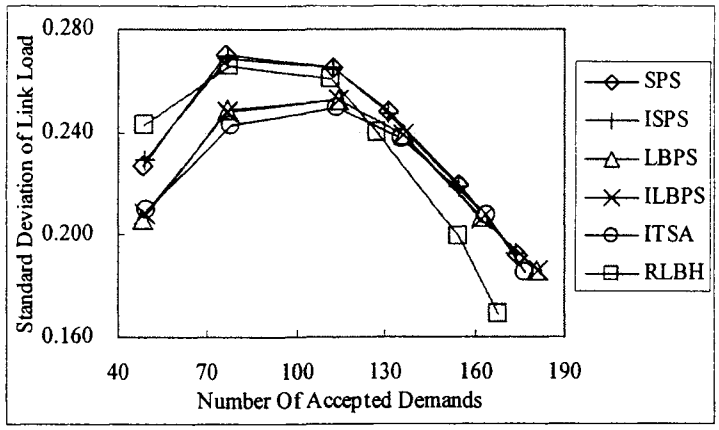


Figure 5.2 Standard deviation of link load in GCN

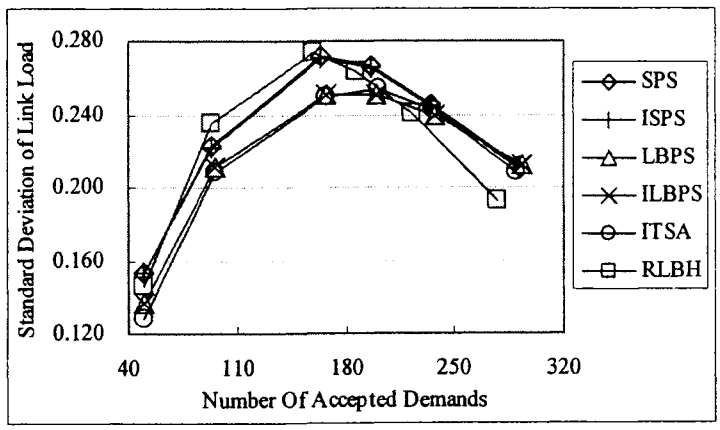


Figure 5.3 Standard deviation of link load in MCI

5.4.2. Capacity Usage

The related average reserved capacity per accepted demand (V) of all the six restoration schemes in the three networks are shown in Figure 5.4 through Figure 5.6, each as a function of the number of accepted demands (n_d) in the network. The following important observations can be made from these figures: (1) in all of the three tested networks, the SPS and ISPS schemes perform equally, almost identically; (2) the same observation is true for the LBPS and ILBPS schemes; (3) the LBPS and

ILBPS schemes always outperform the SPS and ISPS schemes; and (4) the LBPH scheme always yields the worst capacity performance when compared with other schemes investigated. Observation (3) was not particularly to be expected, because the SPS and ISPS schemes attempt solely to minimize the capacity usage in the network. Whereas the LBPH and ILBPH schemes attempt to simultaneously minimize the capacity usage and distribute the network load evenly amongst network links. Thus, these algorithms may often make a tradeoff between capacity performance and load balancing.

Figure 5.4 through Figure 5.6 show that the average reserved capacity per accepted demand decreases when the number of accepted demands increases. The reason for this was explained in section 4.5.2.

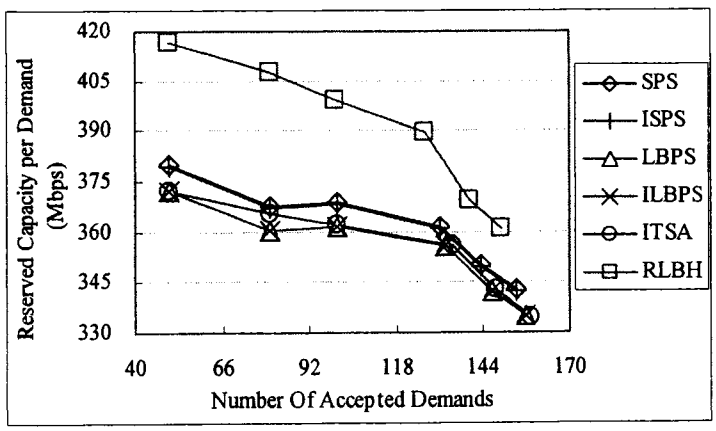


Figure 5.4 Average reserved capacity per accepted demand in NSF

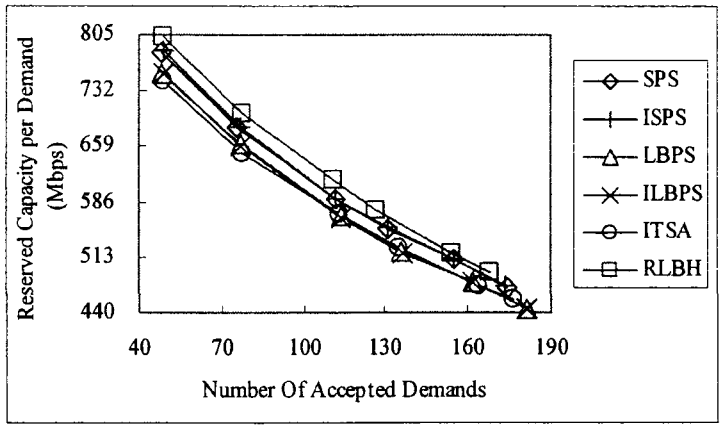


Figure 5.5 Average reserved capacity per accepted demand in GCN

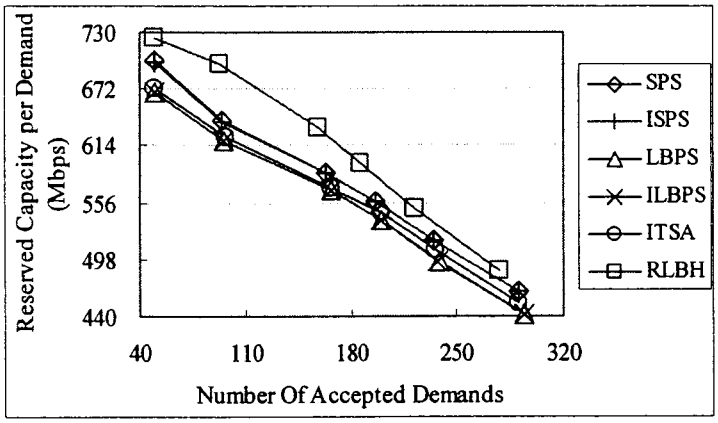


Figure 5.6 Average reserved capacity per accepted demand in MCI

5.4.3. Computation Complexity

The computation complexity of the ITSA scheme has been shown to be $O(K \cdot N^2 \cdot \log[N])$, where N is the number of nodes in the network, and K is the number of iterations [23]. The computation complexity of the SPS, RLBH, and LBPS algorithms can be derived as $O(2 \cdot N \cdot \log[N])$, since these algorithms execute the Dijkstra algorithm two times, and the complexity of the Dijkstra algorithm has been shown to be $O(N \cdot \log[N])$ [50]. The computation complexity of the ISPS and ILBPS

algorithms can be obtained as $O(3 \cdot p \cdot N \cdot \log[N])$, where p is the number of iterations of the algorithm. This is $(3 \times p)$ times the computation complexity of the Dijkstra's algorithm, which is executed in each of the three steps 2 to 4 of the algorithm.

From the above theoretical calculation, it is obvious that the computational complexity of the ITSA scheme is one order of magnitude higher than that of the other schemes investigated in this thesis. We verified the above calculation by executing all the simulations on a Lenovo/IBM personal computer with Intel Pentium D-945 CPU and 1 GB memory. Note that when we simulated the ISPS and ILBPS algorithms, we found that p is not more than 3. The computation time spent by each algorithm to compute link-disjoint paths for all the demands in every demand-matrix is shown in Figure 5.7 through Figure 5.9. The experimental results shown confirm the relative computational complexity of the six algorithms, determined theoretically above.

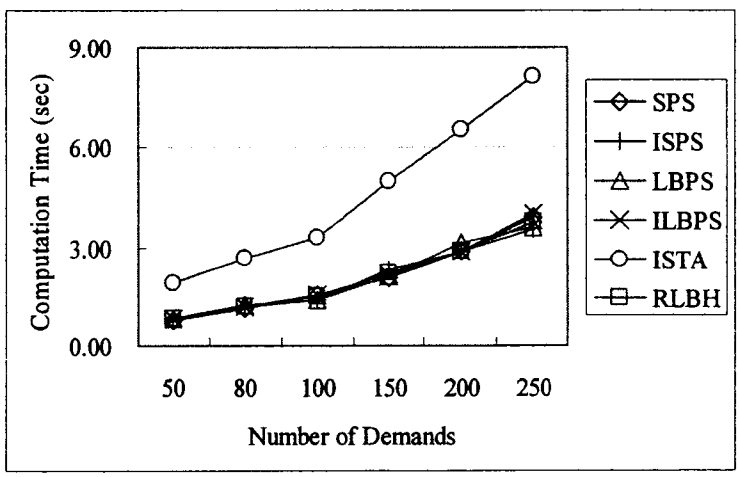


Figure 5.7 Computation time of the algorithms in NSF

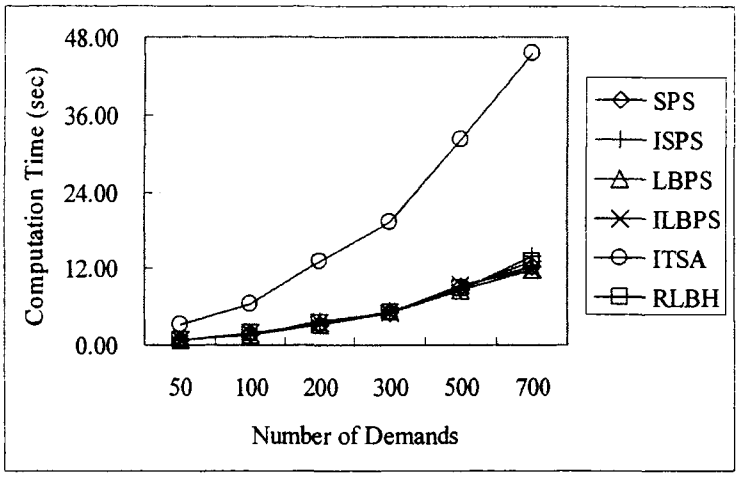


Figure 5.8 Computation time of the algorithms in GCN

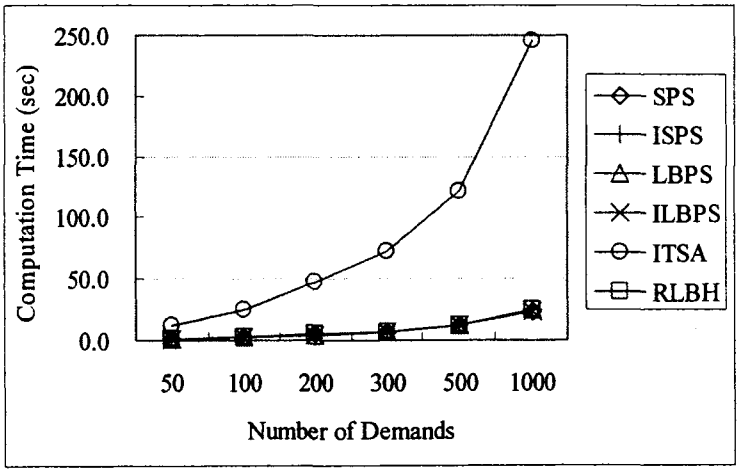


Figure 5.9 Computation time of the algorithms in MCI

5.4.4. Sensitivity Analysis

In this section, we investigate the robustness of the LBPS algorithm to the variations in parameters α_1 and α_2. Figure 5.10 and Figure 5.11 show the *S* and *V* plots obtained for this algorithm with four different sets of values for these parameters, in the NSF network. In both figures, plot A corresponds to parameter values of $\alpha_1 = 0.1$ and $\alpha_2 = 0.01$, plot B corresponds to $\alpha_1 = 1$ and $\alpha_2 = 0.5$, plot C corresponds to $\alpha_1 = 6$ and $\alpha_2 = 3$, and plot D corresponds to $\alpha_1 = 24$ and $\alpha_2 = 12$. Other parameters have

been set as explained in the previous sections. In particular, the bandwidth of each demand has been kept to be 100Mbps as before. These figures show that the results are not very sensitive to the particular setting of these parameters. Similar results were obtained when we varied the values of these parameters for the LBPS algorithm (and also ILBPS algorithm) in the GCN and MCI networks. The corresponding results have been omitted for the sake of space.

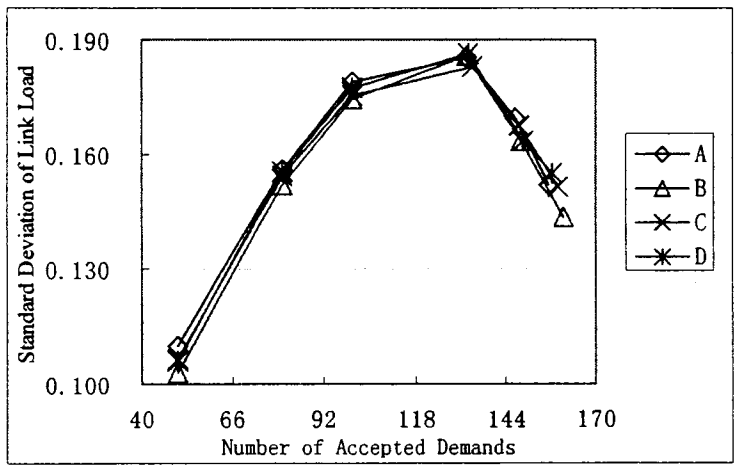


Figure 5.10 Impact of parameters $\alpha_1$and α_2 on the LBPS algorithm's link load performance (S) in NSF

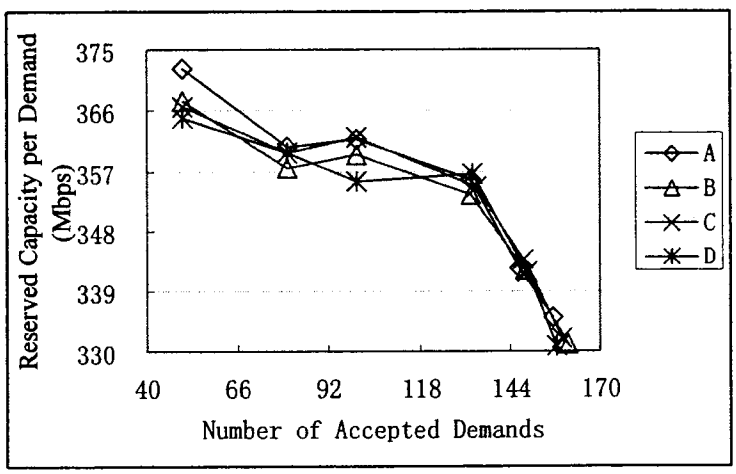


Figure 5.11 Impact of parameters $\alpha_1$and α_2 on the LBPS algorithm's average reserved capacity in NSF

Chapter 6 CONCLUSION AND FUTURE WORK

6.1. Conclusion

We have introduced three novel delay-constrained shared mesh restoration schemes, called Two-Step Delay-Constrained Pool Sharing (TDPS), Hybrid Pool Sharing (HPS), and One-Step Delay-Constrained Pool Sharing (ODPS). While the TDPS and ODPS schemes specifically aimed at minimizing the end-to-end delay of the demands in the network, the HPS aimed at both minimizing the working delay of the demands and saving the use of backup capacity. We simulated these algorithms on the NSF, GCN, and MCI transport networks (which are representative of North American transport backbone networks), and investigated their performances --- in terms of average total end-to-end delay and average reserved capacity of each accepted demand. We have shown that the TDPS and ODPS schemes can yield link-disjoint paths with much less total end-to-end delay than the alternative SPS scheme (presented in the literature), at the cost of minor increase in the capacity usage. We have also shown that the one-step ODPS algorithm outperforms the two-step TDPS scheme in terms of both the delay and capacity performances. It also avoids the trap-topology problem associated with the two-step survivable routing algorithms. We have also developed a hybrid algorithm called Hybrid Pool Sharing (HPS) intended to combine the merits of the TDPS and SPS schemes. While the HPS scheme achieves the same end-to-end delay performance along the working path as with the TDPS scheme, it yields the same amount of backup bandwidth sharing as in the SPS scheme.

Then, we introduced other three novel shared mesh restoration algorithms, called

Load Balancing Pool Sharing (LBPS), Iterative Simple Pool Sharing (ISPS), and Iterative Load Balancing Pool Sharing (ILBPS). While the ISPS scheme specifically aimed at minimizing the use of capacity in the network, the LBPS and ILBPS aimed at both minimizing the capacity and enhancing the traffic load distribution among the network links. We simulated these algorithms on the NSF, GCN, and MCI networks, and investigated their performances --- in terms of even distribution of capacity on all links, total reserved capacity, and computation complexity. We have shown that the LBPS and ILBPS schemes can more fairly distribute the capacity among network links than the alternative schemes (presented in the literature) at the cost of less capacity and/or lower computation cost. The iterative ISPS and ILBPS algorithms can also avoid the trap-topology problem associated with the traditional non-iterative two-step survivable routing algorithms.

6.2. Future Work

In this thesis, we have considered the end-to-end delay and load balancing separately in different algorithms. In the future work, we plan to develop new algorithms that simultaneously incorporate the two QoS requirements as constraints.

In the algorithms proposed in this thesis, end-to-end delay and load balancing are considered as the only QoS requirements of a network service. Actually, there are many other QoS requirements in practice, such as the delay variation (jitter) and data loss. In the future work, these additional QoS requirements will be considered in new routing algorithms.

In addition, we will also examine different levels (classes) of protection in networks. For different applications, networks will provide different levels of protections, such as dedicated protection, shared restoration, and no protection. By integrating these protection levels in a network, the efficiency of the network can be enhanced significantly.

REFERENCE

[1] R. Bhandari, "*Survivable Networks: Algorithms for Diverse Routing*," Kluwer Academic Publishers, Boston, 1999

[2] S. Ramamurthy, L. Sahasrabuddhe, and B. Mukherjee, "Survivable WDM Mesh Networks," J. of Lightwave Technology, Vol. 21, No. 4, pp. 870 - 883 (2003)

[3] P. Tomsu and C. Schmutzer "*Next Generation Optical Networks: The Convergence of IP Interligence and Optical Technology*," Prentice Hall PTR (2002)

[4] J. V. Drake " A Review of the Four Major SONET/SDH Rings," IEEE International Conference on Communications, ICC 93, Vol. 2, pp. 878 - 884 (1993)

[5] M. W. Nonkomo and R. Sewsunker " Optical Backbone Topology Choice," 7th AFRICON Conference in Africa, AFRICON, Vol. 1, pp. 353 - 359 (2004)

[6] J. L. Marzo, E. Calle, C. Scoglio, and T. Anjah, "QoS Online Routing and MPLS Multilevel Protection: A Survey," IEEE Communications Magazine, Issue 10, Vol. 41 (2003)

[7] H.J. Chao and X. Guo, "*Quality of Service Control in High-Speed Networks*," John Wiley & Sons. Inc (2002)

[8] S. L. Tan, C. K. Tham, and L. H. Ngoh, "QoS-based connection set-up in ATM networks," Proc. of IEEE International Conference on Networks, ICON 2000, pp.115 - 119 (2000)

[9] A. Vasilakos, M. P. Saltouros, A. F. Atlassis, and W. Pedrycz, "Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques," IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, Issue 3, Vol. 33, pp. 297 - 312 (2003)

[10]H. de Neve and P. van Mieghem, "A multiple quality of service routing algorithm for PNNI," IEEE ATM Workshop Proceedings, pp. 324 - 328 (1998)

[11]A. Leon-Garcia and I. Widjaja, "*Communication Networks Fundamental Concepts and Key Architectures, Second Edition,*" Mc Graw Hill (2004)

[12]Y. D. Lin, N. B. Hsu, and R. H. Hwang, "QoS routing granularity in MPLS networks," IEEE Communications Magazine, Issue 6, Vol. 44, pp. 58 - 65 (2002)

[13]M. C. Yuen, W. Jia, and C. C. Cheung, "Efficient distributed QoS routing protocol for MPLS networks," Proc. of 11th International Conference on Parallel and Distributed Systems, Vol. 1, pp. 342 - 348 (2005)

[14]D. Ghosh, V. Sarangan, and R. Acharya, "Quality-of-service routing in IP networks," IEEE Transactions on Multimedia, Issue 2, Vol. 3, pp. 200 - 208 (2001)

[15]A. Dubrovsky, M. Gerla, S. S. Lee, and D. Cavendish, "Internet QoS routing with IP telephony and TCP traffic," IEEE International Conference on Communications, ICC 2000, Vol. 3, pp. 1348 - 1352 (2000)

[16]A. Maachl, G. V. Bochmanl, and H. Mouftah, "Congestion Control and Contention Elimination in Optical Burst Switching," Telecommunication Systems, Vol 27, No. 2-4, pp. 115 - 131 (2004)

[17]C. Qiao, Y. Xiong and D. Xu, "Novel Models for Efficient Shared-Path Protection," Proc. of Optical Fiber Communication Conference and Exhibit 2002 (OFC 2002), pp. 546 - 547 (2002)

[18]T. Lee, K. Lee, and S. Park, "Optimal Routing and Wavelength Assignment in WDM Ring Networks," IEEE Journal on Selected Areas in Communications, Issue 10, Vol. 18, pp. 2146 - 2154 (2000)

[19]M. Patel, R. Chandrasekaran, S. Venkatesan, "A Comparative Study of Restoration Schemes and Spare Capacity Assignments in Mesh Networks," Proc. of The 12th International Conference on Computer Communications and Networks (ICCCN 2003), pp.399 - 404 (2003)

[20]H. Naser and H. Mouftah, "Enhanced Pool Sharing: A Constraint-Based Routing Algorithm for Shared Mesh Restoration Networks," J. of Optical Network, Vol. 3, Issue 5, pp. 303 - 323 (2004)

[21]Y. Liu, D. Tipper, P. Siripongwutikorn, "Approximating Optimal Spare Capacity Allocation by Successive Survivable Routing," Proceedings IEEE INFOCOM 2001, vol. 2. 699–708 (2001)

[22]S. Datta, S. Sengupta, S. Biswas, S. Datta., "Efficient Channel Reservation for Backup Paths in Optical Mesh Networks," Proc. of the Global Telecommunications Conference (GLOBECOM'01), Vol. 4, pp. 2104 - 2108 (2001)

[23]P.-H. Ho, H.T. Mouftah, "Shared Protection in Mesh WDM Networks," IEEE Communications Magazine, Vol. 42, Issue 1, pp. 70 - 76 (2004)

[24]P.-H. Ho, J. Tapolcai, H.T. Mouftah, and C.-H. Yeh, "Linear Formulation for Path Shared Protection," Proc. of IEEE International Conference on Communications 2004 (ICC04), Vol. 3, pp. 1622- 1627 (2004)

[25]R. Rainer, M. Iraschko, H. MacGregor, "Optimal Capacity Placement for Path Restoration in STM or ATM Mesh-Survivable Networks," IEEE/ACM Transactions on Networking, Vol. 6, Issue 3, pp. 325 - 335 (1998)

[26]S. Yuan, and J. P. Jue, "A Heuristic Routing Algorithm for Shared Protection in Connection-Oriented Network," Optical Networking and Communications, Vol. 4599, pp. 142 - 152 (2001)

[27]A. Todimala and B. Ramamurthy, "IMSH: An Iterative Heuristic for SRLG Diverse Routing in WDM Mesh Networks", Proc. of IEEE ICCCN 2004, pp. 199-204 (2004)

[28]A. Todimala, B. Ramamurthy, N.V. Vinodchandran, "On Computing Disjoint Paths with Dependent Cost Structure in Optical Networks," Proc. of 2005 2nd International Conference on Broadband Networks, Vol. 1, pp. 145 - 154 (2005)

[29]M. Kodialam and T. V. Lakshman, "Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels Using Aggregated Link Usage Information," Proc. of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), pp. 376 - 385 (2001)

[30]W. Wen, S.J. Ben Yoo, B. Mukherjee, "Quality-of-Service Protection in MPLS Control WDM Mesh Networks," Photonic Network Communications, Vol. 4, issue ¾, pp. 297 - 320 (2002)

[31]G. Li, B. Doverspike, and C. Kalmanek, "Fiber span failure protection in mesh optical networks", Proc. of SPIE OptiComm, Vol. 4599, pp.130-141 (2001)

[32]S. Chen and K. Nahrstedt, "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions," IEEE Network Magazine, Vol. 12, Issue 6, pp. 64 - 79 (1998)

[33]L. Ruan, H. Luo, and C. Liu, "A Dynamic Routing Algorithm with Load Balancing Heuristics for Restorable Connections in WDM Networks," IEEE J. on Selected Areas in Communications, Vol. 22, Issue 9, pp. 1823 - 1829 (2004)

[34]X. Su and C.-F. Su, "An Online Distributed Protection Algorithm in WDM Networks," Proc. of IEEE International Conference on Communications (ICC 2001), Vol. 5, pp. 1571-1575 (2001)

[35]L. N. Vaserstein, "*Introduction to Linear Programming*," Pearson Education (2003)

[36]C. Mauz, "Unified ILP formulation of protection in mesh networks," Proc. of the 7th International Conference on Telecommunications, ConTEL 2003, Vol. 2, pp. 737 - 741 (2003)

[37]H. Long and S. Shen, "QoS Routing in Communication Networks: Approximation Algorithms Based on the Primal Simplex Method of Linear Programming," International Conference on Computer Networks and Mobile Computing, ICCNMC 2003, pp. 61 - 67 (2003)

[38]V. Pasias, D. A. Karras, and R. C. Papademetriou, "A Framework for Traffic Engineering and Routing in Survivable Multi-service High Bit Rates Optical Networks," 48th International Symposium ELMAR-2006 focused on Multimedia Signal Processing and Communications, pp. 353 - 357 (2006)

[39]M. Menth, R. Martin, and U. Spoerlein, "Network Dimensioning for the Self-Protecting Multi-path: A Performance Study," IEEE International Conference on Communications, Vol. 2, pp. 847 - 853 (2006)

[40]R. A. Guerin, A. Orda, and D. Williams, "QoS Routing Mechanisms and OSPF Extensions," Proc. of the Global Telecommunications Conference (GLOBECOM '97), Vol. 3, pp. 1903 - 1908 (1997)

[41]S. Bak, A.M.K. Cheng, J.A. Cobb, E.L. Leiss, "Load-Balanced Routing and Scheduling for Real-Time Traffic in Packet-Switch Networks," Proc. of the 25th Annual IEEE International Conference on Local Computer Networks (LCN'00), pp. 634 - 643 (2000)

[42]Y. Du; T. Pu; H.Y. Zhang, and Y.L. Quo, "Adaptive Load Balancing Routing Algorithm for Optical Burst-Switching Networks," Proc. of Optical Fiber Communication Conference, 2006 and the 2006 National Fiber Optic Engineers Conference (OFC 2006), pp. 3 - 5 (2006)

[43]K. Gopalan, T. Chiueh, and Y.-J. Lin, "Network-Wide Load Balancing Routing with Performance Guarantees," Proc. of the IEEE International Conference on Communications (ICC'06), Vol. 2, pp. 943 - 948 (2006)

[44]H. Naser and H. T. Mouftah, "Availability Analysis and Simulation of Shared Mesh Restoration Networks," Proc. of the 9th IEEE Int'l Symposium on Computers and Communications (ISCC 2004), Alexandria, Egypt, pp. 779-785 (2004).

[45]H. F. Salama, D. S. Reeves, Y. Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing," IEEE INFOCOM '97, Japan, pp. 239 - 250 (1997).

[46]R. Guerin, A. Orda, "QoS routing in networks with inaccurate information: theory and algorithms," IEEE INFOCOM '97, Japan, pp. 350 - 364 (1997).

[47]Z. Wang, J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," IEEE JSAC, pp. 1228 - 1234 (1996).

[48]C. V. Saradhi and C. Siva Ram Murthy, "Dynamic Establishment of Segmented Protection Paths in Single and Multi-Fiber WDM Mesh Networks," Proc. of Optical Networking and Communications (SPIE OptiComm 2002), pp. 211 - 222 (2002)

[49] J. Y. Yen, "Finding The K Shortest Loopless Paths in A Network," Management Science, Vol. 17, No. 11, pp. 712 - 716 (1971)

[50] W. D. Grover, "Mesh-Based Survivable Networks: Options and Strategies for Optical, MPLS, SONET, and ATM Networking," Prentice Hall 2003.

APPENDIX A: PSEUDO-CODE OF PSEUDO-CODE OF ITERATIVE RESTORATION DIJKSTRA (IRD)

Let us use the notations in the following table to describe the IRD algorithm.

TABLE A.1 NOTATIONS FOR THE IRD ALGORITHM

Notation	Description
V	set of nodes in the network
d_i	average delay of link i
$d(x)$	total cost of node x ($x \in$ Network V) from source node A; it is the sum of the cost of links in a possible path from node A to node x.
$l(xy)$	cost of link from node x to node y
$P(x)$	predecessor of node x on the same path to source A
Γ_x	set of neighbor nodes of node x
$C_1(i)$	cost of link i for the first lowest delay path computation
$C_2(j)$	cost of link j for the second lowest delay path computation.
$S_1(r)$	set of links of the first found lowest delay path
$S_2(r)$	set of links of the second found lowest delay path

a. Use the cost function (3.4) to set the cost of link j. This cost function is similar to the cost function of the TDPS scheme for computing the backup path. Again, we denote by $S_1(r)$ the set of links along the first path found in Phase 1 above.

Set: $l(xx) = 0$;
$l(xy) = \infty$, when there is no link between node x and node y;
$l(xy) = C_2(j)$, when node x and node y are connected by link j.

b. Initialization
Set $d(A) = 0$;

$$d(x) = \begin{cases} l(Ax) & x \in \Gamma_A \\ \infty & \text{otherwise} \end{cases};$$

$R = V - \{A\}$,

$P(i) = A \quad \forall x \in R$;

Initialize the following two variables:
Set variable 1: RESTORABLE = FALSE, which means that there is no partial restoration path saved in a buffer.
Set variable 2: IN_RESTORING = FALSE, which means that this iteration is not a restored one.

c. Find $y \in R$ such that $d(y) = \min[d(x)]$, $x \in R$;
Set $R = R - \{y\}$;
IF find a y, THEN: goto sub-step d.
ELSE:

1) IF IN_RESTORING = FALSE AND RESTORABLE = TRUE, THEN:
i) Save the First Path which can be obtained by using *P(x)* (begin with *P(Z)*);
ii) Restore the saved values of *P(x)*, *R*, *y*;
iii) Set IN_RESTORING = TRUE, Begin a restored iteration, Go to sub-step *d*.

2) IF IN_RESTORING = TRUE, THEN:
i) Save the Second Path;
ii) Compare the cost of the First Path and the Second Path;
iii) Select the lowest cost one to be the second lowest delay path.
If there is only computed one path, then this path is the second lowest delay path.

3) GOTO Phase 3.

d. $\forall t \in \Gamma_y$ and $t \in R$;

1) CHECK how many *negative-cost-link-groups* along recent path from source *A* to *t* the path can be got from the parameter *P(t)*;
Comment: *negative-cost-link-group* is a set of negative cost single direction link(s). if a negative cost link is connected to another one, they are belonged to one group; if there are two negative links, they aren't connected, they are belong to two group.

2) IF the number of *negative-cost-link-groups* is ODD, CHECK the available capacity of links along the path which is from source *A* to node *y*. In other words, is there any link whose available capacity is not enough for a working path?
i) IF NO, restore the *l(ty)* and *l(yt)* to original values which is set in sub-step a.
ii) IF YES, CHECK if the cost of the link between t and y is POSITIVE?
IF YES, CHECK if the available capacity of the link between t and y is enough for working path?
IF NO, Chang the *l(ty)* and *l(yt)* to infinity.
IF the number is EVEN, restore the *l(ty)* and *l(yt)* to original values which is set in sub-step a.

e. IF $d(y) + l(yt) < d(t)$, THEN:
1) IF $l(yt) < 0$ AND RESTORABLE = FALSE, THEN: save recent status including all values of *P(x)*, *S*, *y*, and save the restore point as (t_r, y_r) $t_r = t$ and $y_r = y$; set the RESTORABLE tag to TRUE, which means an iteration is saved and any other iteration can not be saved.
2) IF IN_RESTORING = FALSE (it means it's NOT a restored iteration) OR *t* and *y* NOT equal restore point t_r and y_r, THEN:
CHECK is there a close loop between *t* and *y*?
IF NO, SET: $d(t) = d(y) + l(yt)$, $P(t) = y$;

f. $S = S \cup \{t\}$; Go to sub-step c.