TOPOLOGICAL DESIGN OF MULTIPLE VIRTUAL PRIVATE NETWORKS UTILIZING SINK-TREE PATHS

by

Anotai Srikitja

B.S. in Computer Engineering, Chulalongkorn University, Thailand, 1994

M.S. in Telecommunications, University of Pittsburgh, 1996

Submitted to the Graduate Faculty of

the School of Information Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2004

UNIVERSITY OF PITTSBURGH

SCHOOL OF INFORMATION SCIENCES

This dissertation was presented

by

Anotai Srikitja

It was defended on

July 28, 2004

and approved by

David Tipper, PhD, Associate Professor

Richard Thompson, PhD, Professor

Bryan Norman, PhD, Associate Professor

Prashant Krishnamurthy, PhD, Associate Professor

Peter Steenkiste, PhD, Associate Professor

Dissertation Director: David Tipper, PhD, Associate Professor

TOPOLOGICAL DESIGN OF MULTIPLE VIRTUAL PRIVATE NETWORKS UTILIZING SINK-TREE PATHS

Anotai Srikitja, PhD

University of Pittsburgh, 2004

With the deployment of MultiProtocol Label Switching (MPLS) over a core backbone networks, it is possible for a service provider to built Virtual Private Networks (VPNs) supporting various classes of services with QoS guarantees. Efficiently mapping the logical layout of multiple VPNs over a service provider network is a challenging traffic engineering problem. The use of sink-tree (multipoint-to-point) routing paths in a MPLS network makes the VPN design problem different from traditional design approaches where a full-mesh of point-to-point paths is often the choice. The clear benefits of using sink-tree paths are the reduction in the number of label switch paths and bandwidth savings due to larger granularities of bandwidth aggregation within the network.

In this thesis, the design of multiple VPNs over a MPLS-like infrastructure network, using sink-tree routing, is formulated as a mixed integer programming problem to simultaneously find a set of VPN logical topologies and their dimensions to carry multi-service, multi-hour traffic from various customers. Such a problem formulation yields a NP-hard complexity. A heuristic path selection algorithm is proposed here to scale the VPN design problem by choosing a small-but-good candidate set of feasible sink-tree paths over which the optimal routes and capacity assignments are determined. The proposed heuristic has clearly shown to speed up the optimization process and the solution can be obtained within a reasonable time for a realistic-size network.

Nevertheless, when a large number of VPNs are being layout simultaneously, a standard optimization approach has a limited scalability. Here, the heuristics termed the Minimum-Capacity Sink-Tree Assignment (*MCSTA*) algorithm proposed to approximate the optimal bandwidth and sink-tree route assignment for multiple VPNs within a polynomial computational time. Numerical results demonstrate the *MCSTA* algorithm yields a good solution within a small error and sometimes yields the exact solution. Lastly, the proposed VPN design models and solution algorithms are extended for multipoint traffic demands including multipoint-to-point and broadcast connections.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I would like to dedicate this work to my dear mother in life and father in spirit who grant me the opportunity and always encourage me to fulfill my work. Their love is so pure and unconditional.

My deepest gratitude to my advisor, Dr. David Tipper, for his support, encouragement, invaluable guidance throughout the course of my study. His believe in my capacity had given me a strength to do better. This work cannot be completed without his understanding and continuously support.

My thanks to Dr. Richard Thompson who has always be like a father to me. He had given me the best opportunity to expand my horizon in work and in life. His kindness has deeply touched me in many ways.

I would also like to thank the rest of committee members−Dr. Prashant Krishnamurthy, Dr. Bryan Norman and Dr. Peter Steenkiste, for their comments, useful suggestions and for taking the time to make sure this work was a quality one. Many thanks to all staff members at SIS school who makes it a delightful experience while working there.

Thanks to all dear Thai friends who have make me feel more like home during my stay in Pittsburgh. They had given me lots of laugh and joy which are hard to find anywhere else. I personally feel overwhelmed by their generosity.

Last, I would like to pay my gratitude to Mr. Saran Maitreewet whom always gives me invaluable advices. Along the way, I have gradually learned to live a peaceful life and stay focus in a present moment. His guidance has awaken me spiritually and brought the best out of me.

1. Introduction

1.1. Introduction

Virtual Private Networks (VPNs) provide a private and dedicated environment over a shared private or public network infrastructure. In general, a VPN appears to its clients as if it is a dedicated private network, with exclusive use of the intermediate infrastructure, but in reality the shared resources are used among different types of traffic. VPNs were originally built to interconnect PBX and Centrex facilities oriented toward voice services in the framework of nation-wide networks [40]. With the advent of standard broadband technologies such as the asynchronous transfer mode (ATM) and frame relay, a deployment of VPNs supporting integrated service for voice, data and video applications together over a service provider's network appears to be economically appealing since it allows high-speed access with performance and Quality of Service (QoS) guarantees.

Another notable alternative is offering VPN services over the public Internet. The Internet is considered to be the most ubiquitous public data-centric network. During the past several years, there has been an overwhelming demand for broadband services driving the Internet to support almost all types of traffic. Public and private IP networks have promptly launched new mission critical and real-time applications which cannot tolerate unpredictable losses as well as require bandwidth and latency guarantee. Departing from existing "best effort" paradigm, the Next Generation Internet (NGI) aims to provide QoS guarantee such that a specific level of service performance can be assured. Two frameworks have been developed by the

Internet Engineering Task Force (IETF) community including the Integrated and Differentiated service models. However, the NGI has recently been geared toward the deployment of the Differentiated service (DiffServ) model and the Multiprotocol Label Switching (MPLS) packet forwarding technique, which together make it possible for QoS provision to be done in a scalable manner in terms of better manageability of bandwidth granularities and connection types. Based on these frameworks, the QoS-based VPNs can be built on top of an NGI infrastructure to offer different QoS guarantee to user-applications. A draft standard for VPN services over the Internet has been developed by the IETF organization [11, 35].

1.2. Next Generation Internet Architecture

The Multiprotocol Label Switching (MPLS) technique provides a connection-oriented, QoS-based approach to the NGI together with a traditional, connectionless, best-effort approach. MPLS uses the label swapping based forwarding similar to frame relay and ATM to handle IP traffic regardless of lower layer technology. It specifies functionalities for the efficient designation, routing, forwarding, and switching of IP traffic flows through the network. MPLS is an ongoing solution developed for the core IP backbone network. Considering that the demand for transmission bandwidth will be gigantic in upcoming years, the core IP architecture is expected to run over the Optical Transport Network (OTN). Therefore, the bridging of IP-based technologies such as MPLS and those in OTN such as Dense Wavelength Division Multiplexing (DWDM) network or optical circuits of synchronous optical network (SONET) is necessary.

At the DWDM (Dense Wavelength Division Multiplexing) layer, multiprotocol lambda (or wavelength) switching (MPLambdaS) proposed in [7] will extend the label-switching concept to include wavelength-routed and switched lightpaths. Potentially, it will resolve the

electronic-switched bottleneck. With MPLambdaS, a logical topology (completing any-to-any connectivity) consists of wavelength paths will be built to carry IP packets on top of WDM physical network composing of optical crossconnect (OXC) or optical add-drop multiplexer (OADM). However, a large channel granularity of the wavelength capacities ranging from 10 Gbps to 40 Gbps may be too large for end-to-end traffic profiles. Another concern is scalability arising from building such connectivity is that a large number of unique wavelength channels are necessary. For this reason, Generalized MPLS (GMPLS) [9, 60] has been standardized for the next generation optical Internet, which supports multi-granularity of switching types including fiber-, waveband- and lambda-switching in the optical domain. In other words, GMPLS is equipped with the ability of tunneling MPLS label switched paths (LSPs) which provision optical flows with mutli-granularity and a mechanism of assigning generalized labels to bundled consecutive wavelengths into a waveband or a whole fiber. GMPLS also incorporates switching in a finer granularity where the label switched path can be time division multiplexed (TDM) optical circuits of synchronous optical network/digital hierarchy (SONET/SDH).

Overall, to this end, in the NGI architecture, it is very promising that the IP layer will be overlayed on top of MPLS/MPLambdaS/GMPLS switching layer which will again sit on top of underlying optical physical infrastructure. Once the infrastructure is in place, the network management problem is then to determine how to efficiently map various virtual/logical topologies on top of each other. This problem has been done in the past in circuit-switched, frame relay and ATM networks. Considered to be a relatively longer-term procedure, logical topology design will provision virtual networks and determine the optimal configuration in both route and capacity/granularity wise.

1.3. Problem Statement

In a traditional Internet, traffic is forwarded based on destination IP address along the shortest possible routes computed based on traffic-insensitive-link metrics which does not always make good use of available network resources. This shortcoming is overcome by deploying labeling-switched technologies, e.g. MPLS, where traffic routes can be provisioned for QoS assurance using an MPLS explicit-path routing feature where the routing/forwarding path can be fixed/predetermined. Under the NGI architecture, QoS-based VPNs can be efficiently built over carrier network where the VPN logical topology is composed of a set of explicit LSPs. In MPLS, an explicit LSP can be thought of as virtual trunks carrying aggregation of IP traffic like a Virtual Path Connections (VPC) in ATM network. In supporting QoS-based VPNs over the NGI infrastructure using MPLS, one main issue is to provide a performance guarantee to coexisting VPNs having different service classes and topologies. A prerequisite to delivering QoS service guarantees is to have a good network design with proper traffic engineering, otherwise the network must be over-dimensioned which is not a cost effective strategy and yields a low network utilization.

In general, a QoS-based VPN design must consider multiple performance metrics such as call blocking, packet loss and delay and other QoS measures at the traffic layer. Within an MPLS domain, one can create a sink-tree, that is a directed routing tree ending at one network node (an egress node), to route an aggregate traffic demand through it. This makes the design of QoS-based VPNs in the NGI different from those of circuit-switched networks and connection-oriented packet networks such as ATM where a point-to-point path is used. In other words, with MPLS, it is possible to create multiple logical sink-trees carrying traffic of multiple VPNs. Note that traffic of different VPNs with the same QoS requirement may be carried on the same routing

tree and may/may not share network bandwidth. A big question is how to construct a tree and how to incorporate it in the network design model.

The goal of this research is to propose a mathematical model and solution algorithm for QoS-based VPN design over a core NGI backbone network supporting MPLS. In this thesis, the VPN design problem is modeled as a mixed integer programming (InP) problem assuming a linear capacity cost structure. The objective of the InP model is to minimize the cost of laying out a VPN supporting different traffic types and service classes on a given topology while meeting QoS requirements. Realizing sink-tree routing paths, the proposed model aims to find an optimal layout for multiple VPNs supporting multi-service classes for different time periods (multi-hour periods) considering that the traffic demand may vary during the course of the day. Bandwidth aggregation at different levels will also be considered in the link capacity allocation. The followings are assumed to be known including a physical topology and its bound on the maximum link capacity, a capacity cost, a traffic demand matrix of all VPNs and their QoS profiles.

The VPN design utilizing sink-tree routing paths turns out to be NP-complete problem. One cannot expect to find a time efficient algorithm to obtain the exact solution to the problem. Several search methods have been attempted in the past for similar InP problems including branch and bound, greedy heuristics, simulated annealing, and genetic algorithm techniques. The brand and bound technique has reported to give an exact solution but in general is too time consuming. The greedy heuristics can be fast but may easy get stacked in local minima which results in far-from-optimal solution. The main body of this thesis then aims to find an efficient solution method to the multiple VPN design problem which can be applicable for large networks of realistic sizes.

Major contributions of this dissertation are as follows

- The design of multiple VPNs over MPLS-like infrastructure network, using sink-tree routing, is formulated as a mixed integer programming problem to simultaneously find VPNs logical topologies and their dimension to carry multi-service, multi-hour VPNs traffic.

- A heuristic path selection algorithm is proposed to scale the VPN design problem by choosing a small-but-good candidate set of feasible sink-tree paths to solve the optimization problem over. The proposed heuristics introduces a new selection criteria which limiting a number of links used in a tree other than a number of hops used in a point-to-point path selection.

- Minimum-Capacity Sink-Tree Assignment (*MCSTA*) algorithm is devised to approximate optimal bandwidth and sink-tree route assignment for multiple VPNS in a polynomial computational time.

- Variations of *THE MCSTA* algorithm are considered including *MCSTA_APL*, *MCSTA_SPL* and *MCSTA_DPL* algorithms which employ a resource utilization index as criteria in a group demand ordering.

- The VPN design model is extended for multipoint traffic demands including multipoint-to-point and point-to-multipoint connections. Multiple broadcast trees as well as a single broadcast tree are considered for point-to-multipoint connections.

1.4. Organization

The remainder of this thesis is organized as follows. Chapter 2 will develop a background on VPNs in general to their requirements and functionalities of each component, then MPLS will be reviewed and explained in detail on how QoS-based VPN can be built over NGI backbone through the use of MPLS. Various aspect related to QoS-based VPN design over NGI architecture will also be thoroughly discussed. Chapter 3 presents previous work on virtual network design for circuit-switched and ATM networks. This is to give a review on a traditional design model for a virtual network used in the literature, since, from a design perspective, the concept of virtual networks in general can be applied to VPNs. Chapter 4 discusses the proposed design model for QoS-based VPN over an MPLS network. A detailed description will be given of the MPLS VPN design issues, that illustrate how they differ from a traditional design model, and discuss the design criteria used to develop the model. Then, mathematical models are proposed. Chapter 5 introduces a path selection heuristics aiming to scale the VPN design problem by choosing a small-but-good candidate set of feasible sink-tree paths. Chapter 6 will highlight the efficient solution algorithm called Minimum-Capacity Sink-Tree Assignment (*MCSTA*) devised to solve the problem of the multiple VPNs design. Performance of the solution algorithm will be thoroughly illustrated by comparing with branch and bound algorithm. Then, Chapter 7 shows extended VPNs design model and approximation algorithms for multipoint traffic including multipoint-to-point and broadcast connections. Finally, chapter 8 will conclude the thesis and address future works.

2. VPN Background

This chapter aims to provide a general perspective on VPNs in terms of service requirements and frameworks. The following sections will give a background on MPLS and how it will be used to facilitate the implementation of VPN services in NGI. Then, the main issues relating to the design of QoS-based VPNs over the NGI supporting MPLS are discussed.

2.1. VPN Overview

More specifically in this context, the term "Virtual Private Network" will be used to refer to a class of services that use a shared network infrastructure to emulate the characteristics of a private network. These characteristics can be expressed in terms of performance, reliability, security and quality of service. Note that many VPNs will typically co-exist over the same network infrastructure.

2.1.1. Requirements of VPN Services

The main attribute of a VPN service is a resource guarantee (i.e. network bandwidth and buffer space) offered by a service provider to emulate the characteristics of a dedicated network span over its customer sites. The VPN may be required to support voice and video as well as data applications in an economic fashion. In addition, for a customer whose sites locate on different operator domains, interoperability and service integration are desirable. In general, the main requirements of a VPN service can be grouped into four categories: economic, management, performance, and security.

(a) Economic Perspective

The main incentive for implementing VPN service is its economic benefits. Cost savings are realized through efficient bandwidth management, service integration and network management costs. Implementing and maintaining private networks, which often have complex topologies, are expensive because it involves expensive leased lines, long-distance dial-up, and switched services. Besides the cost of expensive equipment, operational costs can be saved by outsourcing the management and maintenance equipments to a service provider. This implies that the network design and management responsibilities are the VPN service provider's obligations. The service provider must allocate bandwidth to multiple VPNs on its infrastructure. This must be done in such a way that its revenue is maximized as well as meeting user-traffic profile criteria. Additionally, VPN services may allow companies to pay only for actual usage with no idle line.

(b) Management Perspective

At present, the increasing number of dispersed remote offices and mobility of workers makes private networks management difficult because individual networks may use different protocols, different applications, different carriers and different network management systems. This may involve both wire-line and wire-less network data services. Moreover, interfacing between two VPNs can be difficult. For some customers, interoperability and compatibility are key criteria for VPN services. Additionally, VPN services must provide customers with simple and homogeneous management of various end-to-end connections. More specifically, on-demand end-to-end connections should be available.

(c) Performance Perspective

Real-time and multimedia applications such as voice and video by nature require a certain level of performance (i.e., bandwidth and latency) and QoS guarantees. VPN services should allow user access to high bit-rate bandwidth and support variable-bit-rate traffic where a fluctuation in peak demand is expected. A minimum set of resources should be allocated to assure a given QoS level to a customer. A VPN should assure bandwidth and QoS level similar to that of a dedicated network. Additionally, for critical applications that rely on network availability such as military and financial services, in case of a network failure, a network recovery/restoration scheme must be implemented (e.g., through a traffic rerouting) to protect the VPN services from being disconnected.

(d) Security Perspective

Another important requirement for VPN services is a security provision. A VPN should provide a level of security similar to that of real private network. It must guarantee to keep transmitted data strictly confidential (e.g., through data encryption processes) as well as to prevent unauthorized users from gaining access to network-attached resources (e.g., through user-validation processes) or VPN resources.

2.1.2. VPN framework

From a design perspective, the concept of a virtual network in general can be applied to VPNs. The notion of virtual networks has long been used to refer to a logical network layout over several network architectures including circuit-switched and ATM networks. In [26, 27], the common generic definition of a virtual network is explained based on the potential

applications. Three broad categories can be classified namely: service oriented, user oriented, and management oriented virtual networks.

In a service oriented virtual network, separated virtual networks can be implemented for different service classes having different QoS requirements over the same physical infrastructure to simplify QoS management. Generally, the bandwidth allocation/dimension of various virtual networks should provide sufficient Grade-of-Service (GoS) (e.g., connection blocking probability) and fairness to different service classes.

In a user oriented virtual network, a separate virtual network is created for a group of users who have specific requirements (e.g. guaranteed throughput, customized control algorithms, resource management under user control, increased security and reliability, group tariff, etc.). The potential applications in this category are VPNs and multi-point connections.

Management oriented virtual networks are created to facilitate management functions such as fault-management and call admission control functions. For example, a separate backup virtual network can be created to provide protection against a network failure such as network node and link failures. Similarly, a separate virtual network can be built for signalling functions.

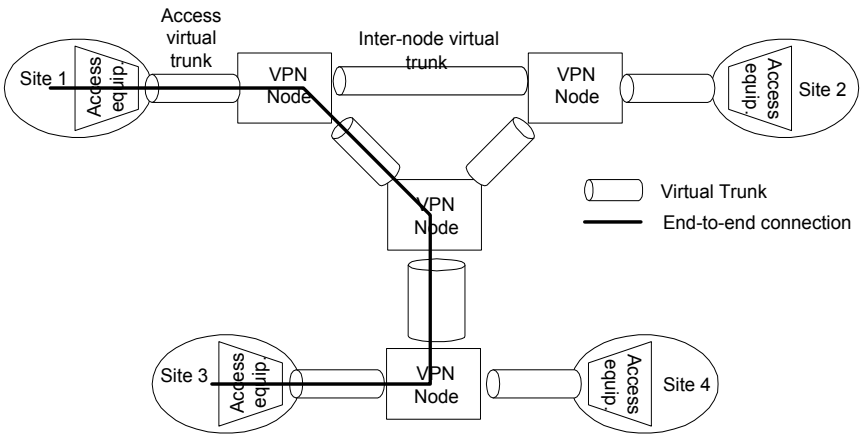


Figure 2.1: Architecture of a VPN

2.1.3. VPN Components

A VPN is composed of three main components namely: virtual trunks, VPN nodes and access nodes [40] (see Figure 2.1). Here, the term virtual trunk refers to a logical transport entity or a logical link between VPN network elements. It should be noted that a virtual trunk may be composed of one or more lower layer links (i.e. physical network links). There are two types of virtual trunks, namely access virtual trunks and inter-nodal virtual trunks. An access virtual trunk is a logical link between an access node and a VPN node. An inter-nodal virtual trunk is a logical link between the VPN nodes. An access node is the point where traffic from customer sites is multiplexed and injected into the VPN. The access node is sometimes referred to as an ingress node. VPN nodes are where virtual trunks are terminated or originated within the network. It could be a switched or cross-connect point of an end-to-end connection.

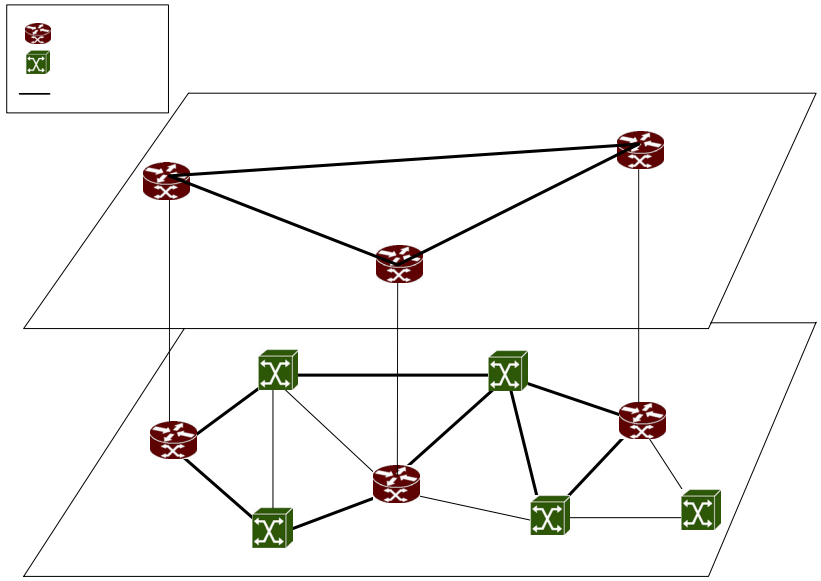


Figure 2.2: Overlay of VPN over a service provider network

Figure 2.2 shows an example of an overlay VPN on top of a service provider network. A node in the overlay VPN represents a VPN node and a link is a logical VPN link or VPN virtual trunk which consists of one or more link(s) in a service provider network. As shown in this example, the logical VPN link A and (B, C) consists of two and three links, respectively, in the lower-layer network. An actual path taken by a virtual trunk, over a service provider network, can be changed or adjusted by network management depending on how the network is designed and engineered.

2.2. MPLS Background

Multiprotocol Label Switching (MPLS) technology simplifies the packet forwarding function at the core of an IP network using a connection-oriented mechanism inside the connectionless IP networks. With MPLS, the standard destination-based hop-by-hop forwarding paradigm is replaced with a label-swapping forwarding paradigm, which is based on a simple short-label exact match. Thus, a packet can be forwarded at a very fast speed.

2.2.1. MPLS components

The MPLS network architecture consists of MPLS-capable routers, called label-switching routers (LSRs), in the MPLS domain, and MPLS edge routers at the edge where packets enter and exit the MPLS domain, called MPLS ingress and egress routers respectively. The MPLS ingress router will map the packet, as it enters the MPLS domain, to a corresponding forwarding equivalence class (FEC) and a label-switched path (LSP) based on the IP header. Figure 2.3 shows an example of packet forwarding in MPLS service provider network. Typically, traffic from customers originated from a customer edge (CE) router will enters and exits the MPLS domain at a provider edge (PE) routers (or LSRs). As packets from CE router-A, which are

destined to CE router-C, enter the MPLS network, PE router will assign them with the FEC-1. The packets will be forwarded in the same manner to the exiting PE router that connects to a destined router (CE router-C, in this example). The LSP is the path through which packets of a particular FEC will be forwarded and is set up for each route through the MPLS network. Then, based on a corresponding LSP identifier, a short fixed-length label is assigned as the packet is forwarded to the next hop. Note that a label is treated as a local significant identifier, thus, label swapping is done at the LSR as a packet is forwarded through the MPLS network. Consequently, the packet is simply forwarded along the LSP using only the label assigned to it.

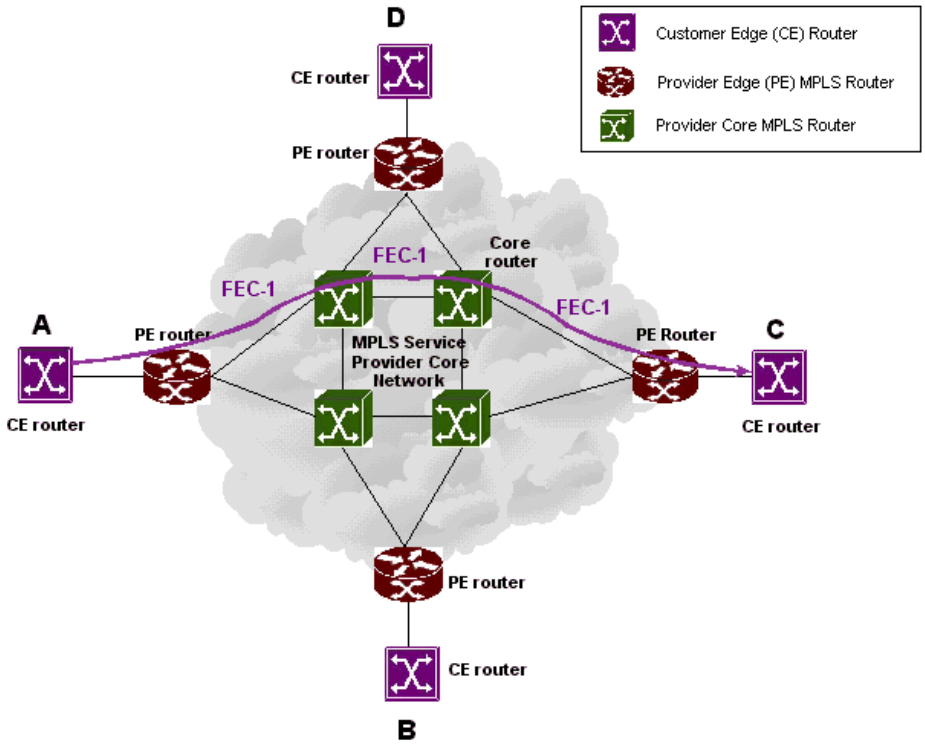


Figure 2.3: Mapping packets into a FEC forwarding class.

2.2.2. Forward Equivalent Class

Forward Equivalent Class (FEC) is defined in the MPLS standard as a group of packets which is forwarded in the same manner (e.g., over the same path, with the same forwarding treatment). FEC can be used to classify packets in different granularities. For example, based on an address prefix classification, FEC can be associated with packets going to a particular destination or, in conjunction with different types of services, it can be associated with packets to a destination with a distinct service class.

If a FEC is created to identify all packets going through the same egress LSR, packets from different ingress LSRs in the same FEC will be forwarded through the same LSP structure exiting at a particular egress LSR. The LSP in this case is a multipoint-to-point path forming a logical sink-tree structure. Traffic from different sources exiting at the same egress node can be aggregated within the MPLS network. Different levels of traffic aggregation are possible in the MPLS domain. Through traffic aggregation, the total number of labels that need to be distributed and managed can be reduced, thus making MPLS scalable and manageable for a backbone network.

2.2.3. Label Assignment and Binding

Each LSR along the LSP must negotiate a label that will be used for each FEC with its upstream and downstream neighbors. A downstream label assignment is used by default where the downstream LSR assigns a label for each FEC and the label binding information is distributed to its upstream LSR. LSRs will use a Label Distribution Protocol (LDP) to exchange label/FEC binding information. Different label distribution protocols, e.g. RSVP-TE and CR-LDP [3, 8, 44, 73], might be used for different purposes or in various circumstances [75].

With label/FEC binding information, each LSR creates a forwarding information base (FIB), a database that contains the label mapping of its incoming and outgoing links for each FEC. The LSR will use a label of an incoming packet as an index to the FIB to find the outgoing link and if a new label is needed for the packet. The label/FEC binding must be renegotiated when needed or dynamically (for example, when a forwarding path is changed) and the FIB must be updated accordingly.

2.2.4. Route Selection in MPLS

A label-switched path (LSP), a path that is taken by packets within the same FEC, can be selected based on different routing algorithms. For a particular FEC, the route selection may be based on a hop-by-hop routing or an explicit routing. In a hop-by-hop routing, each LSR independently choose the next hop for each FEC. A standard routing protocol used in existing IP networks such as RIP (Routing Information Protocol) and OSPF (Open Shortest Path First) can be used to select the next hop. However, to support different service requirements, a LSP may be established on a route different from a shortest path. Therefore, the information on the service-related properties of each link must be known and used at each router to calculate an optimal route for each FEC.

In an explicit routing [6], a single LSR, generally an ingress LSR or an egress LSR, will select the route to establish a LSP. For example, an egress LSR may compute an entire path for a tree ending at it. The explicit routing capability facilitates QoS service support and traffic engineering over MPLS networks. With explicit routing, each traffic stream between an ingress and egress node-pair can be individually routed through a preferred path. Thus, it allows a path that can satisfy the QoS requirement of a traffic stream (e.g., a path with enough bandwidth) to be explicitly identified. Note that the QoS profile of a traffic stream may be specified in

experimental bits of the MPLS label attached to a packet. As a packet is forwarded along a predetermined LSP, it can be treated in an appropriate manner such that its QoS requirement is maintained. Using an explicit routing mechanism, a set of defined paths may be set up through the MPLS network for certain traffic streams for various traffic engineering purposes. For example, one path may be preferred over another to allow load-balancing across multiple links. In general, traffic engineering can be implemented over an MPLS network to support load balancing, dynamic fall over to backup paths, or re-optimization of traffic routes for better network resource utilization and transport performance.

2.3. QoS-based VPN over NGI

Over a IP backbone network, building QoS-VPNs can be easily deployed with MPLS. Multiple VPNs can be constructed on the same network through the use of different MPLS Forward Equivalent Classes (FECs). As mentioned earlier, FEC is used to define traffic that will be forward in the same manner through a MPLS network. Thus, different FECs could be used to classify traffic from different VPNs which may or may not use the same forwarding path and may or may not share a portion of network bandwidth. One example, shown in Figure 2.4, is to define different FECs for VPNs traffic destined to different egress routers. VPN-1 traffic exiting at the same egress PE router is assigned to FEC-1 while packets of VPN-2 are assigned to FEC-2. In this case, a label switched paths (LSPs) of FEC-2 forms a directed inverted tree rooted at the egress node (or sink-tree). In allocating network bandwidth to VPNs, note that, the bandwidth of a path segment after a merger point can be shared by traffic within one VPN. In other words, traffic multiplexing can be achieved naturally. Here, the term bandwidth

aggregation will be used interchangeably with bandwidth sharing and statistical bandwidth multiplexing throughout the paper.

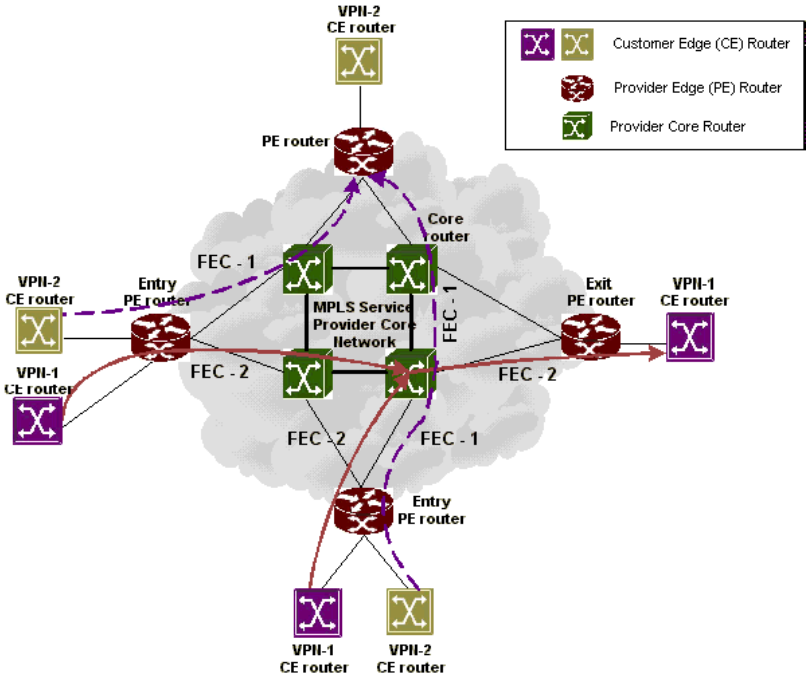


Figure 2.4: QoS-VPN implementation over MPLS core network.

2.4. VPN Design Issues over NGI

The major challenges in deploying QoS-based VPNs over the Internet are delivering performance guarantee and security assurance to a degree that is comparable to a real private network. Deploying MPLS over an IP network makes it easier for VPN services to provide performance at the required level. Since MPLS is designed to have a traffic engineering capability, provisioning for QoS guarantees to traffic in different classes of service are possible. In terms of security, the goal is to protect VPN data from maliciously or accidentally misconduct. The IP Security Protocol (IPSec) [49] was developed for this purpose.

Assuring performance of VPN services, a service provider has to be concerned with capacity provisioning and routing coexisting VPNs having different service classes and

topologies over the same network infrastructure. Furthermore, in VPN design, one must be concerned with scalability and manageability in order to be able to support a large number of customers. In another words, a well-designed VPN must be easy to manage and at the same time attain bandwidth efficiency. Over a MPLS network, this means that the number of label switched paths (LSPs) and required labels must be kept small. In term of capacity efficiency, different levels of traffic aggregation may be considered. For example, one may consider aggregation of traffic from different VPNs belonging to the same service class or aggregation of traffic from VPNs exiting at the same egress node. Finally, another concern a VPN designer must deal with is fault management. A VPN must be equipped with traffic restoration functions and an adequate amount of spare capacity to protect VPN traffic in case of a network failure.

In summary, a prerequisite to delivering QoS service guarantees is to have a good network design with proper traffic engineering, otherwise the network must be largely over-provisioned which is not a cost effective strategy and yields a low network utilization. In doing so, a matrix of traffic demands that must be handled by a network must first be determined. This can be computed from all SLAs (Service Level Agreements), between customers and service providers, which identify various classes of service and specify how much traffic in each service class a user is allowed to send. Then, the VPN management system can run the network engineering/design algorithms to determine how traffic should be efficiently mapped onto an existing network topology while maintaining an optimal use of network resources.

In general, a QoS-based VPN design must consider multiple performance metrics such as call blocking, packet loss and delay and other QoS measures at the traffic layer. One unique aspect of MPLS is that a packet forwarding path can be a sink-tree path, that is a directed routing tree ending at one network node (an exit/outgoing node). This makes the problem of QoS-based

VPN design in the NGI different from those of circuit-switched networks and connection-oriented packet networks such as ATM. In other words, with MPLS, it is possible to create multiple logical sink-trees carrying traffic of multiple VPNs. Note that, the traffic of different VPNs with the same QoS requirement may be carried on the same routing tree and may/may not share network bandwidth. A fundamental problem is how to construct a tree and how to incorporate it in the network design model.

In MPLS, two types of traffic connections must be supported, and thus to be considered in a design perspective, including point-to-point and multipoint connections. Multipoint connections include broadcast (point-to-multipoint), merge (multipoint-to-point), composite (point-to multipoint and multipoint-to-point) and full multipoint (multipoint-to-multipoint). In addition to the traffic type, different QoS classes should be considered. Two frameworks are proposed by the IETF to be used for NGI namely the Integrated and Differentiated service models. In the Differentiated service model, for example, QoS classes are composed of premium service, assured service and best-effort service. In the premium service class, packet loss, delay and delay jitter must be bounded. The traffic of this class requires an absolute bandwidth guarantee. In the assured service class applications have the ability to tolerate a certain amount of delay and loss. For this traffic class, a mean bandwidth guarantee is needed along with a statistical delay bound. The best-effort service class corresponds to current Internet service with no QoS guarantees. Another aspect that should be addressed in a QoS-based VPN design model is the probability of call return. In certain service classes, many end-user applications such as Web-based applications or voice telephony applications tend to have an automatic call retry or repeat attempt feature. When a requested call/connection is first rejected, there is a certain

probability that the rejected call/connection will be retried. The portion of call return traffic will, therefore, increase the amount of traffic offered to the VPN.

3. Virtual Network Design

VPNs over a circuit-switched or an ATM network, are often viewed as a logical mesh network with point-to-point demands between node pairs. A logical full-mesh is a topology where each point-to-point demand pair is independently given a logical link that may be routed over multiple switched-points. A switched-point is a cross-connect switch in a circuit-switch network and an ATM switch in an ATM network. VPN design over a circuit-switched network differs from an ATM network in several ways. First, in a circuit-switched network, transmission subsystems in different levels are related in a hierarchy, while, in an ATM network, there is no such hierarchical structure. However, a non-hierarchical network design can be applied over a circuit-switched network operating with non-hierarchical routing. Another difference addressed in VPN design is capacity structure. In a circuit-switched network, link capacity is calculated in a unit derived from a unit of a trunk capacity in a lower hierarchy, whereas, in an ATM network, link capacity can be varied in a finer granularity.

This chapter first presents a summary on circuit-switched network VPN design. Next, ATM network VPN design will be discussed in greater detail since its characteristics and design criteria are somewhat similar to VPN design over a NGI network. Lastly, works on MPLS-based VPN design are then reviewed.

3.1. VPN Design over Circuit-Switched Network

For circuit-switched networks, a large body of literature exists on dimensioning hierarchical and non-hierarchical networks. Both single-hour and multi-hour models are considered. Most of the literature formulates the dimensioning problem for circuit-switched networks as a non-linear optimization problem. Several solution approaches are proposed by iterating between solving a set of non-linear equations and a solution of linear programs [5, 33].

For a traditional hierarchical telephone network, a trunk dimensioning is usually done to minimize the total network cost while meeting the grade-of-service requirements which can be stated in terms of a blocking probability on the final trunk groups. The papers [70-72, 80] considered the optimal dimensioning of a hierarchical network for a single traffic demand matrix over a single-hour period. Since the blocking probability on the final trunk groups does not represent the actual grade of service as perceived by the users, Beshai and Horn [13] proposed a dimensioning algorithm for hierarchical networks under end-to-end blocking constraints. In the multi-hour network dimensioning problem [5], the optimization model takes into account the fact that traffic flow demand can vary substantially during the day and thus yields different periods of peak demand between any node-pair in the network. Compared to dimensioning a network based on the maximum busy hour, the multi-hour network design can significantly reduce network cost. The multi-hour dimensioning problem can be reduced to a single problem, or a sequence of single-hour problem [5]. Using a technique of non-differentiable optimization, a solution approach to the mathematical model, for two-level networks, with a linear cost objective function, is proposed by [28-30]. For large networks with more complex hierarchies, this approach is not considered efficient.

Girard, Lansard and Liau applied an LP relaxation technique to solve a dual problem of the multi-hour dimensioning model.

The dimensioning problem in a non-hierarchical network is different from one in a hierarchical network since user traffic is represented as a flow that is dynamically routed through the network. Secondly, a grade of service constraint cannot be directly expressed as a link blocking probability but is given in term of an end-to-end blocking probability. In addition, there is no difference between high usage and final trunk groups. Katz [46] dimensioned nonhierarchical networks by trying to find a set of feasible flows in which the end-to-end blocking probabilities are equalized as much as possible. The model assumes that the network cost is minimized subjected to uniform grade-of-service constraints. Dimensioning a nonhierarchical network is modeled as a multicommodity flow problem in [5, 12, 52]. The flow is optimally routed over an uncapacitated network to minimize the capacity cost of carrying the flow on links. The cost function is shown to be a complex nonlinear function of link capacities with linear flow constraints.

In Berry's work [12], the traffic demand is divided into the first-offered and overflow traffic that is carried by two different trunk groups. One well-known approach in dimensioning nonhierarchical networks is the Unified Algorithm (UA), a heuristic method, developed for dynamic nonhierarchical routing (DNHR), and used in AT&T network design [5]. The Unified Algorithm is used for both trunk-dimensioning and routing optimization based on a multi-hour traffic engineering approach. The heuristics are devised to replace non-linear variables and thus make UA computationally efficient for a large network. Several papers [16, 41] have studied dimensioning of adaptive routing networks based on residual

capacity. The heuristics is proposed for a multi-hour optimization with fixed size of trunk group.

The circuit-switched network design work discussed above only considers traffic having point-to-point connections in single- and multi-rate connections including [33, 48]. In [61], the authors address circuit-switched network design with multipoint connections, particularly point-to-multipoint connections. The study was done to investigate the impact of tree selection on the optimal routing and network dimensioning problem in the presence of point-to-multipoint traffic. Their results suggest that to achieve route optimality, it is advisable to spread traffic evenly among trees with few links in common. On the other hand, for the optimal dimensioning, one should concentrate traffic on few trees having a small number of links with a large capacity. Applying the above results to a network design problem, a heuristics method was proposed by same authors in [62]. This is done in two stages. The first stage is to maximize traffic concentration by solving a dimensioning problem using a set of maximum spanning tree paths. The second stage is to try to move some point-to-point traffic that is routed over costly and long paths to more direct links.

3.2. VPN Design over ATM

ATM is a cell-based switched transport technology designed to support various services on broadband networks. The service classes, with different traffic characteristics, different qualities of service, and different bandwidth requirements and holding times, makes VPN design over ATM different from circuit-switched networks. To assure that the network resources are enough to provide different QoS guarantees, appropriate traffic engineering mechanisms are used in conjunction with a connection admission control (CAC) scheme.

Among other things, the first has to do with a network capacity allocation to ensure that a network is equipped with enough capacity to support a certain traffic matrix. The CAC is used to limit access to the network such that network resources are kept below an appropriate level such that QoS commitment could be given to all traffic connections admitted to the network.

In ATM networks, multiple Virtual Channels (VCs) of the same characteristics can be grouped together and placed inside pre-established Virtual Path (VPs). The call setup time is reduced if there exists a virtual path with enough capacity between a requested node-pair since no extra processing time at the intermediate nodes is required. This makes it easier for network management and control since it can be done at the VP level. The statistical multiplexing of VCs going through the same VP is possible and will give a better resource utilization. The role of virtual paths in ATM networks has been investigated in [1, 2, 14, 15, 25, 47, 59, 79].

3.2.1. ATM VPN Design

ATM VPN design mainly focuses on how a virtual path connection (VPC) could be efficiently layout on top of a physical network. The VPC overlay network or the logical network is where virtual channel connections will be routed. Determining the VPC route configuration and assigning VPC capacity are related and must be done together to achieve the optimum cost effective design. This can be formulated as an optimization problem which is usually large and complex since the number of possible VPC topologies typically grows exponentially with the network size. Moreover, VPC capacity allocation is itself a complex

problem due to different numbers and sizes of VCCs that can be efficiently packed into each VPC to fully utilize the network link capacity.

In the literature, a design of a virtual network over ATM can be categorized into three possible architectures [26, 27, 40] based on how a "Virtual Network Link" (VNL) or "virtual trunk" is constructed. This is shown in Figure 3.1 where a VNL link may be composed of a single VP or multiple VPs and an end-to-end traffic is carried via VC.

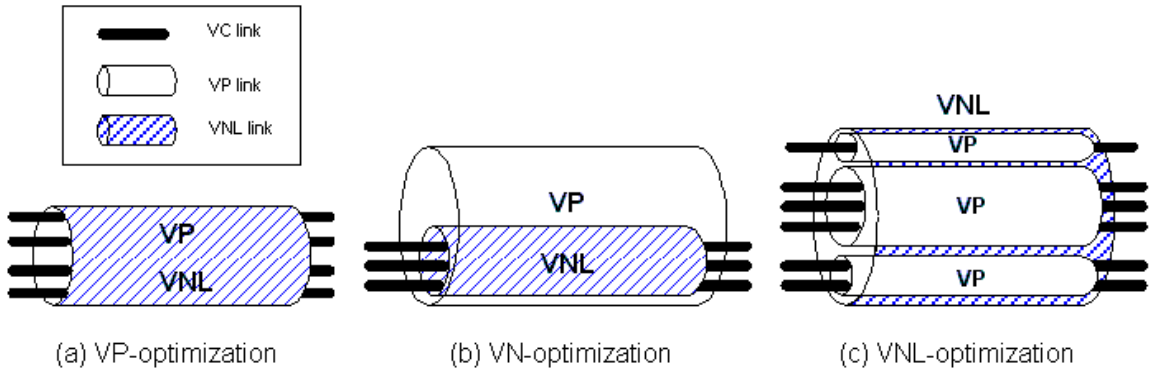


Figure 3.1: ATM-VPN link structure

a) Single-hop VP (SHVP) architecture

In a SHVP architecture, a VNL is composed of a single VP, and, for every traffic demand pair, there is a direct VNL/VP link between them. The end-to-end connections multiplexed through a VC traverses only one VP-hop and there is no transit node. Figure 3.2 (b) shows an example of a SHVP design of 5-node network, shown in Figure 3.2 (a), with four switching nodes and one cross-connect node. A switching node is where traffic joins and leaves a network, and cross-connect node serves as a transit point of traffic inside a network. It is shown in Figure 3.2(b), in SHVP design, there are total of 6 direct VP/VNL links upon

which VC connections carrying demand traffic are multiplexed. In this case, the VNL/VP links represent bi-directional links.

SHVP architecture is sometimes referred to as VP-mesh design where a VP route determines the optimal routing path. The VP capacity assignment is done based on a given traffic demand matrix. Since statistical multiplexing among multiple VCs going through different VPs is not possible, the total network bandwidth may not be optimal with this architecture. However, the connection setup time is reduced and an admission control procedure is simplified since each node appears to be one-hop apart from each other.

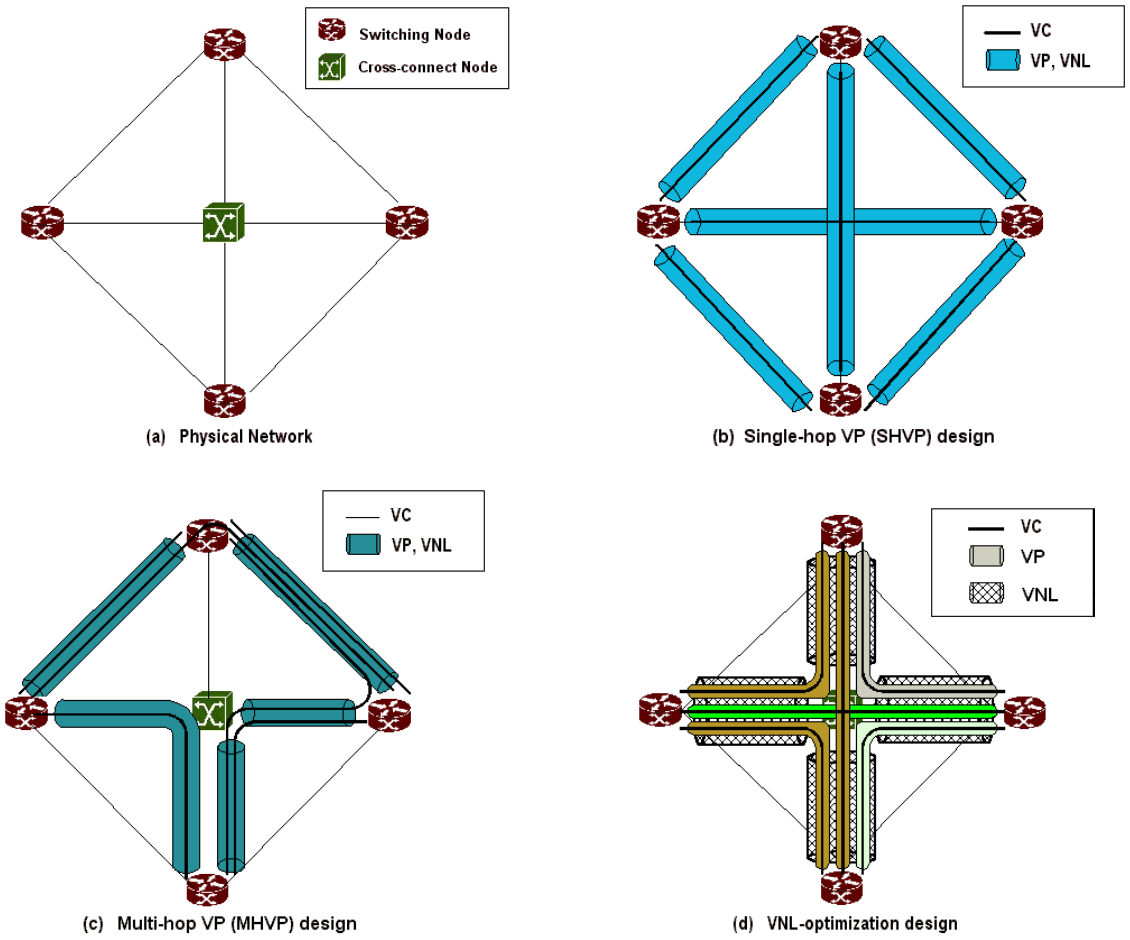


Figure 3.2: ATM VPN Design

b) Multi-hop VP (MHVP) architecture

In a MHVP architecture, a VNL is composed of a single VP but a VC carrying aggregated traffic between each node pair may traverse one or multiple VP-hops. Hence, the VP capacity and route assignment optimization is done to attain a minimum total VN bandwidth cost for a given traffic demand matrix subject to Grade of Service (GoS) constraints (such as connection blocking probabilities). In an example of a MHVP design, shown in Figure 3.2 (c), there are 5 VP/VNL links and multiple VCs carrying traffic that traverses through them (in single or multiple VP-hop(s). Compared to the VP-mesh design, the required network bandwidth may be lower since statistical multiplexing among multiple VCs going through different VPs can be achieved. Nevertheless, VCs switching cost is introduced and the connection set-up time may be longer. Thus, in this case, a bandwidth cost must be trade off with a switching cost.

c) VNL-optimization architecture

In a VNL-optimization architecture, there is a direct VP between every traffic demand pair. A VC will go through one VP-hop but a VP may traverse many VNLs. Each VNL may carry multiple VPs. Thus, the VP-mesh design is realized as in the SHVP architecture. However, in this case, the bandwidth and route optimization is done for each VNL. An example is shown in Figure 3.2(d) where four VNL links are created to carry multiple-direct VPs/VCs carrying demand traffic between the four switching nodes.

An important implication of the above architectures is that, in the SHVP and MHVP architecture, the core ATM backbone switching network is considered, while, in the VNL-optimization architecture, the ATM switches are located only at the edge of the lower level cross-connect network where the bandwidth of VNL links can be managed within the

network and outside of the ATM domain. Most of the literature related to ATM VPN design considers either single-hop or multi-hop VP design cases.

Compared to circuit-switched networks, dimensioning ATM networks is more difficult since multiple QoS measures for different service classes, such as cell loss and delay, must be considered in addition to call blocking and routing constraints. The concept of "effective bandwidth" [27, 56, 77] was introduced to represent service rates required by different VCCs of different service classes. Using the effective bandwidth concept, traffic flows with different characteristics and QoS requirements can be represented as being steady with a deterministic bandwidth requirement. This largely simplifies the optimization model for ATM VPN design problem. ATM traffic demand at a call level is often modeled with Poisson arrival process. Thus, call-level traffic in an ATM network can be viewed as circuit switched multi-rate traffic [57] except that the ATM bandwidth can be arbitrary and does not derive from a basic bandwidth unit as in circuit-switched network. It was noted in [58] that classes with bandwidths in very different magnitudes should not have the same call blocking probabilities.

3.2.2. ATM Virtual Path Network Design

In logical Virtual Path (VP) based VPN design, previous work has been appeared on dimensioning SHVP (Single-hop VP) networks [22, 32, 36, 37, 42, 65, 66, 82] and MHVP (Multi-hop VP) networks [2, 4, 10, 19, 20, 50, 55, 67]. In SHVP network design, a separate VP is established for each end-to-end traffic demand. Every source and destination pair is viewed as being one-hop apart or, in another words, a logical full meshed of VPs is realized. Hence, each VC will only go through one VP and the end-to-end traffic streams are not

allowed to mix. A MHVP network design aims to optimize the total network bandwidth where VPs can be established between any node-pair other than a traffic demand node-pair. Therefore, VCs can be routed over multiple VPs. Noted that SHVP is a special case of MHVP. A major trade off with these two approaches is a switching cost versus a capacity cost. Typically, the SHVP approach realizes an operational simplicity with no VC/VP switching cost associated but yields a higher capacity cost. In contrast, MHVP design can achieve lower bandwidth utilization but has to consider VC/VP switching cost.

As in circuit-switched network design, ATM VP network design can be optimized for a single-hour period or multi-hour period. In the literature, most of the work on VP network design is for a single-hour period [2, 4, 17, 19-22, 31, 32, 36, 37, 50, 65-68, 76, 82]. Only a few papers have considered multi-hour VP network design including Medhi [64] and Bauschert [10].

a) Single-hour Single-hop VP Network Design

In the literature [32, 65, 82] studied the problem of designing a logical ATM network (or packet-switched network) on top of a circuit-switched network. A logical link between ATM switching nodes is called an "Expressed Pipe" [32]. A Digital Crossconnect System (DCS) system is assume to be used for dynamic reconfiguration and bandwidth allocations of the logical network. These papers develop a non-linear integer programming model to minimize the total average queueing delay. The average queueing delay is chosen as the indirect measure of buffer overflow probability to be minimized [32]. A Jackson network [51] (i.e., a network of independent M/M/1 queues) is assumed. [32] assumes a static, non-bifurcated routing of traffic where routes are defined at system generation for each communicating node-pair. In [65, 82], a route for each demand pair is chosen from a set of

precalculated candidate routes. Solution techniques based on Frank-Wolfe's steepest descent method are devised in [32, 82], while, in [65], the Lagrangian relaxation technique is used to obtain lower bounds on the delay.

In [36, 37], a logical VP network design problem is formulated as a nonlinear integer programming problem. The proposed model aims to minimize the average call blocking probability with a constraint on available physical network resources for a given traffic demand matrix. A heuristic algorithm based on the greedy principle is developed to solve the problem where the path set used in each iteration is precalculated. The weakness of the proposed heuristic is that it could not establish bounds on the solution.

Several works [22, 42, 66] address the VP route assignment problem over a physical topology of infinite capacity. A polynomial time algorithm is presented in [22] to approximate an optimal VP route assignment that minimizes the maximum link load. The formulation in [42] seeks to minimize the path bandwidth usage while satisfying link level Grade of Service (GoS) requirements. It is shown to over engineer the network and a lower bound cannot be obtained. In addition, the end-to-end GoS is not considered and can be higher than the expected link-level GoS.

In [66], the problem of VP logical network design is formulated in the framework of multi-rate, circuit-switched, loss networks. The underlining assumption is that, for a very large network link capacity, with the use of leaky bucket regulator, the effective bandwidth simply equals the mean rate. Offered traffic is carried on routes which maximized network revenue.

b) Single-hour Multi-hop VP Network Design

The joint topology, capacity and routing assignment for multi-hop VP networks is presented in [2, 4, 55]. Lee and Yee assumed a discrete value of logical link capacity and formulate the problem as a nonlinear mixed integer programming problem. The total cell loss rates of all VPs is to be minimized. An integrality relaxation and a round-off scheme are used in the computational process. Ahn et al. [2] solved this problem by dividing the problem into two subproblems: (i) traffic flow optimization, and (ii) the VP-layout generation. The authors modeled the first subproblem as a mixed 0-1 integer linear programming which is NP-hard. A heuristic approach is devised to find a VP-layout to satisfy connectivity, delay, and switching constraints as well as to minimize the switching and the call setup cost. This decomposition approach does not take into account the statistical multiplexing effect. Unlike [2, 55], Aneroussis and Lazar tried to maximize the total network revenue expressed as a weighted sum of carried demand. A combination of nodal constraints on the processing of signaling messages, and constraints on the VC blocking probability of each source-destination pair are considered.

The problems of virtual path dimensioning and virtual circuit routing assignment are jointly determined in the works by Cheng and Lin [16, 19, 20, 50]. Cheng and Lin [19, 20] formulated this problem as a non-linear nondifferentiable combinatorial optimization model to minimize the expected call blocking rate subject to call set-up time constraints for a given physical topology, physical link capacity and traffic demand matrix. In the formulation, the call setup time constraint is interpreted as the maximum number of physical links used by each VC. Cell loss requirements of VC connections are translated into an effective capacity requirement of the physical link, which is a function of a number of physical channels for a fixed bandwidth unit. In [20], a two-phase solution procedure is proposed to find a Pareto

optimum solution, which is computed by relaxing the integrality constraints and applying a greedy heuristic.

The trade-off between increased capacity costs and reduced control cost are shown in the work by Kim [50]. The author formulated the multicommodity integer flow model in which capacity cost, buffer and control/switching cost are to be minimized while meeting QoS and GoS requirements. The effective bandwidth used is based on Guerin's approximation [39] for heterogeneous ON-OFF source and is applied to account for the cell level QoS. The formulation is shown to be a non-linear integer optimization problem of the NP-hard type. The solution can be approximated by applying the exterior penalty function method to solve the non-linear problem by relaxing integer constraints.

Oki and Yamanaka [67] attempted to find the optimum logical topology for ATM networks. Link addition and elimination techniques are used iteratively to minimize the network cost while satisfying end-to-end QoS requirements including a call blocking rate, a cell loss rate and delay bound of different service classes. The VPs and VCs are always routed using the shortest physical- and VP- hops respectively. The result from the computational procedure yields a non-optimal solution, consisting of VP logical topology and its assigned capacity for a given physical topology with infinite capacity. Their results suggest that a mesh-type VP topology be used for non-bursty delay-sensitive traffic class while a less-connected VP topology (i.e. logical ring topology) should be used for highly bursty traffic class to achieve a capacity gain from a statistical multiplexing.

c) Multi-hour VP Network Design

Models and algorithms for multi-hour VP network design based on a SHVP approach are presented in [63, 64]. Separate VPs are established for different traffic classes and source-

destination pairs. Statistical multiplexing is allowed among traffic within the same VP. Different VPs passing through the same physical network link are deterministically multiplexed. In another words, each VP is assigned a bandwidth portion that is not shared by other VPs. A multi-hour solution is obtained by solving an optimization problem to determine the optimal VP layout and its dimension for each design hour separately.

The work by Bauschert [10] extends the Unified Algorithm (UA) [5] to solve the multi-hour ATM network design problem for MHVP network with nonhierarchical VC routing. The design problem is formulated as a cost minimization model where both transmission costs as well as VP/VC switching cost are considered. A decomposition method is applied to obtain the solution. For a given physical network, the problem is solved by first finding the minimum cost VP layout which is assumed to remain unchanged for each design hour. Then, the optimal VC routing for each hour period is determined separately to compensate for the load variations subjected to meet the end-to-end call blocking constraints.

3.3. MPLS Network Design

Work on VPN design over a MPLS network is rarely found in the literature. Only recently has few works appeared on formulating optimization models to solve traffic engineering problem in general over MPLS networks. In [34], authors established integer programming formulations for flow assignment problem given a set of point-to-point LSPs. The model is extended to solve a capacity planning problem. The authors points out many assumptions made in the proposed models for further improvement including: (i) one-to-one relationship between traffic trunks and LSPs, (ii) no aggregation, de-aggregation and merging of LSPs. (iii) one or more feasible solutions exist.

Topological design of a single VPN is formulated as a mixed-integer programming model [69]. Point-to-point paths are used for a set of source and destination node-pairs. The authors developed a heuristic algorithms based on the branch-and-bound approach to approximate the optimal solution. A linear relaxation is done for link-path formulation and the lower bound is obtained by routing all pair demands through the shortest paths.

The use of multipoint-to-point LSPs is proposed in [78]. The route selection, LSP creation and flow assignment problem are solved separately. The set of paths for traffic between all node-pairs are first determined then LSPs are created to include those paths. A constraint is used in LSPs selection to at least have two routes which do not share any single node to each ingress/egress node. LSP creation is modeled as 0-1 integer programming problem. Using pre-selected LSPs, another problem is formulated for flow assignment to minimize the maximum link load.

An integer programming formulation for the general VPN tree computation problem is proposed in [53] for the hose model where only the amount of incoming and outgoing traffic are declared for each VPN endpoints instead of all pair demands. In the model, the bandwidth reserved on a link must be equal to the maximum possible aggregate traffic between the two sets of endpoints connected by the link. Infinite network link capacity is assumed. A polynomial-time solution algorithm based on a breath-first search is developed for a special case when incoming and outgoing traffic for each VPN endpoint are equal. For arbitrary demand, an algorithm based on the primal-dual method is devised. The model is extended to the case when network links have capacity constraints in [54].

A mixed integer multiobjective programming formulation of the VPN design problem is proposed in [23]. The model aims to minimize both resource usage and link utilization to

achieve a bandwidth cost savings while load balancing traffic across the network. The model is shown to have NP-hard complexity, therefore, the author developed a continuous approximation algorithm to estimate the solution based on two main assumptions. First, the aggregated demand must be infinitely divisible. Second, the aggregated demand can be routed over multiple routes.

4. QoS-based VPN Design Model

This thesis proposes mathematical formulations for the problem of designing multiple VPNs over a service provider IP infrastructure supporting MPLS to carry multi-service, multi-hour traffic from various customers. Here we exploit pre-computed sink-tree paths (multipoint-to-point paths) over which VPN traffic is routed in a MPLS core network. In the model, different levels of bandwidth aggregation/multiplexing occur across different service classes and routes within one VPN, but not across different VPNs. It is clearly shown that the design problem formulations yield a NP-hard complexity. Therefore, numerical study is conducted for simple cases where only single-service, single-hour VPN traffic is considered. Obtaining the solution to this problem provides a benchmark measure and a guidance to solution feasibility.

4.1. VPN Design Methodology

The design of VPNs is a major part of the traffic engineering procedure which generally can be done offline to obtain VPNs optimal logical topology and virtual network link (VNL) dimension as illustrated in Figure 4.1. During an operational period, network management will monitor changes in traffic pattern, network topology or link cost metrics. When it notices any changes that will affect the current settings, it will start a global optimization procedure to perform an offline recomputation. Note that, the optimization procedure can be done separately for each VPN or jointly for all VPNs.

For QoS-based VPNs over MPLS, the proposed network design process aims to find the optimal logical sink-tree(s) and its dimension so as to minimize the total network cost while satisfying QoS constraints. Three main tasks are involved in the design process: (i) Tree generation/selection, (ii) Dimensioning and (iii) Routing Optimization. The first task involves generating a candidate set of logical trees for a given source and a set of destinations. The second task is to determine the amount of bandwidth allocated to each link in a tree whose bandwidth may/may not be shared by different VPNs traffic. The last task aims to find an optimal route assignment for a given traffic demand. In general, these three main tasks can be solved independently or jointly.

The transport network topology and node locations (e.g. locations of MPLS edge routers and core routers) are used to generate a set of candidate multipoint-to-point tree paths which is then reduced to a feasible set based on traffic QoS constraints. For instance, a bound on maximum delay can be translated into a maximum hop limitation. This precomputed set of trees is used in an optimization model over which the optimal routes and capacity requirements are determined. The bandwidth allocation/dimension of the virtual network should provide sufficient Grade-of-Service (GoS) (e.g., connection blocking probability) and fairness to different services while satisfying several performance constraints at the traffic layer including packet loss rate and delay. Here, the concept of "effective bandwidth" is exploited to represent a service rate required by each traffic connection in various service classes. Effective bandwidth calculation will encapsulate the QoS requirements in terms of packet loss rate and delay. Thus, by using the concept of effective bandwidth, traffic flows with different characteristics and QoS requirements can be represented as being steady with a deterministic bandwidth requirement. This simplifies our optimization model. Lastly, the

routing optimization will optimally assign a route to a traffic demand, given a set of candidate routes and link capacities. Other than minimizing cost of laying out a given traffic, a route assignment may also aim to balance the load across the network such that the number of over-utilized links and under-utilized link is reduced.

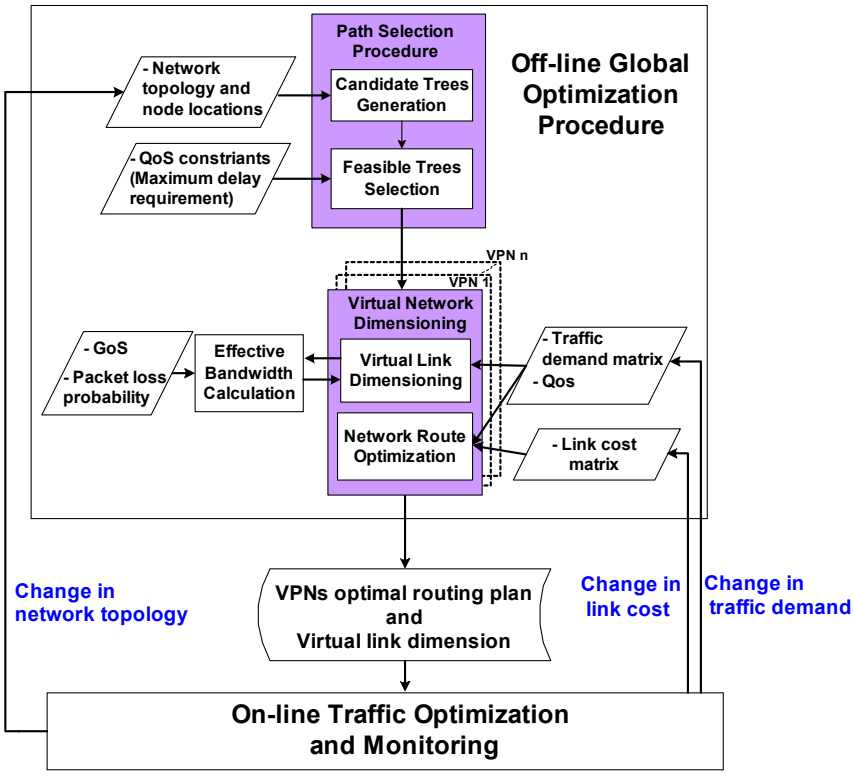
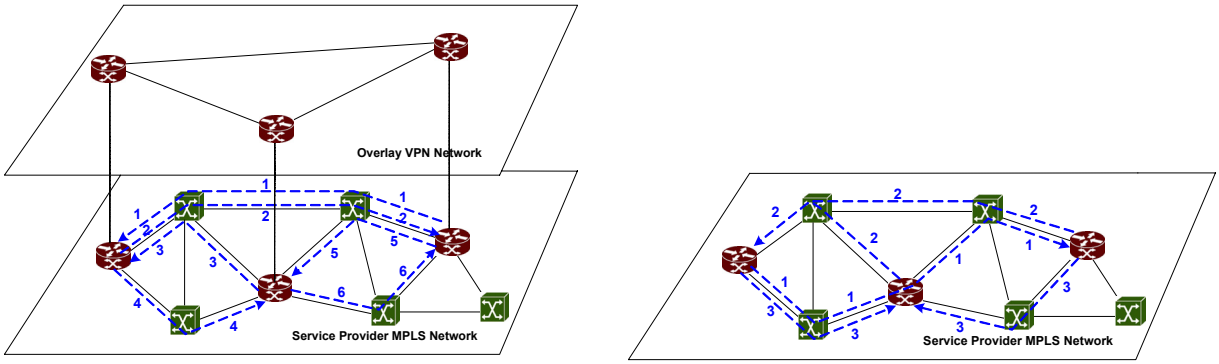


Figure 4.1: VPN traffic engineering procedure

4.2. Issues in VPN Design over NGI

Based on the proposed design approach, VPNs design problem is formulated as a mixed integer-programming model. The objective of the InP model is to minimize the cost of laying out a VPN of different traffic types and service classes on a physical network while meeting certain QoS requirements. The following are assumed to be known: (i) the physical topology and its bound on the maximum link capacity, (ii) capacity cost, (iii) the traffic demand matrix of all VPNs and QoS profiles. There are certain aspects which should be

addressed in designing VPNs over NGI and, thus, are included in the mathematical model formulation as discussed below.



(a) *Full-Mesh Design* *(b)* Sink-tree Design

Figure 4.2: Full-mesh versus sink-tree design

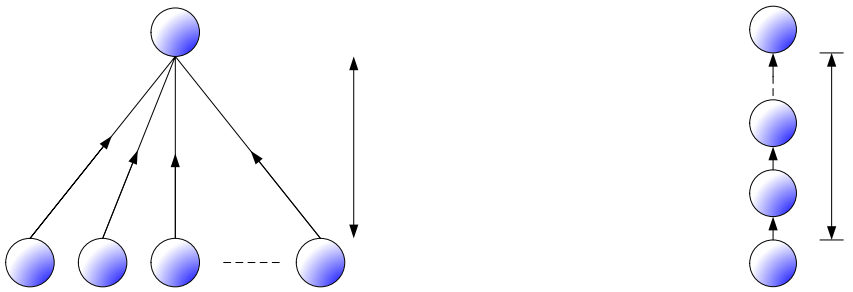
4.2.1. Sink-tree LSP path

As previously mentioned, a label switched path (LSP) in MPLS can be a multipoint-to-point path, in here, referred to as a sink-tree path. One clear benefit of using as a sink-tree path is scalability in term of management since fewer LSPs must be created compared to using a point-to-point path between each demand pair. Thus, management can be much easier when dealing with a large number of VPNs. For example, in a full-mesh design over a N-node network, a point-to-point path will be assigned to every node-pair having demand between them. The total number of LSPs required in a full-mesh design is $N\cdot(N-1)/2$. However, this number can be reduced to N paths using a sink-tree design. Figure 4.2 displays a full-mesh versus sink-tree design for a 3-node VPN over an 8-node MPLS network. Assume that there is a directional demand of one unit between 3-node pairs in the VPN network and each link in the MPLS network has one unit cost. A full-mesh design

requires 6 LSPs compared to 3 LSPs in sink-tree design. Both designs use the same links in MPLS network. However, the full-mesh design has 14 unit cost while the sink-tree design has a cost of 12. The cost saving is a result of the capacity efficiency gain attained when traffic is merged in a sink-tree design.

4.2.2. Tree Selection

The choice of routing trees, over which traffic will be mapped, is important as it affects goodness of the solution obtained as well as a computational time. To reduce the problem size (and thus a computational time) for a large network, a precomputed candidate set of trees is used in the model over which the optimal routes and capacity assignment is determined. Given a physical network topology, a path set is generated for each source and its destinations. This set is limited by a maximum hop-count allowed between each source-destination pair such that the maximum end-to-end delay is bounded. Typically, the choice of a routing tree largely affects the capacity required, especially, if the bandwidth of traffic flows is aggregated and multiplexed when they are merged within the network. Note that, this is possible only for connections within the same service class, in which a statistical multiplexing can be achieved and a portion of allocated bandwidth can be shared.



(a) Sink-tree with 1-hop depth *(b) Sink-tree with (n-1)-hop depth*

Figure 4.3: Sink-tree routing paths for n nodes

In tree selection, there exists a trade off, between minimizing an end-to-end delay requirement versus minimizing cost of link capacity. Illustrated in Figure 4.3 are two different choices of a sink-tree for *n* nodes. The depth of a tree is defined as the maximum distance between a root node and any leave node. Figure 4.3(a) show a sink-tree of 1-hop paths with a maximum depth of one. This choice of a tree yields a minimum delay between demand node-pairs, but, since 1-hop paths merge at the root node, no bandwidth aggregation is possible. In another extreme, the tree shown in Figure 4.3(b) with a depth of $N-1$ yields a maximum bandwidth gain due to statistical multiplexing at almost all links. Different types of trees including spanning trees, shortest (distance) path trees and Steiner trees (minimum-cost trees) are among potential choices.

4.2.3. VPN service classes

In MPLS, different QoS classes will be supported. Here we consider the IETF Differentiated Services QoS classes of premium service, assured service and best-effort service. In the premium service class, packet loss, delay and delay jitter must be bounded. The traffic of this class requires an absolute bandwidth guarantee. In the assured service class applications have the ability to tolerate a certain amount of delay and loss. For this traffic class, a mean bandwidth guarantee is needed along with a statistical delay bound. The best-effort service class corresponds to current Internet service with no QoS guarantees.

4.2.4. Call retry attempts

One important aspect is the issue of call retry or repeat attempt. End-user applications such as Web-based applications or voice telephony applications tend to have an automatic call repeat attempt feature. Therefore, when a requested call/connection is first rejected, there is a certain probability that a rejected call/connection will be retried, as shown in Figure 4.4.

The probability of call return affects the amount of traffic seen at the access points of the network. More specifically, the call return traffic will increase the amount of traffic offered to the VPN. Assuming that the probability of call retry is known, the network dimensioning must be adjusted to reflect the increase in actual offered traffic.

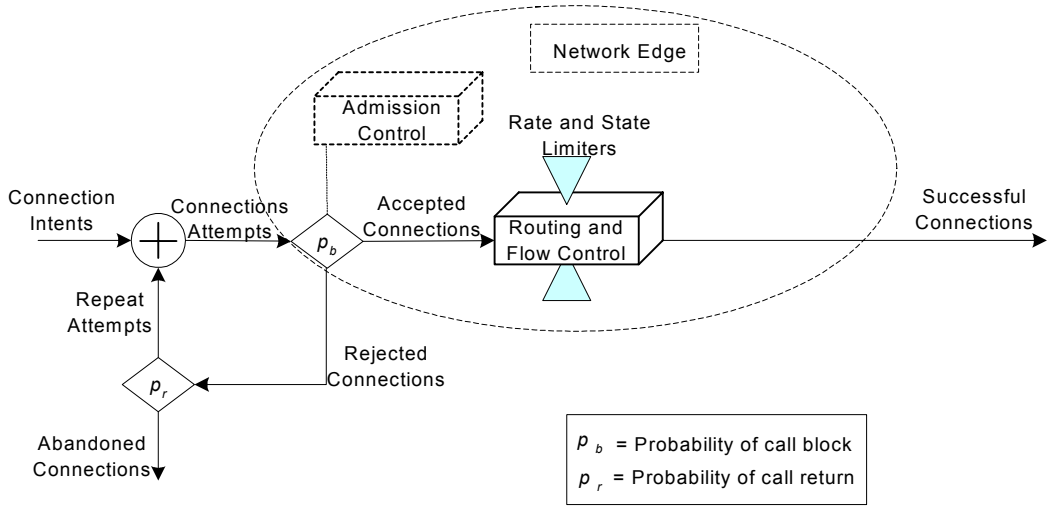


Figure 4.4 : Call repeat attempt feature

4.2.5. Multi-point Connections

Two types of traffic connections can be supported over MPLS and should be considered including point-to-point connections and multipoint connections. Multipoint connections include broadcast (point-to-multipoint), merge (multipoint-to-point), composite (point-to multipoint and multipoint-to-point) and full multipoint (multipoint-to-multipoint) [43]. While multipoint connections can be supported using multiple point-to-point connections, the resulting level of network resource utilization will be low. Thus, one seeks a good methodology to efficiently map multipoint traffic to candidate routes and wisely allocate bandwidth to each segment of a selected route corresponding to different sets of connections. For example, in a multipoint-to-point connection, if only one source can be

active at a time, the bandwidth allocated to a segment from the merge point to the downstream point can be a maximum of all source-transmitted rates. This is not considered a portion of bandwidth that can be shared among different VPNs. Initially, the proposed research will focus on point-to-point connections, then, extensions to the model for multipoint connections will be given.

4.3. MPLS VPN design formulations

Given a network topology, node location and link capacity, an optimization model is formulated for VPN design. A physical network is represented by a graph $G(N, L, C)$ where N, L and C is a set of nodes, links and maximum link capacity of the network respectively. M ($M \subset N$) is a set of edge nodes (edge routers) where traffic demands enter or exit the network. Thus, $N - M$ represents a set of core nodes (core routers). The complete notation used in the proposed formulations is given below. For each link $l \in L$, utilization factor α_l limits the proportion of the link capacity C_l to be allocated for VPN traffic. This utilization factor may be used to protect certain links from being overly subscribed or subjected to potential congestion. For example, a smaller value of α_l may be assigned to links connecting to core-routers than ones connecting to edge-routers. This factor is assumed to be known.

4.3.1. Notation

L Set of links in the network.

N Set of nodes in the network.

M Set of edge nodes in the network, $M \subset N$.

C_l Maximum capacity of link $l \in L$

α_l Utilization factor of link $l \in L$

K Demand set index, $K \subset M$

$P_k^{v,s}$ Set of feasible sink-trees ending at node $k \in N$ spanning all nodes $m \in M$ of service class $s \in S$ of VPN $v \in V$

$D_{v,s,k}$ Set of point-to-point demand pairs in demand set $k \in K$ of service class $s \in S$ of VPN $v \in V$

$B_{v,h,s,k}^d$ Bandwidth requirement of demand pair $d \in D_{v,s,k}$ of service class $s \in S$ of VPN $v \in V$ during hour-period $h \in H$

Y_l Sizing (topology) variables, specifically the capacity assigned to VPN traffic on link $l \in L$

ψ_l Cost of a capacity on link $l \in L$ (measured in units of $ per Mbps)

$U_{v,h}^l$ Capacity at link $l \in L$ allocated to VPN $v \in V$ during hour-period $h \in H$

$X_{v,h,s,k}^p$ Demand-path routing decision variables

= 1 if path $p \in P_k^{v,s}$ is used for demand set $k \in K$ of service class $s \in S$ of VPN $v \in V$ during hour-period $h \in H$

= 0 otherwise

$\gamma_{p,d}^l$ Link path incidence matrix

= 1 if demand pair $d \in D_{v,s,k}$ of set $k \in K$ that uses path $p \in P_k^{v,s}$ is directed using link $l \in L$

= 0 otherwise

$EB_{v,h,s,k}^l$ Estimated BW requirement of all demand type $k \in K$ on link $l \in L$ of service class $s \in S$ of VPN $v \in V$ during hour-period $h \in H$

$Eqv(B, T_s, QoS_s)$ Equivalent bandwidth calculation function for traffic in service class $s \in S$ with requirement of bandwidth amount B (with traffic descriptor T_s and quality of service requirement QoS_s)

4.3.2. Traffic demand

A complete matrix of VPNs traffic demands is assumed to be given. It can be derived from all SLAs (Service Level Agreements), between customers and service providers. SLAs typically specify various classes of service and how much traffic in each service class a user is allowed to send. In another words, for each source-destination (ingress-egress) node-pair, the matrix of each VPN specifies the required bandwidth and its QoS parameters (i.e., end-to-end delay requirement, delay jitter). Traffic demand $D_{v,s,k}$ will be assigned a route based on its egress node $k \in K$, where $K \subset M$ is called the demand set index.

4.3.3. Selection of candidate paths

Feasible sink-trees or multipoint-to-point paths are used in the optimization model where a traffic demand may be assigned. The feasible path set $P_k^{v,s}$ can be pre-computed for VPN v and service class s having different QoS requirements (i.e., maximum end-to-end delay requirement). A path $p \in P_k^{v,s}$ is selected from candidate sink-tree paths which are spanning trees rooted at egress node $k \in K$ and spanning over all the edge nodes $m \in M$ or a subset of the edge nodes. A set of candidate paths can be generated by enumerating all distinctive spanning trees. Algorithms to determine such trees can be found in many places in the literature including [24]. The maximum end-to-end delay requirement is translated in to the hop-count limitation constraint. This constraint will limit the path set where only feasible paths are selected from all candidate paths.

4.3.4. Bandwidth calculation

The bandwidth requirements at each link are estimated based on an effective bandwidth calculation [39] where the traffic parameters such as connection peak rate and its

burstiness are taken into account. Two classes of services in a Differentiated Service model are considered including premium service and assured services.

(i) Premium service

For premium service, each traffic connection is allocated a bandwidth equal to a source peak rate R_{peak}. Assuming that η connections are multiplexed within one link, the total allocated bandwidth (Eqv)

$$Eqv = \eta \cdot R_{peak} \tag{4.1}$$

where η is derived from an inverse Erlang formulation such that a grade of service constraint (GoS) of a connection (i.e., connection blocking probability - P_b) is met. Specifically,

$$\eta = InvErlang(a, P_b) \tag{4.2}$$

where a is a source utilization or the offered load of a connection.

(ii) Assured service

Source traffic in the assured service class is characterized by its source peak rate - R_{peak}, utilization factor - ρ, and mean burst period - b. In this case, the allocated bandwidth **(** Eqv **)** is less than $\eta \cdot R_{peak}$, specifically

$$Eqv = \min\left\{\eta \cdot m + \alpha'\sigma,\quad \eta \cdot \hat{c}_i\right\} \tag{4.3}$$

where $\alpha' = \sqrt{-2\ln(\varepsilon) - \ln(2\pi)}$ given $m-$ a mean bit rate, $\sigma-$ a variance bit rate, and $\varepsilon-$ buffer overflow probability. The equivalent capacity estimation for each source $\hat{c}_i$ is

$$\hat{c}_i = R_{peak}\,\frac{a - 1 + \sqrt{(a-1)^2 + 4\rho a}}{2a} \tag{4.4}$$

where

$$a = -\frac{b}{B}(1-\rho)\ln\varepsilon$$

It is assumed that $B-$ buffer size and $\varepsilon-$ packet loss ratio are known. The number of connections multiplexed η can be found as before from an inverse Erlang formulation.

4.3.5. A VPN design model without bandwidth aggregation

Using a sink-tree routing path, traffic demands can be merged within the network, thus the required bandwidth after the merged point can be allocated separately for each demand-pair or multiplexed together within the same service class. The latter yields a reduction in the bandwidth required especially for traffic in the assured service class, due to a statistical multiplexing gain. The basic formulations are given below. The model assumes that the followings are given: (i) link utilization factor α_l and maximum link capacity C_l, (ii) set of traffic demand pair $D_{v,s,k}$ and bandwidth requirement $B^d_{v,h,s,k}$, (iii) a precomputed sink-tree path set $P^{v,s}_k$ and a corresponding link path incidence matrix $\gamma^l_{p,d}$. The formulation seeks to find each VPN link capacity allocation $U^l_{v,h}$ and their route $X^p_{v,h,s,k}$ for all VPNs in each hour period.

VPNBA model formulates the VPN design problem when bandwidth aggregation is not possible. The objective of the formulation is to minimize the total capacity cost in providing service to all VPNs. For each VPN, service class, and hour period, constraint (1) selects only one path from a pre-computed set of feasible sink-tree paths ending at egress node P_k for each demand set k. Constraints (2) – (5) impose that capacity assigned at each link must not be greater than a maximum utilization limit of link capacity ($\alpha_l \cdot C_l$). Note that, in constraint (2), the capacity calculation is done separately for each traffic demand-pair.

Constraints (6) – (7) require that routing variables and capacity assignment variables must be positive. This formulation yields different route assignment and capacity allocation at different hour-periods.

VPNBA : ***Minimize*** $\sum_{l \in L} \psi_l * Y_l$

Subject to :

(1) $\sum_{p \in P_k} X^{p}_{v,h,s,k} = 1$; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$

(2) $EB^{l}_{v,h,s,k} = \sum_{d \in D_{v,s,k}} Eqv\left(B^{d}_{v,h,s,k}, T_s, QoS_s\right) * \left(\sum_{p \in P_k} \gamma^{l}_{p,d} * X^{p}_{v,h,s,k}\right)$
; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$, $l \in L$

(3) $\sum_{s \in S} \sum_{k \in K} EB^{l}_{v,h,s,k} \leq U^{l}_{v,h}$; for all $v \in V$, $h \in H$, $l \in L$

(4) $\sum_{v \in V} U^{l}_{v,h} \leq Y_l$; for all $h \in H$, $l \in L$

(5) $Y_l \leq \alpha_l \cdot C_l$; for all $l \in L$

(6) $X^{p}_{v,h,s,k} \in \{0,1\}$; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$, $p \in P_k$

(7) $Y_l \geq 0$; for all $l \in L$

4.3.6. VPNs design model with bandwidth aggregation.

Here, the VPWBA design model is introduced when bandwidth of various traffic demands is aggregated at a common link where possible. Using a precomputed sink-tree routing path, traffic demands can be merged within the network, thus the required bandwidth after the merged point can be allocated as the aggregated bandwidth of multiplexed traffic within the same service class. Here, the aggregation only occurs within a demand set destined

to the same egress node of a VPN. In this case, the objective function and constraints are similar to VPBNA except for constraint (2), where the total traffic demands routed on one link is aggregated for bandwidth allocation.

VPWBA : ***Minimize*** $\sum_{l \in L} \psi_l * Y_l$

Subject to :

(1) $\sum_{p \in P_k} X^p_{v,h,s,k} = 1$; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$

(2) $EB^l_{v,h,s,k} = Eqv\left(\left(\sum_{d \in D_{v,s,k}} B^d_{v,h,s,k} * \sum_{p \in P_k} \gamma^l_{p,d} * X^p_{v,h,s,k}\right), T_s, QoS_s\right)$; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$, $l \in L$

(3) $\sum_{s \in S} \sum_{k \in K} EB^l_{v,h,s,k} \le U^l_{v,h}$; for all $v \in V$, $h \in H$, $l \in L$

(4) $\sum_{v \in V} U^l_{v,h} \le Y_l$; for all $h \in H$, $l \in L$

(5) $Y_l \le \alpha_l \cdot C_l$; for all $l \in L$

(6) $X^p_{v,h,s,k} \in \{0,1\}$; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$, $p \in P_k$

(7) $Y_l \ge 0$; for all $l \in L$

4.4. Single-hour single-service class model

The mixed-integer formulations for the VPN design problem, shown previously, have a NP-hard complexity. A simplified version of these formulations can be derived considering only traffic demand of a VPN within one service class and hour-period. Thus, the *VPNBA* model can be reduced to :

SVPNBA : ***Minimize*** $\sum_{l \in L} \psi_l * Y_l$

Subject to :

(1) $\sum_{p \in P_k} X_k^p = 1$; for all $k \in K$

(2) $\sum_{k \in K} \sum_{d \in D_k} Eqv\left(B_k^d, T, QoS\right) * \left(\sum_{p \in P_k} \gamma_{p,\,d}^{l} * X_k^p \right) \leq Y_l$; for all $l \in L$

(3) $Y_l \leq \alpha_l \cdot C_l$; for all $l \in L$

(4) $X_k^p \in \{0,1\}$; for all $k \in K$, $p \in P_k$

(5) $Y_l \geq 0$; for all $l \in L$

In the same manner, the *VPWBA* model can be reduced to :

SVPWBA : ***Minimize*** $\sum_{l \in L} \psi_l * Y_l$

Subject to :

(1) $\sum_{p \in P_k} X_k^p = 1$; for all $k \in K$

(2) $\sum_{k \in K} Eqv\left(\left(\sum_{d \in D_k} B_k^d * \sum_{p \in P_k} \gamma_{p,\,d}^{l} * X_k^p \right), T, QoS \right) \leq Y_l$; for all $l \in L$

(3) $Y_l \leq \alpha_l \cdot C_l$; for all $l \in L$

(4) $X_k^p \in \{0,1\}$; for all $k \in K$, $p \in P_k$

(5) $Y_l \geq 0$; for all $l \in L$

Obtaining a solution to *SVPNBA* and *SVPWBA* is easier than *VPNBA* and *VPWBA*. A pilot study was conducted by translating *SVPNBA* and *SVPWBA* using the AMPL model description language. Solution is obtained by running CPLEX 7.1 InP solver on a Sun Blade1000 workstation with 750 MHz processor and 2 gigabytes of memory. Branch and bound solution technique is employed.

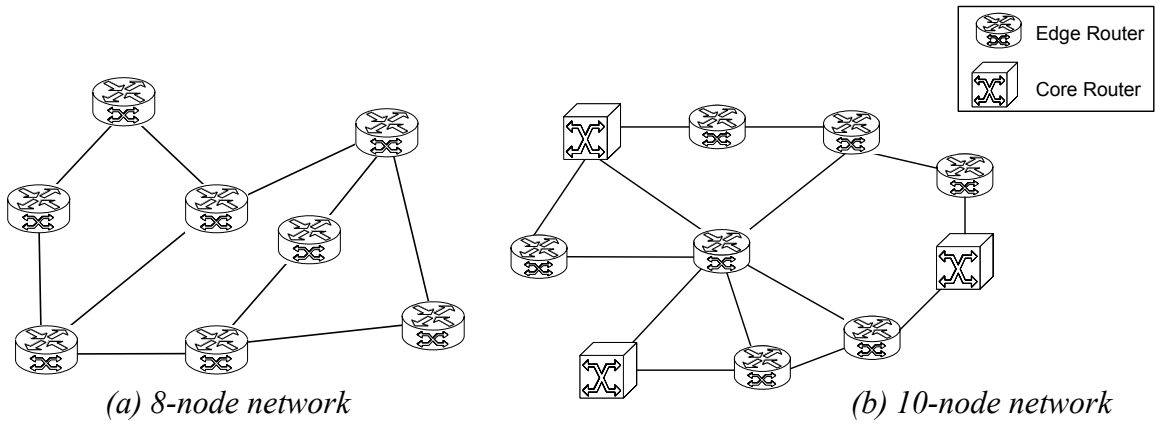


Figure 4.5 : Small networks under study

First, analysis was done on small networks with 8 and 10 nodes, (shown in Figure 4.5), where 7 ingress/egress nodes have demand entering and exiting. Each link has OC-1 capacity of 50 Mbps. The capacity is divided into smaller unit with a basic rate of 64 kbps and the link cost is the cost per capacity unit which is assumed to be equal for all links. The link utilization factor α_l, which is total capacity of link allocated to all VPNs traffic, is varied. Different demand patterns are generated including symmetric and asymmetric demand. In the symmetric demand (full-mesh demand) cases, all node-pairs within the set of edge nodes have a demand between them. In the asymmetric case, each VPN has asymmetric load of demand with nonzero demand only from a subset of network nodes. The nodes with

Topology	Full-Mesh Design			Sink-Tree(s) Design (w/o BW aggregation)			Sink-Tree(s) Design (with BW aggregation)		
	Optimal Cost	Simplex Iterations	No. of LSPs	Optimal Cost	Simplex Iterations	No. of LSPs	Optimal Cost	Simplex Iterations	No. of LSPs
Symmetric demand with fixed-load									
8-node	104	54	56	104	241	8	56	473	8
10-node	174	162	90	174	308	10	90	925	10
Symmetric demand with variable-load									
8-node	346	64	56	346	99	8	203	2,016	8
10-node	562	112	90	562	242	10	410	4,239	10
Asymmetric demand with variable-load									
8-node	166	42	33	166	101	7	105	319	7
10-node	212	76	35	212	216	7	129	565	7

Table 4.1 : Comparison for different design cases

nonzero demand were randomly selected. The offered load to the network can be fixed or varied according to discrete *Uniform(1,5)* distribution.

For a small network, it is possible to include a precomputed path set that includes all possible trees spanning all edge routers. Thus, the optimal cost can be obtained as shown in Table 4.1. Overall, it is shown that the optimal costs obtained from a sink-tree design without bandwidth aggregation are not different from ones obtained from a full-mesh design. However, when bandwidth aggregation is possible using multipoint-to-point LSPs, the sink-tree design introduces a cost saving approximately by 30-40 percent.

In terms of complexity, a sink-tree design model has a large number of technical variables and constraints. Thus, branch and bound process uses a larger number of simplex iterations solving a sink-tree model to find an optimal solution compared to a mesh-design model. For the same problem size, the number of simplex iterations is about double for a sink-tree design without bandwidth aggregation when compared to the mesh design. Note

from Table 4.1 that when bandwidth aggregation is considered in the sink-tree design, the number of simplex iterations to solve for an optimal solution increases significantly.

The results in Table 4.1 illustrate a clear benefit of using multipoint-to-point LSPs in the small number of labels and unique LSPs required for all VPNs traffic. This means that the number of records maintained for label mapping in a forwarding table at every LSR can be reduced using sink-tree LSPs. This in fact makes it easier to manage and, thus, more scalable for a large backbone network when a large number of VPNs are coexisting.

For medium to large networks, it is not possible to include a complete set of recomputed path between all egress routers due to the complexity of the problem. This is true for both full-mesh design and sink-tree design cases. Therefore, a hop limit constraint is introduced to reduce the size of the problem. A hop limit constraint in a path set is rather practical since a demand should not be routed through a long path which violates a maximum delay bound. Moreover, since the main objective of a design model is to minimize a capacity cost, a path that is too long is generally not selected in the optimal solution because a demand that is routed through a large number of hops tends to consume more capacity.

An application of sink-tree design for medium and large networks is then studied through another set of experiments where a hop limit constraint is enforced in a path set. Here, experiments were conducted over the two backbone networks shown in Figure 4.6 and 4.7 with 15 and 55 MPLS routers where 10 and 21 of those are edge routers respectively. The first represents a medium-size network with OC-12 links while the latter is an actual high-speed optical backbone network with OC-192 links. There are 4 VPNs and each VPN generates demand between 6 randomly selected edge routers. The demand generated could be symmetric or asymmetric demand varied according to a discrete uniform within {10, 15,

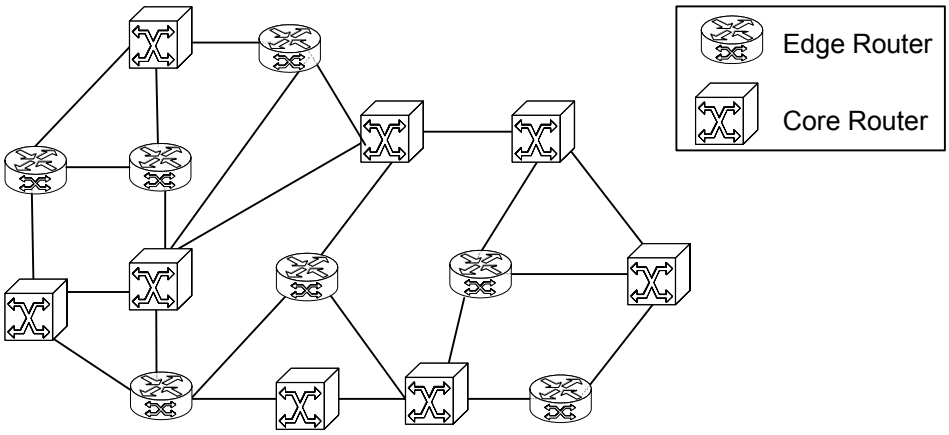


Figure 4.6 : Tested network with 15 routers and 24 links

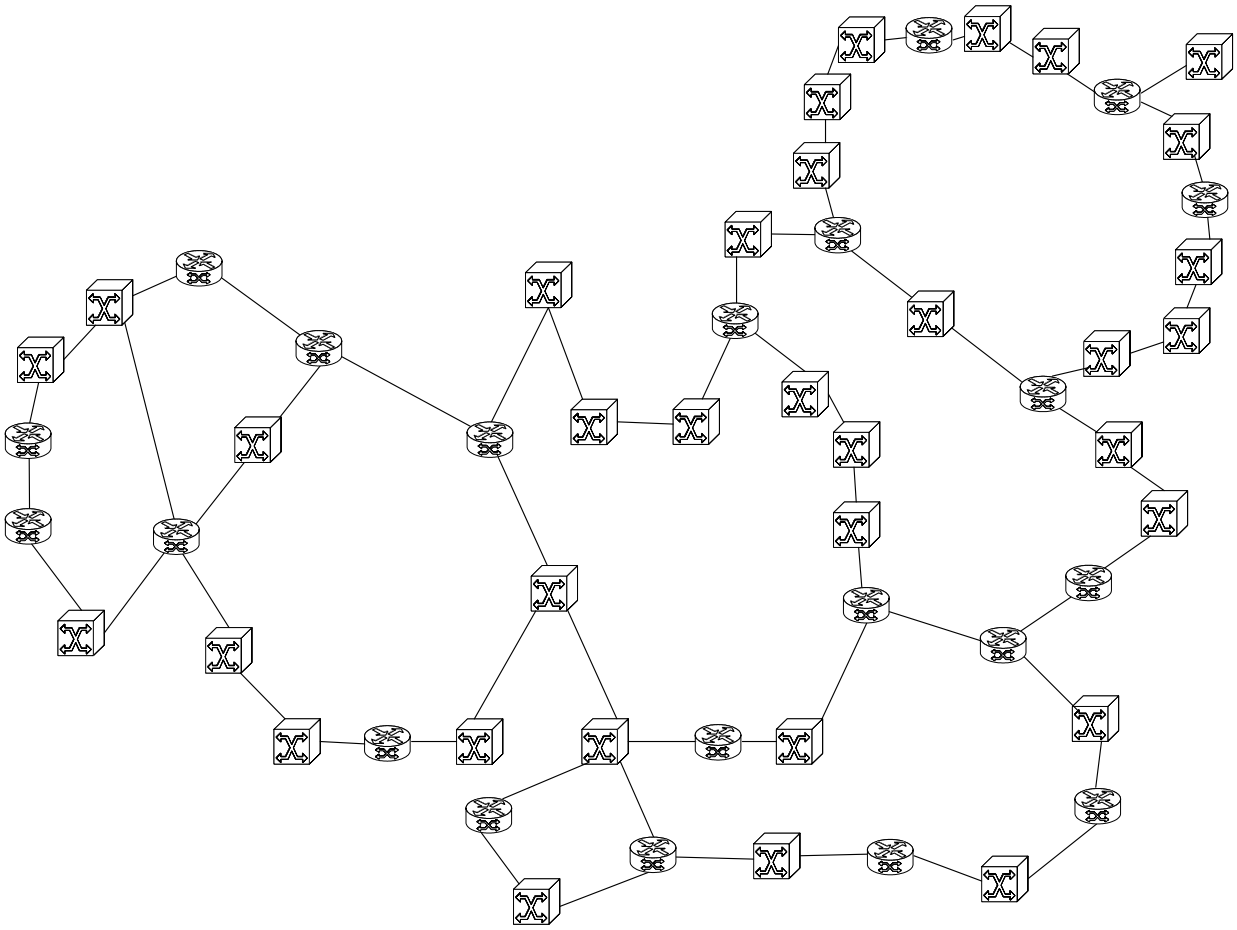


Figure 4.7 : Level-3 Network with 55 routers and 62 links

Topology	Full-Mesh Design			Sink-Tree(s) Design (with BW aggregation)		
	Optimal Cost	#Label / #LSPs	# links used Avg. (Max)	Optimal Cost	#Label / #LSPs	# links used Avg. (Max)
Symmetric Demand						
15-node	7587	120 / 87	5.79 (8)	6932	24 / 24	5.67 (8)
55-node	34417	120 / 110	23.04 (32)	30936	24 / 24	22.25 (32)
Asymmetric Demand						
15-node	5467	80 / 64	5.65 (8)	5083	20 /20	5.45 (7)
55-node	26999	108 / 86	20.83 (28)	24974	24 / 24	19.92 (27)

Table 4.2 : Comparison for different design cases

20, 25, 30} T1-unit rates. Characteristics of the traffic sources are assumed to be known *(R_p =300 Kbps, b =300 msec, ρ =0.2)*. Capacity will be allocated for each traffic sources such that the connection blocking probability at the edge router will not exceed 10^{-5}.

Table 4.2 compares the optimal solution obtained from a full-mesh design versus a sink-tree design. By utilizing sink-tree paths in the design, the cost can be reduced approximately by 10 percent for symmetric and asymmetric demand. It is clearly shown in all cases that fewer labels are required in a sink-tree design. Also, the number of LSPs is less when sink trees are used. From Table 4.2, one can observe that the optimal solution obtained from a sink-tree design yields a lower average number of links used as compared to a full mesh design in all cases. Besides, the maximum number of links used in sink-tree LSPs is smaller for the asymmetric demand case. These results indicate that the optimal sink-tree paths tend to share many links in common when possible. By sharing many links in common, capacity savings is promoted through bandwidth aggregation at shared links.

Moreover, it is observed that when demand traffic is not aggregated in a full-mesh design, traffic demand is simply routed through a shortest-path LSP. A sink-tree design,

however, uses multipoint-to-point LSPs, and, thus, the optimal path between a source-destination node pair may be longer than the one given by a full-mesh design when demand tends to be aggregated within the network.

Topology	Full-Mesh Design		Sink-Tree(s) Design (with BW aggregation)	
	Number of Constraints	Number of Technical Variables	Number of Constraints	Number of Technical Variables
15-node	360	5448	3720	314496
55-node	740	1188	9572	779248

Table 4.3 : Problem size of Full-Mesh Design versus Sink-Tree Design

Even though, a sink-tree design successfully demonstrates a bandwidth gain over a full-mesh design, it may have a limited practicability in a large network due to its complexity. This fact could be comprehended when the complexity of mesh-design and sink-tree design models is compared in term of number of constraints and technical variables (see Table 4.3). Even for a small network of 15 nodes, the number of constraints and technical variables are large compared to those in a full-mesh design. For the 55-node network, these numbers are much larger than the full-mesh design. The sink-tree design problem is more complex due to the fact that the size of pre-computed sink-tree paths is large since all possible tree-paths must be included. With a limitation in term of computational time as well as a memory space, the optimization process could be exhausted. Therefore, in order to apply a sink-tree design model to a realistic-size network, where a large number of VPNs demand are being deployed, the size of the problem should be reduced. One approach is to reduce the size of a precomputed sink-tree path to be searched from by the optimization process. Only a

small set of candidate paths which are more likely to be used in an optimal solution should be included in the design model. By analyzing the optimal route assignment in various sink-tree design cases, it has been observed that the optimal trees agree on sharing as many common links when possible. Considering this, a heuristic path selection is proposed to scale down the sink-tree design problem by efficiently select a candidate tree having less number of links in it. The detail of heuristic path selection will be elaborated more in Chapter 5.

4.5. Summary

In this chapter, the MPLS-based multi-hour VPNs design problem is formulated as a mixed integer programming model when a sink-tree routing path is employed with and without bandwidth aggregation. When VPNs is modeled using multiple logical sink trees, not only the number of label-switched paths is reduced but it also allows the possibility of bandwidth savings. Sample numerical results for different demand patterns shows that a sink-tree design without bandwidth aggregation yields similar results as a full-mesh design where demand is simply routed along a shortest-path. However, when bandwidth aggregation is considered in a sink-tree design, demand was routed along a tree that is different from a shortest-path tree such that link capacity could be shared among multiple connections, and, therefore, the total capacity cost reduction is realized. An extensive analysis of the problem solutions suggests that the optimal trees agree on sharing many common links when possible. Thus, a simple but good heuristics could be applied by imposing a number of links used in the path selection procedure to reduce the problem size and improve performance in solution process.

5. Tree Selection Heuristics

The VPN design model utilizing sink-tree routing paths is a complex problem. Finding optimal layout and capacity plan often involves enumerating a set of all possible routing tree paths. For example, if a routing tree is a tree spanning over all network nodes, the number of all distinct spanning trees can be as much as $N^{(N-2)}$ trees for a fully connected *N*-node network [24]. It is obvious that the problem size (i.e. the number of technical/decision variables) grows exponentially with the number of nodes. Therefore, applying a standard approach to solve such a problem is difficult and sometimes computationally prohibitive due to the complexity of the problem.

The size of the optimization problem largely depends on number of candidate sink-trees to be searched over. Limiting the candidate set of sink-tree paths will speed up the computation process and makes it possible to obtain a solution for a realistic-size network within a feasible amount of time. This approach has been widely used in many network design problems, most of which focus on routing traffic demands over a set of point-to-point paths subjected to various optimization constraints. Nevertheless, when a pre-computed path set is a multipoint-to-point or a sink tree path, several complicated issues arise since the number of trees to be searched over is much larger than the number of point-to-point paths. Here, a tree selection heuristics is proposed to scale the VPN design problem by choosing a small-but-good candidate set of feasible sink-tree paths. The proposed heuristics introduces the new selection criteria which limit the number of links used in a tree rather than a number of hops used in a point-to-point path selection.

5.1. Tree Selection Criteria

When multiple point-to-point connections are mapped over a multipoint-to-point route, a merging of demand yields a large bandwidth reduction by multiplexing multiple demands. To minimize the total bandwidth usage of multiple VPNs utilizing multipoint-to-point routes, traffic is concentrated on a small number of links. In another words, multiple optimal trees use as many links in common as possible to increase the gain in bandwidth aggregation. This fact has been observed across multiple demand scenario including symmetric and asymmetric demand over different sizes of infrastructure networks. Therefore, with traffic concentration, it is advisable to install large capacities on a small number of links rather than installing relatively small capacities on many links.

These observations suggest some guidelines in choosing good candidate tree paths to be search over in a VPNs design model. Typically, in a VPNs design model using a precomputed point-to-point path set, a hop limit constraint is introduced to impose a bound on the maximum delay as well as to reduce the problem size. However, when bandwidth aggregation is considered in multipoint-to-point routes, the set of tree paths used in the optimal solution tend to use a small number of links. In view of that, heuristics is proposed to reduce problem size by shrinking the precomputed set of candidate tree paths by imposing a limit in number of links used in candidate trees, in addition to the hop-count limitation. Both the hop-count and the number of links used in a tree are two critical factors in selecting a good candidate set of trees. By imposing the hop-count constraint, it simply avoids choosing single-branch trees which may introduce an undesirable delay violation. When trees having a fewer number of links are chosen, bandwidth sharing is promoted.

For N - node network, a tree spanning all nodes $n \in N$ uses at most $N-1$ links and a tree spanning a set of edge nodes $m \in M$ ($M \subset N$) will use at least $M-1$ links. Thus, a tree path will be selected only if it uses less than R links where $M-1 \leq R \leq N-1$. The value of R will be depended on network topology. Note that for a small dense network the value of R is closer to $M-1$ but for a sparse medium-to-large network the value of R is closer to $N-1$. Choosing a good number for R will greatly reduces the problem size especially when the network is large and the number of edge nodes is small compared to the number of network nodes. The proposed heuristics opts to select a fixed number of such a tree with the smallest R value. Numerical results are presented to study the effect of the size-limited candidate tree set and measure an improvement in term of computational time used to solve a sink-tree design problem.

5.2. Tree Selection Heuristics

A tree selection heuristics is proposed here. In addition to imposing a hop-count limit, the proposed heuristics will choose a fixed number of sink-tree paths by ranking trees based on the number of links used. Essentially, trees having a fewer number of links are preferred. The proposed heuristics tree selection algorithm (shown in Figure 5.1), operates in three phases: (i) *Node and link elimination,* (ii)*Tree generation,* (iii)*Tree ranking.*

In the node and link elimination phase, the network size is reduced by removing nodes and links that serve only as transit points of the traffic. The transit node is defined as a non-switched node where traffic can not be merged and a transit link is a link connected to a transit node. Specifically, the algorithm will delete a non-demand node having node-degree of two and replacing its links with a direct link between its neighbor nodes. With a node and

link elimination, the size of the problem size can be reduced considerably without changing the solution. In other words, if a direct link, that replaced deleted nodes and links, should be included in the candidate tree, both deleted nodes and links will be included in the candidate tree as well, and vice versa. Therefore, the optimal tree is still included in the set of candidate trees generated from a reduced network. By doing this, the number of all distinctive trees spanning over all edge routers $m \in M$ can be significantly reduced especially for a large-and-sparse network having low average node-degree. Note that a similar approach was adopted for the design of survivable backbone networks [38], where the approach was termed meta-mesh abstraction.

In the tree generation phase, all distinctive trees spanning over a set of demand nodes are generated. Given a cost function, finding a minimum-cost tree spanning over a sub set of nodes is similarly to solving the classical Steiner tree problem which is known to be an NP-hard problem. Solving such problem is prohibitive given limited time and resources. However, enumerating all distinctive trees spanning over a set of nodes is a much easier task. Tree enumeration could typically be done in a polynomial time [24]. Therefore, it makes much more sense to generate all possible trees and, then, select "good" candidate trees to be optimized based on certain cost and criterion.

In a network *G*(*N, L)*, having *N* nodes and *L* links, all distinctive trees spanning over *N* nodes can be generated using a complete enumeration method. The number of all distinctive spanning trees for a fully connected *N*-node network has been proved to be N^{N-2} for a fully connected mesh network [18]. Generally, for a typical network, the number of distinctive spanning trees can be bounded by

$$J = \left| B_0 \cdot B_0^t \right| \tag{5.1}$$

Where B_0 is the incidence matrix of the network G with one row removed (i.e., reduced incidence matrix with *N*-1 independent rows) [74]. As an example, the number of all distinctive spanning trees for tested-networks, shown in Chapter 4, is computed and listed in Table 5.1. For 8-node network, there is a total of 60 distinctive trees but for the 55-node network this number could be as large as 20,466,224. It is obviously shown that the number of trees grows rapidly with the number of nodes. Nevertheless, when the node and link elimination is applied to the 55-node network having 62 links, the reduced network contains 24 nodes and 32 links as shown in Figure 5.2 and total number of distinctive trees is reduced to 196,147. This reduction is nearly 100 times smaller. The benefit of node and link elimination process is especially significant for a network having low average node degree as is the case in North American backbone network.

Additionally, the generated tree set is further reduced size with an introduction a hop-count limit constraint. A tree is simply removed from the candidate set if the number of hops between any node-pair becomes larger than a hop-count limit. Choosing the right value for the hop-count limit effects both the size of candidate trees set as well as a solution obtained from the optimization procedure. Using a single value of a maximum hop-count may not be effective since it is quite rigid for a node-pair that is far apart and too loose for a node-pair separated by a single-hop away. Here, a hop-count limit is defined as a shortest distance between a demand node-pair plus a hop-count factor (δ). In general, the value of hop count factor should be selected based on a network topology, i.e., δ should be small for a dense network, and large for a sparse network.

Begin :

/ Node and link elimination */*

Let R = a set of nodes in a reduced network;

Initialize $R = N$;

for each non-demand nodes $w \in (N–M)$

{ if (*node-degree(w)* ≤ 2)

for node i, j adjacent to r,

{ replace link *(i,r)* and *(r,j)* with link *(i, j);*

let $R = R - \{r\}$; }

}

/ Tree generation */*

for all nodes R in a reduced network

{ let v = the size of possible candidate sink-tree T_k

destined to node k;

while(*size-of*(T_k) $< v$)

{ find a distinctive tree t spanning over a set of

demand nodes M ;

/ Hop-count limitation */*

for each destination node $k \in M$

for each demand-pair $(s,d) \in MxM$ where $d = k$, $s \neq k$

if (path-length $p(s,k)$ < shortest distance $q(s,k) + \delta$)

let $T_k = T_k \cup \{t\}$;

}

}

/ Tree ranking */*

for each sink-tree set T_k

{ for all $t \in T_k$

let $\pi(t)$ = tree rank based on the number of links used;

let u = the size of candidate sink-tree $G_k \subseteq T_k$;

choose candidate of trees $g \in G_k$ of size u where

$\{\pi(g_1), \pi(g_2) \ldots, \pi(g_u)\} \leq \{\pi(t_{u+1}), \pi(t_{u+2}), \ldots, \pi(t_v)\}$;

}

End

Figure 5.1 : Heuristics Tree Selection Algorithm

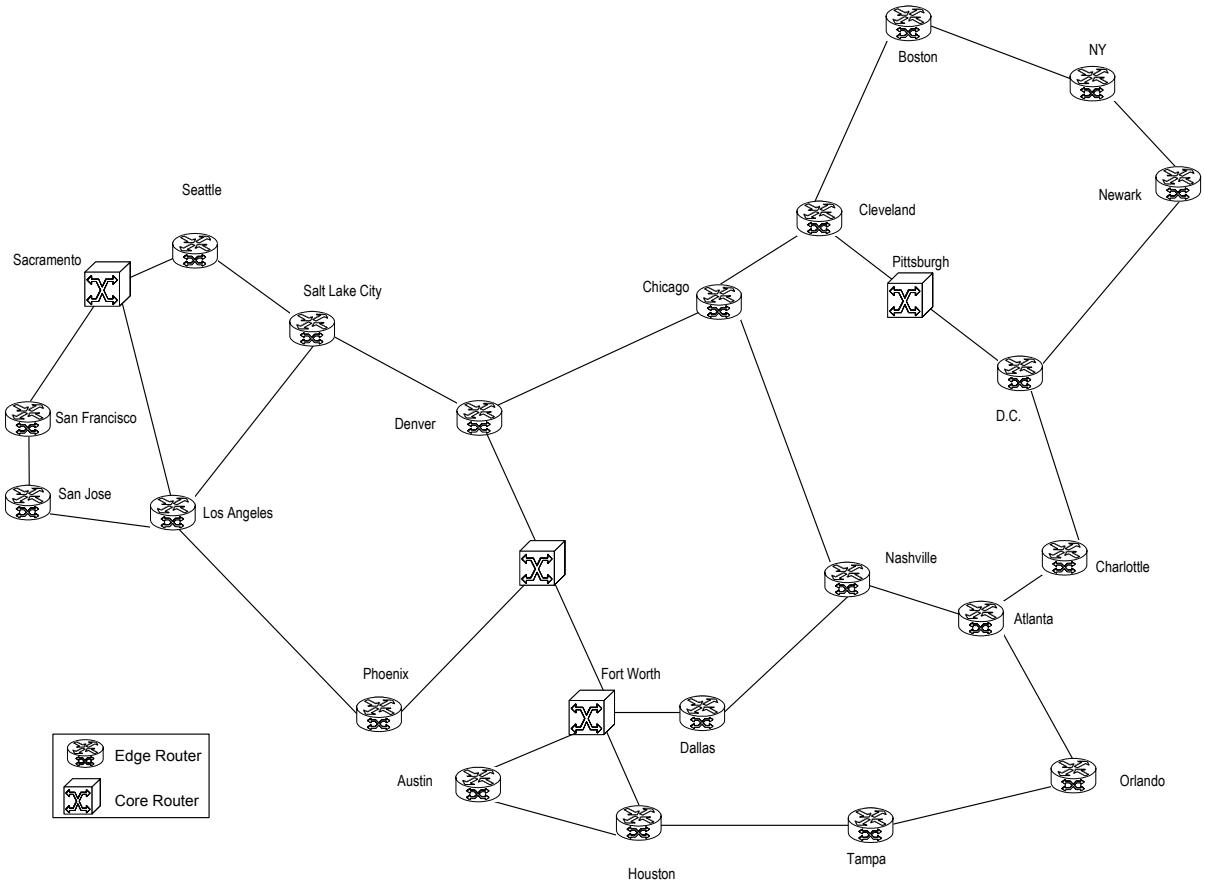


Figure 5.2 : Reduced 55-Node Network

Network	The Number of Distinctive Spanning Trees
8-node	60
10-node	383
15-node	112,252
55-node	20,466,224
Reduced 55-node	196,147

Table 5.1: Number of Distinctive Spanning Trees

Lastly, in tree ranking phase, a set of feasible trees will be ranked based on the number of links used in a tree path. This process is done separately for each egress node to choose candidate trees ending at it. Specifically, a fixed number of trees having small weight (i.e. number of links) will be chosen to be used in an optimization procedure. Essentially, trees having less number of links are preferred.

5.3. Performance Evaluation

Consider the two backbone networks shown in Figure 4.5(b), 4.6 and 4.7 with 10, 15 and 55 routers where 7, 10 and 21 of those are edge routers respectively. The first two represent medium-size networks with OC-12 links while the latter is an actual high-speed optical backbone network with OC-192 links. There is a total of 4 VPNs and each VPN generates demand between randomly selected edge routers. The demand generated could be symmetric or asymmetric demand varied uniformly within {10, 15, 20, 25, 30} T1-unit rate. Characteristics of the traffic sources are assumed to be known (R_p=300 Kbps, b=300 msec, ρ=0.2). Capacity will be allocated for each traffic sources such that the connection blocking probability at the edge router will not exceed 10^{-5}. The multiple VPNs design problem is solved using AMPL with CPLEX 7.1 InP solver running on a Sun Blade1000 workstation with 750 MHz processor and 2 gigabytes of memory. A branch and bound solution technique is used.

5.3.1. Effect of Number of Candidate Tree Paths

The performance of the proposed path selection heuristics are compared in term of computational time measured by the real CPU time in seconds and the total capacity cost

obtained for different numbers of tree paths which is precomputed using the proposed tree selection heuristics. Complete numerical results are listed in Appendix A. Selected results are shown in Figure 5.3-5.17.

The optimal cost and computational time of the 10-node network for 1 and 4 VPNs at different number of tree paths are shown in Figure 5.3 and 5.4. For the case of the 10-node network, maximum hop-limit of 9 is chosen to observe the real effect of using a small number of links in candidate tree set which is not restricted by a hop-count constraint. Overall, the results establish the fact that as the number of paths increases, a computational time increases exponentially, while, the obtained solution converges closer to the optimal solution. Comparing results of 1 and 4 VPNs, the computational time has shown to be increasing with the number of VPNs. Besides, it is observed that the number of tree paths requirement increases with more number of VPNs. For a single VPN, the optimal solution can be reached at the tree set size of 180 trees with 0.33 seconds of CPU time; for 4 VPNs the optimal solution can be found at the tree set size of 250 trees with 1000 seconds of CPU time. For both cases, the number of candidate trees required to solve for the optimal solution has shown to be smaller than when all possible trees are included.

The effect of imposing different hop-limit values in the proposed heuristics path selection is studied in the case of the 15-node network as illustrated in Figure 5.5. The hop-count factor (δ) of 2 and 5 are used for a comparison. When δ=5, the optimal solution can be found at 3000 paths. However, when δ=2, the number of trees path required to compute for optimal solution reduces to 1000 paths and the solution can be obtained in less CPU time. With a small hop-count factor, the obtained solution will converge faster to the true optimum using a smaller set of candidate paths. However, if the hop-count factor is too small,

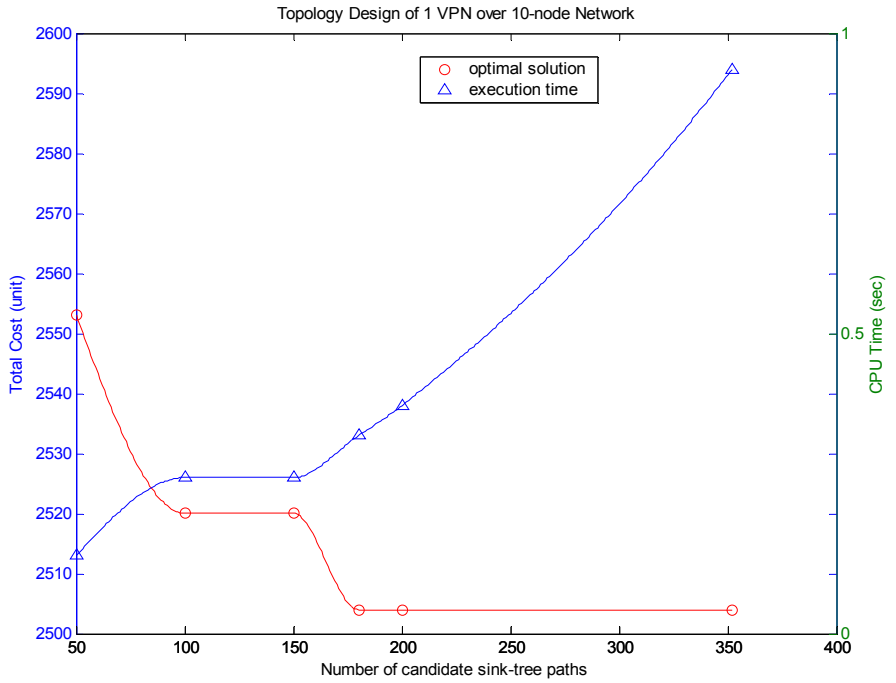


Figure 5.3 : Optimal cost versus CPU time of 1 VPN over the 10-node network

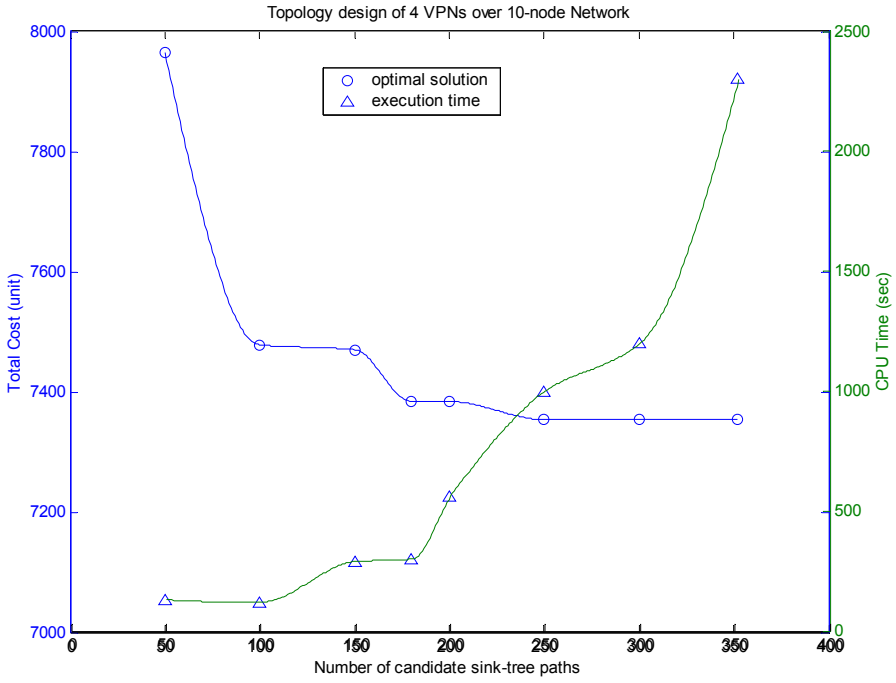


Figure 5.4 : Optimal cost versus CPU time of 4 VPN over the 10-node network

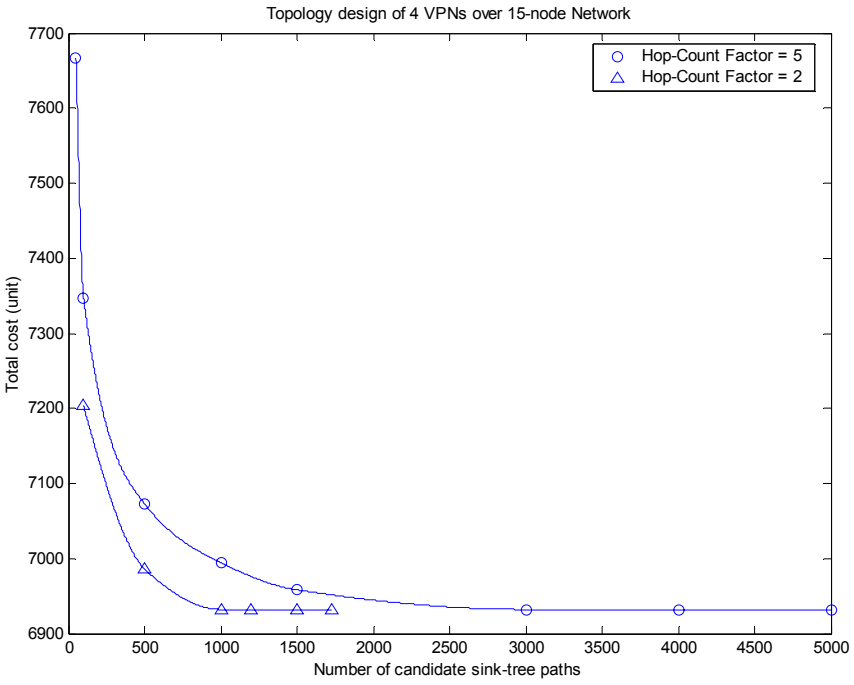


Figure 5.5 : Effect of hop-limit in heuristics path selection

the optimization may not be able to reach a true optimum or may find it infeasible to find any solution. This implies that choosing the right value for hop-count factor in the heuristic path selection algorithm will greatly affect the goodness of the candidate sink-tree paths. Therefore, with the right value of the hop-count factor, the optimization procedure takes less effort and reaches the optimal solution faster.

The advantage of the heuristics path selection can be more pronounced in the case of the 55-node network as illustrated in Figure 5.6 and 5.7. After a certain number of tree paths, the computational time swiftly rises to a high number. Yet, the optimal solution could be established at a smaller number of tree paths. With Symmetric demand (shown in Figure 5.6), the true optimal solution can be best approximated with 600 sink-tree paths and takes almost 1 hour of computational time. Figure 5.7 presents the case of Asymmetric demand

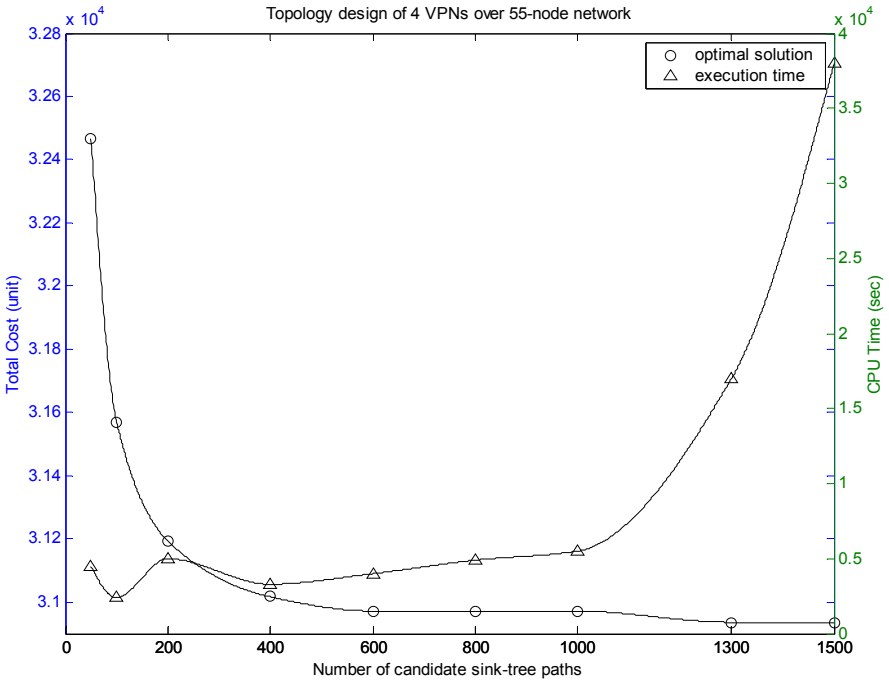


Figure 5.6 : Optimal cost versus CPU time of 4 VPNs over the 55-node network

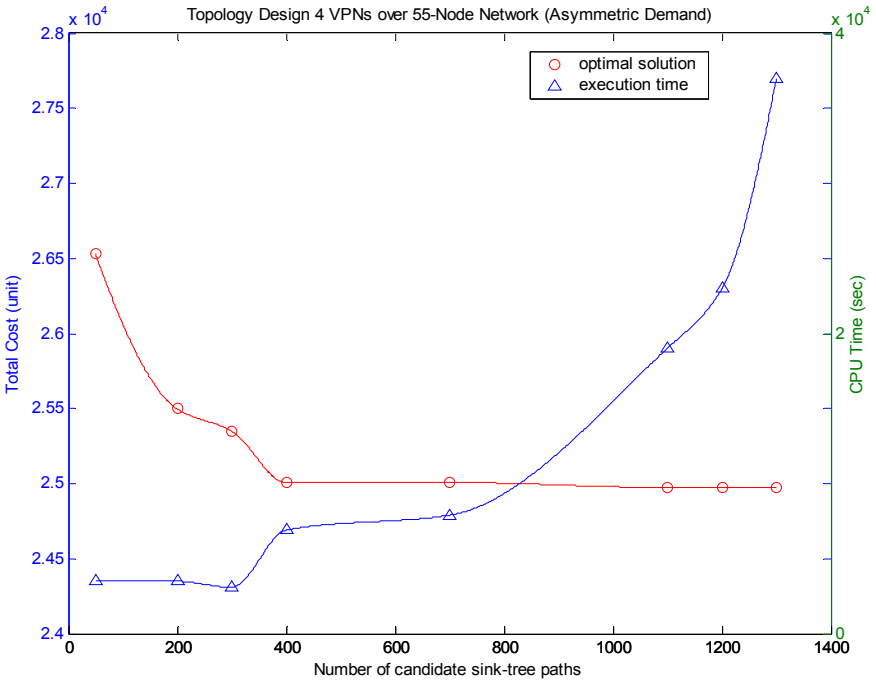


Figure 5.7 : Optimal cost versus CPU time of 4 VPNs over the 55-node network

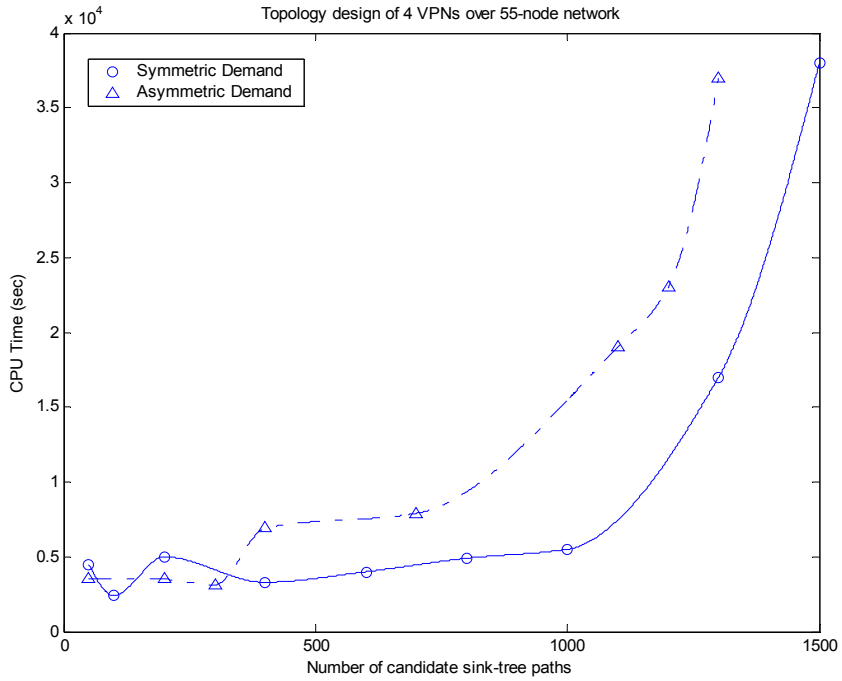


Figure 5.8 : Comparison of CPU time in symmetric and asymmetric demand cases

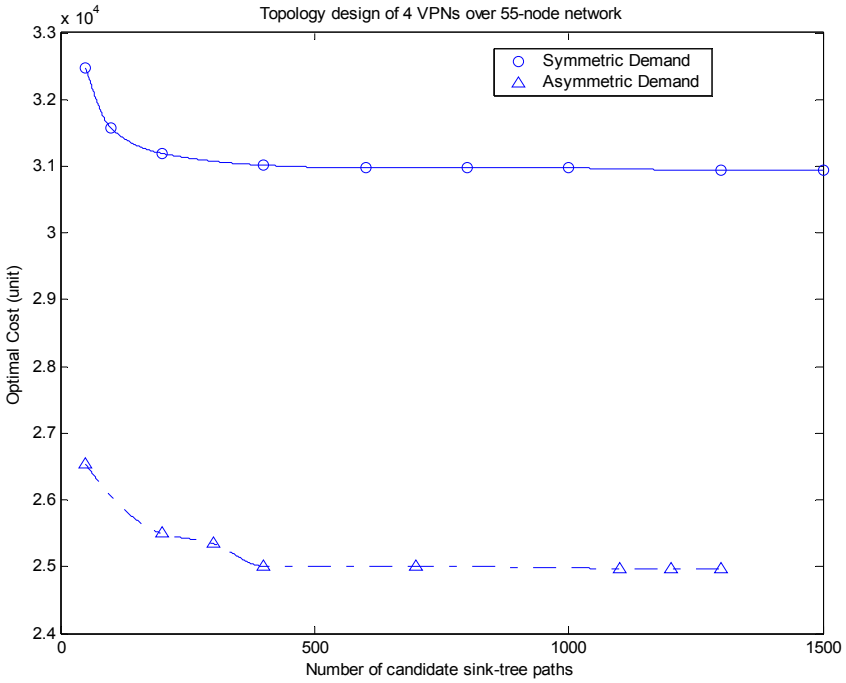


Figure 5.9 : Comparison of optimal cost in symmetric and asymmetric demand cases

under the same load. In this case, the optimal solution can be best approximated at 400 sink-tree paths and it takes 2.8 hours of computational time. Even though, a smaller number of sink-tree paths may be used in the case of asymmetric demand, the optimization procedure finds it harder to solve for the optimum and, therefore, consumes more CPU time. This could clearly be seen in Figure 5.8 where the CPU time of symmetric and asymmetric demand cases is compared at different numbers of tree paths. At less than 400 number of tree paths, the computational time of both cases are comparable. However, when the number of tree paths increase, the computational time is much larger when demand is asymmetric. The optimal cost comparison is also made for symmetric and asymmetric demand as shown in Figure 5.9. The total capacity cost of routing asymmetric demand is clearly demonstrated to be larger than when demand is symmetric. For symmetric demand, traffic tends to spread out equally in both directions across the network. Under the same load, when demand is asymmetric, traffic in one direction may be much larger, and may deplete capacity at several links. As a result, some demand may have to take a longer route to avoid such link which, in fact, requires more bandwidth and higher cost.

Although, it has been shown so far that the computational time of the optimization procedure using the branch and bound algorithm increases when the candidate tree paths increases, in some cases, randomness in the computational time has been observed. This is due to the fact that the node fathoming process in the branch and bound algorithm depends on an estimated lower bound and a best solution found in each branch. Under different search spaces, this process responds differently based on the internal heuristics applied. Therefore, few cases show that the computational time sometimes does not necessary increase with a larger size of tree paths.

5.3.2. Effect of Increasing the Number of VPNs

The analysis is further conducted to study the effect of increasing the number of VPNs under different traffic load scenario. In this study, symmertric and asymmetric demand of multiple VPNs is generated and the link capacity limit is assigned based on a demand matrix to represent the correct load value. The load level is varied from light to heavy load within the values such that a feasible solution can still be found for each network topology. A complete numerical results is listed in Appendix A. Only selected cases will be illustrated to complete a discussion.

Figure 5.10 shows the average computational time of multiple random symmetric demand sets over the 10-node network at 0.3 load. The ranges show the minimum and maximum CPU time. The CPU time is shown to increase exponentially with the number of VPNs. Besides, the gap between the minimum and maximum CPU time recorded is found to be increasing when more random demand pairs are generated. The optimal cost, however, increases at a linear scale (see Figure 5.11). Different rate of cost increasing has been observed when the load is varied from 0.3 to 0.9. At a light load (ρ=0.3), this is quite predictable since capacity at many links has still not been fully utilized and traffic could still be routed over an optimal tree spanning over a group of edge nodes. Nevertheless, at a high load (when link capacity is smaller at ρ=0.9), routing the same demand traffic over a non-optimal route (i.e. a longer route) introduced extra bandwidth requirement and cost. For asymmetric demand, the cost increment becomes more significant as shown in Figure 5.12.

Computational time is also compared at various load levels for symmetric demand in Figure 5.13. Slightly increase in CPU time is observed when load changes from 0.3 to 0.6. On the contrary, at 0.9 load, considerable increase in CPU time becomes significant. In case

of Asymmetric demand (shown in Figure 5.14), the CPU time increment become even much larger in scale. The difference in computational time for symmetric and asymmetric demand is clearly illustrated in Figure 5.15. As the number of VPNs increases, the CPU time taken to find the optimal solution for asymmetric demand is much larger in scale than when demand is symmetric.

In addition, it further demonstrates that, when laying out more number of VPNs, the amount of additional capacity cost becomes larger. The optimization procedure also found it more difficult to simultaneous solve for the best optimal route configuration under a high load scenario, especially, when many VPNs are being considered. Therefore, a computational time is significantly increased comparing to a low load scenario.

Overall, when demand is asymmetric, a large cost increase and longer computational time is observed especially at a high load scenario. Asymmetric demand has been found to not be able to achieve a good gain of bandwidth aggregation, when compared to symmetric demand, and, thus, results in higher bandwidth usage. When a network is highly loaded, symmetric demand may have a larger amount of traffic sent in one direction than the other and causes link congestion. Therefore, demand may take a non-optimal route and introduces even more bandwidth usage especially at a high load. This effect is more pronounced as the number of VPNs increases.

As before, when a large number of VPNs are simultaneously deployed on sink-tree paths, an increase in the computational time has been observed in cases of the 55 and 15-node networks as shown in Figure 5.16 and 5.17, respectively. Note that, in some cases, under a high load scenario, the branch and bound search had exhausted the memory available and could not find the true optimal solution. The results therefore only represent the best

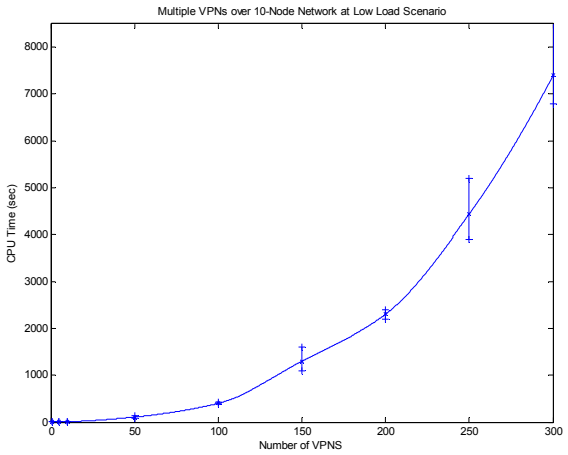


Figure 5.10 : Average Computational Time, the 10-node network (Symmetric Demand)

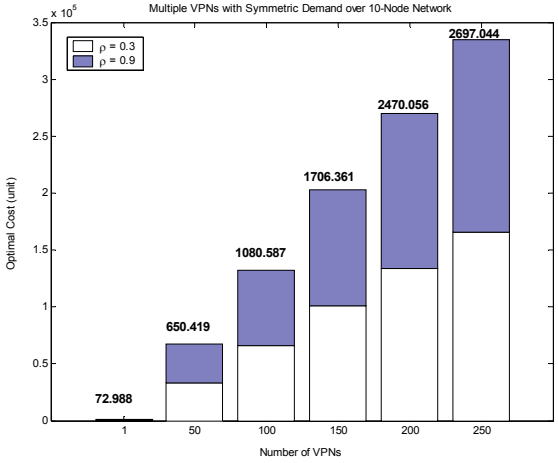


Figure 5.11 : Optimum Cost Comparison overthe 10-node network at Different Load (Symmetric Demand)

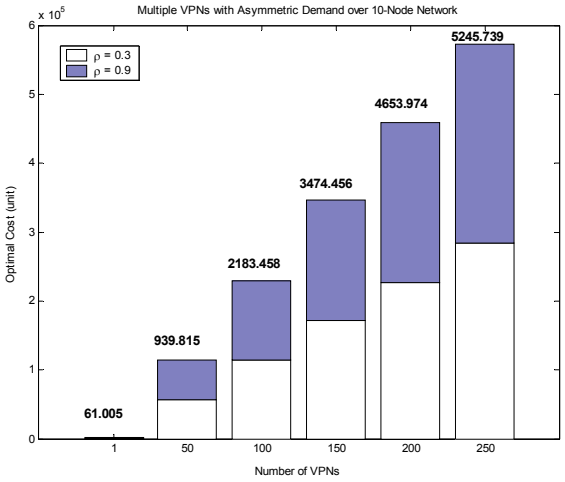


Figure 5.12 : Optimum Cost Comparison at Different Load over the 10-node network (Asymmetric Demand)

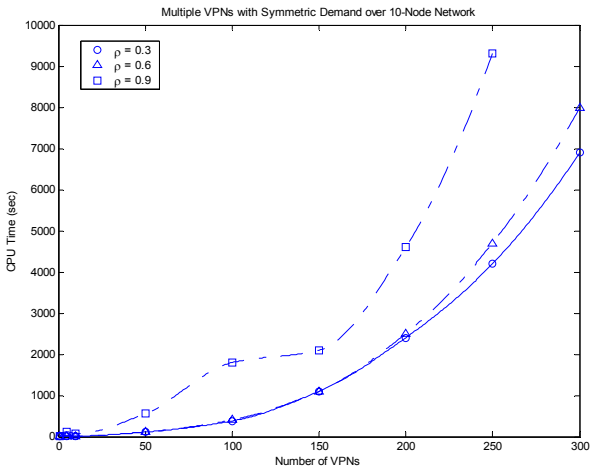


Figure 5.13 : Computational Time over the 10-node network at Different Load (Symmetric Demand)

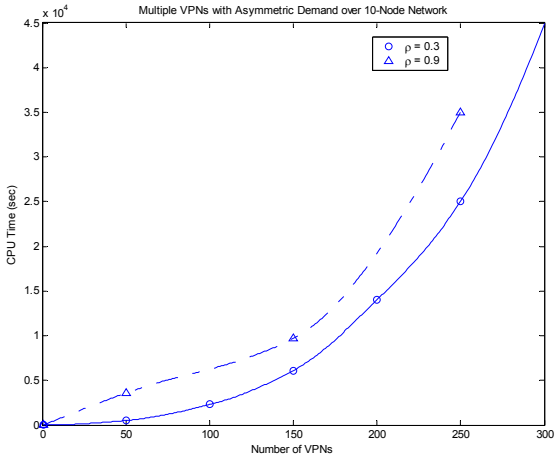


Figure 5.14 : Computational Time over the 10-node network at Different Load (Asymmetric Demand)

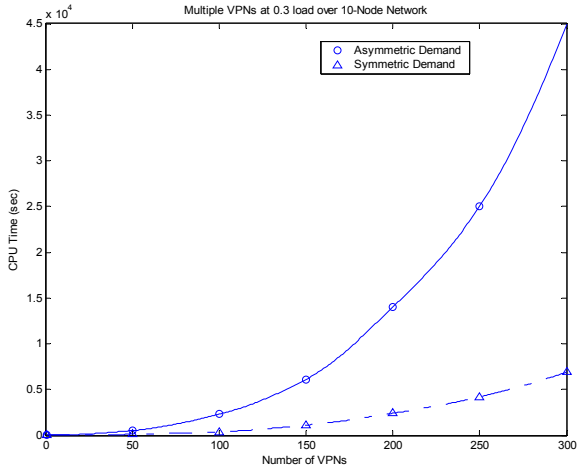


Figure 5.15 : Computational Time Comparison of Symmetric and Asymmetric Demand over the 10-node network

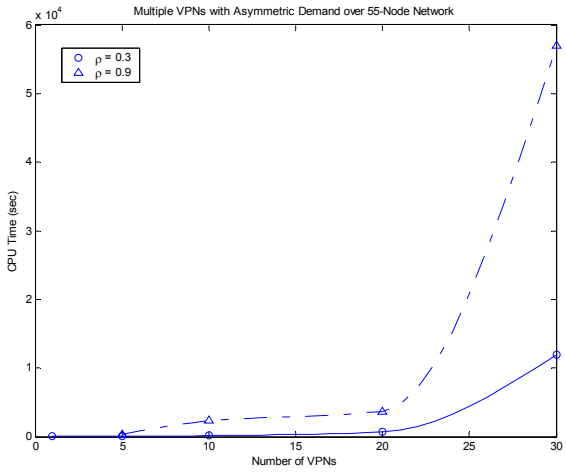


Figure 5.16 : Computational Time over the 55-Node Network at Different Load (Asymmetric Demand)

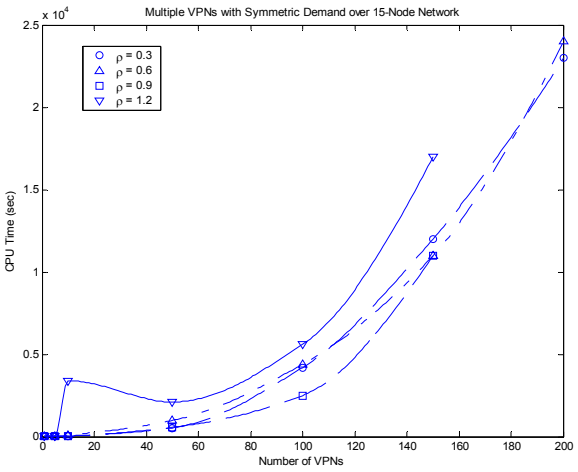


Figure 5.17 : Computational Time over the 15-node Network at Different Load (Symmetric Demand)

integer solution found (see Appendix A.) At 0.9 load, when the number of VPNs increases from 20 to 30 VPNs, the computational time has dramatically increased by nearly 300 percent. Due to memory limitations, for the 55-node network case, the number of VPNs can only be varied up to 30 VPNs when the branch and bound technique is applied. This is also the case for the 15-node network, the number of VPNs can be varied up to 200 VPNs at a high load. These results imply that using a standard optimization approach to solve a VPN design problem utilizing a sink-tree paths may have a limit applicability in practice when a large number of VPNs are taking into account.

5.4. Complexity of Multiple VPNs Design

The running time of the heuristic path selection algorithm includes the time spent in the node and link elimination process, tree generation process and tree ranking process. In node and link elimination process, each node is visit one-by-one and checked to see if the node and its associated links should be eliminated from the network. Therefore, this process only takes

$\Theta(N)$ time for a *N*-node network. The tree generation process is done based on a decision-tree search principle which completely enumerates all possible spanning trees without duplications. After all distinctive trees are generated, to check the hop-count limit, each tree must be traversed from any root node to all other nodes in the tree. Typically, the tree traversals could be done in $\Theta(N)$. Therefore, if there are total of ***v*** number of distinctive trees, the hop-count constraint can be verified within $\Theta(v \cdot N)$ time. Subsequently, the tree ranking process will select a fixed size of tree set based on the number of links in a tree. Specifically, only ***u*** numbers of trees with the smallest number of links are selected from all feasible tree of size ***w***. In the implementation, Quicksort is used to select such trees and the worst case performance for the selection process is $\Theta(w^2)$. However, on the average, the selection could be done in $\Theta(w)$. The complexity of the tree generation process depends largely on the tree creation procedure which can be run in polynomial. Note that the heuristics tree selection algorithm could be done from time to time for a set of the all distinctive trees which could be generated once and reused for the network of interest. Therefore, given a set of all distinctive trees, the tree selection heuristics can be run in $\Theta(w^2)$.

5.5. Conclusion

In this chapter, a tree selection heuristic is proposed to scale the VPNs design problem when sink-tree paths are utilized. The problem size is reduced by selecting a good candidate set of tree paths based on the fact that optimum trees tend to have less number of links in them. In the proposed heuristic, a sink-tree set is size-limit, which results in ranking trees based on the number of links used in a tree. Numerical results clearly prove the benefit of the tree selection heuristics in reducing the candidate set of sink-tree paths to be searched over and allowing the

optimal solution to be obtained for a larger problem size. Nonetheless, when a large number of VPNs are considered to be layout simultaneously, the standard optimization approach is not practical, and, therefore, other solution approaches need to be found.

6. Solution Algorithms

Although, the complexity of the VPNs design model utilizing sink-tree paths could be reduced using a heuristics path selection algorithm, when a large number of VPNs are taken into account, solving the mathematical model using standard approach still consumes a considerable amount of time considering that the VPN planning and management must be done frequently. In order to efficiently map a large number of traffic demands from many different VPNs onto an existing network topology while maintaining the optimal use of network resources in all time periods, a solution algorithm using a heuristics-based approach must be employed. In this chapter, a new heuristic termed the *Minimum-Capacity Sink-Tree Assignment* (***MCSTA***) is proposed to approximate the optimal bandwidth and sink-tree route assignment for multiple VPNs. The *MCSTA* algorithm and its variations are demonstrated to give a good approximation and sometimes yields the exact solution within a polynomial computational time.

The VPNs design utilizing sink-tree paths modeled as a mixed-integer programming problem falls into the class of many well-know problems in combinatorial optimization which are NP-complete to solve exactly. The problem involves enumerating on a large set of trees spanning over a group of edge nodes while optimizing for minimum bandwidth cost. This is equivalent to *the Steiner tree problem* which aims to find a minimum-cost tree connecting a subset of nodes. The Steiner tree problem is known to be NP-hard [45, 81]. A large body of literature exits on solution approaches for Steiner tree problems and its variants. Approaches include complete enumeration, dynamic programming, and branch-and-bound techniques in addition to various heuristic procedures.

One of the outstanding open questions in computational complexity theory has been "NP=P?", i.e., whether the class of NP-complete problems admits a polynomial-time solution. The class of NP-complete problems represents a colossal collection of problems for which no polynomial-time algorithms are known to solve for an exact solution, including the Steiner tree problem. The NP-hardness of such problems can be overcome only by compromising the quality of the solution for the sake of being able find to sub-optimal solution within a reasonable computation time. Many approximation algorithms for optimization problems, therefore, seek to find a solution with a small multiplicative error over all instances within polynomial time according to the problem size. However, no polynomial-time approximation algorithm has a substantially better performance ratio unless something nearly equivalent to P=NP is true. In other words, finding approximation algorithms for such problems with considerably better performance ratios is tantamount to nearly solving the "P=NP?" question.

6.1. Capacity and Route Assignment Criteria

Over an infrastructure network in which multiple sink-tree routing paths are simultaneously optimized for multiple coexisting VPNs, a bandwidth saving over an aggregated link on a sink-tree path can be significant as compared to routing a demand over a traditional point-to-point path. When a group of traffic demands, going in the same direction, is routed on a sink-tree path ending at one egress router, both route and capacity assignment for such demand should be done differently than a traditional full-mesh design utilizing a point-to-point paths.

An optimal layout and capacity assignment for multiple sink-trees found using a branch and bound technique, over a small scale problem, reveal notable features of the optimal-tree layout pattern. First, optimal sink-trees promote bandwidth aggregation by using many links in

common. Therefore, a tree-layout prefers to have large capacity allocated on less number of links rather than having many links with small capacity. Secondly, a large demand was shown to be routed on a short-hop path over part of a minimum spanning tree when the bandwidth cost of all links are equal. Finally, a small demand is likely to be routed on a long-hop path where demand heading to the same destination will be merged and aggregated on large capacity links. As a result, a small demand may take a slightly longer route than a typical shortest path. Note that the above observation is made given that the capacity cost is equal at all links.

This chapter, hereby, refers to a link on a minimum spanning tree as a "Spine Link". Thus, the refined objective in routing VPNs demand using multipoint-to-point LSPs is to aggregate as much as possible demands on "Spine Links". A smaller demand may be aggregated over it or routed around it. This leads to a significant problem in sink-tree route ordering for multiple VPNs demand. Intuitively, a large demand, which is likely to consume more resources, should be laid out first. The idea can be viewed as letting large flows go through main pipes having large capacity while leaving small flows to go through pipes having some left-over capacity.

Taking into account of such patterns, the Minimum-Capacity Sink-Tree Assignment (*MCSTA*) algorithm is proposed to approximate the optimal bandwidth and sink-tree route assignment for multiple VPNs. The key aspect of the *MCSTA* algorithm is to route each sink tree based on a sum of demand offered. Demand within each VPN is grouped based on its destination to be routed over a sink-tree path. A group of demand heading to the same destination which has the largest sum of bandwidth demand will be routed first. Each demand pair within the same group will also be routed based on its capacity. The largest demand is first routed over a shortest possible path. However, the *MCSTA* algorithm will choose to send the demand over spine links

on the minimum spanning tree if the cost can be justified. If the source and destination nodes of the demand are directly connected via a spine link, a demand will be sent through a spine link. For each pair of demand to be routed, an aggregated cost (incremental cost) will be recalculated at all links. Note that an incremental cost can be calculated on each link given that a demand can be multiplexed with an existing connection being in place on that link otherwise an incremental cost is merely the cost of a whole demand bandwidth at that link. Subsequently, a successive shortest path routing is performed, while, in every iteration, the cost on a minimum-cost path will be compared against the cost of routing the demand over spine links on minimum spanning tree route. When an incremental cost can be justified, the route on spine links is prefer over a shortest path.

6.2. Minimum-Capacity Sink-Tree Assignment (*MCSTA*) Algorithm

Given an infrastructure network G of N nodes, where $M \subseteq N$ are edge nodes having demand entering and exiting. Each network link $l \in L$ is associated with capacity cost ψ_l (measured in units of $ per Mbps), link capacity C_l and a utilization factor α_l which limits the proportion of link capacity C_l to be allocated for VPN traffic.

Demand matrices of all VPNs are grouped based on the egress node such that multiple demands flowing toward the same destination are routed together. Let D^v represents a VPN demand matrix $v \in V$ which can be written as

$$D^v = \begin{bmatrix} d^v_{1,1} & d^v_{1,2} & d^v_{1,3} & d^v_{1,4} & \cdots & d^v_{1,m} \\ d^v_{2,1} & d^v_{2,2} & d^v_{2,3} & d^v_{2,4} & \cdots & d^v_{2,m} \\ d^v_{3,1} & d^v_{3,2} & d^v_{3,3} & d^v_{3,4} & \cdots & d^v_{3,m} \\ d^v_{4,1} & d^v_{4,2} & d^v_{4,3} & d^v_{4,4} & \cdots & d^v_{4,m} \\ \vdots & \vdots & \vdots & \vdots & \ddots & d^v_{5,m} \\ d^v_{m,1} & d^v_{m,2} & d^v_{m,3} & d^v_{m,4} & \cdots & d^v_{m,m} \end{bmatrix}_{M\ x\ M} \tag{6.1}$$

D^v is a $M\ x\ M$ matrix which $d^v_{m.k}$ is a directional demand of VPN $v \in V$ entering at ingress node $m \in M$ and exiting at egress node $k \in K$ ($K \subseteq M$). Then, D^v_k is a column vector of point-to-point demand of VPN $v \in V$ destined to egress node $k \in K$.

$$D^v_k = \begin{bmatrix} d^v_{1,k} \\ d^v_{2,k} \\ d^v_{3,k} \\ d^v_{4,k} \\ \vdots \\ d^v_{m,k} \end{bmatrix} \tag{6.2}$$

B^v_k is defined as the aggregated bandwidth of demand vector D^v_k

$$B^v_k = d^v_{1,k} + d^v_{2,k} + d^v_{2,k} + d^v_{4,k} + \cdots + d^v_{m,k} \tag{6.3}$$

or

$$B^v_k = \sum_{m \in M} d^v_{m,k} \tag{6.4}$$

The egress demand vector D^v_k will be sorted in a decreasing order of the aggregated bandwidth B^v_k. The egress vector D^v_k having the largest B^v_k will be the first to be routed. Each demand pair within a demand D^v_k will also be selected to be route based on its $d^v_{m.k}$ bandwidth in decreasing order of bandwidth.

Minimum tree spanning over *M* edge nodes are determined. Let $SP_{m,k}$ be a set of links on a minimum spanning tree path between edge node *i* and *j* such that

$$s_l = \begin{cases} 1 & if\ SP_{m,k}\ includes\ link\ l \in L \\ 0 & otherwise \end{cases} \tag{6.5}$$

Let $\beta_{k,l}^v$ represent a bandwidth assignment of demand D_k^v over link $l \in L$. For each demand pair $d_{m.k}^v$ in D_k^v, an incremental cost $(\nabla \theta_{k,l}^v)$ of carrying a demand $d_{m.k}^v$ is calculated on all links $l \in L$.

$$\nabla \theta_{k,l}^v = \{\, eqv(\beta_{k,l}^v + d_{m,k}^v) - eqv(\beta_{k,l}^v) \,\} \cdot \psi_l \tag{6.6}$$

Function $eqv(\cdot)$ represents the mapping of offered demand bandwidth to an effective bandwidth allocated over a network link as described previously in Chapter 4. Note that if the bandwidth to be assigned on link $l \in L$ exhaust the maximum allowable capacity $\alpha_l \cdot c_l$, then a large penalty cost Ω is assigned so that the link is avoided.

The total cost (ε_k^v) of carrying demand $d_{m.k}^v$ over a minimum spanning tree is, therefore, determined by

$$\varepsilon_k^v = \sum_{s_l \in SP_{m,k}} (\nabla \theta_{k,l}^v \cdot s_l) \tag{6.7}$$

Let $MCP_{m,k}$ denote the minimum cost path from source node m to egress node k. A minimum-cost path $MCP_{m,k}$ is determined based on an incremental cost $\nabla \theta_{k,l}^v$ matrix. The total incremental cost (τ_k^v) of routing $d_{m.k}^v$ over a minimum-cost path is given by

$$\tau_k^v = \sum_{h_l \in MCP_{m,k}} (\nabla \theta_{k,l}^v \cdot h_l) \tag{6.8}$$

where,

$$h_l = \begin{cases} 1 & if\ MCP_{m,k}\ includes\ link\ l \in L \\ 0 & otherwise \end{cases}$$

Initialization :

For all VPNs $v \in V$, links $l \in L$, egress node $k \in K$ ($K \subseteq M$)

{ $\beta_{k,l}^{v} = 0$;

$D_k^v = D^v \text{ x } \Lambda^{\mathrm{T}}$ where $\Lambda = [\Lambda_1\ \Lambda_2\ \Lambda_3 \ldots \Lambda_i]_{1\text{xK}}$

$\Lambda_i = 1$ if $i = k$, otherwise $\Lambda_i = 0$

}

For all node pairs (i, k)

Find a minimum spanning tree path $SP_{i,k}$ between node pair (i, k)

Begin :

Let $\Phi = \{ D_{k_1}^{v_1}, D_{k_2}^{v_2}, D_{k_3}^{v_3}, \ldots, D_{k_i}^{v_i} \}_{1\text{x(M·V)}}$

such that $B_{k_1}^{v_1} \geq B_{k_2}^{v_2} \geq B_{k_3}^{v_3} \geq \ldots \geq B_{k_i}^{v_i}$ where $B_k^v = \sum_{m \in M} d_{m,k}^v$

For each D_k^v in Φ

{ Let $\Theta = \{ d_{i_1,k}^v, d_{i_2,k}^v, d_{i_3,k}^v, \ldots, d_{i_m,k}^v \}_{1\text{xM}}$ such that $d_{i_1,k}^v \geq d_{i_2,k}^v \geq d_{i_3,k}^v \geq \ldots \geq d_{i_m,k}^v$

For each $d_{i,k}^v$ in Θ

{ Let $\varepsilon_k^v = 0$;

For all link $l \in L$

{ /* *Calculate an incremental bandwidth and cost at all link* */

$\nabla\beta_{k,l}^v = \{ eqv(\beta_{k,l}^v + d_{m,k}^v) - eqv(\beta_{k,l}^v) \}$;

/* *Check the capacity constraint* */

If $\sum_{\forall i \in \{K-k\}} \sum_{\forall v} \beta_{i,l}^v + \nabla\beta_{k,l}^v \quad \geq \quad \alpha_l \cdot c_l$

$\nabla\theta_{k,l}^v = \Omega$;

Else

$\nabla\theta_{k,l}^v = \nabla\beta_{k,l}^v \cdot \varphi_l$;

/* *Calculate an incremental cost on a minimum spanning tree* $SP_{i,k}$ *path* */

$\varepsilon_k^v = \varepsilon_k^v + \nabla\theta_{k,l}^v \cdot s_l$;

}

Cont.

Figure 6.1 : The *MCSTA* Algorithm

/*Choose a direct link on a minimum spanning tree*/
If exist link (*i*, *k*) in $SP_{i,k}$

$\beta_{k,l}^{v} = \beta_{k,l}^{v} + \nabla\beta_{k,l}^{v}$;

Else

{ /* Calculate total incremental cost on a minimum cost path */

Find a minimum cost tree path $MCP_{i,k}$ between (*i*, *k*) using Dijkstra's shortest path algorithm;

Let $\tau_k^v = \sum_{h_l \in MCP_{m,k}} (\nabla\theta_{k,l}^{v} \cdot h_l)$;

/* Choose minimum spanning tree path */

If $\varepsilon_k^v \le \tau_k^v$

For each link *l* in $SP_{i,k}$, let $\beta_{k,l}^{v} = \beta_{k,l}^{v} + \nabla\beta_{k,l}^{v}$;

/* Choose minimum incremental cost path */

Else

For each link *l* in $MCP_{i,k}$, let $\beta_{k,l}^{v} = \beta_{k,l}^{v} + \nabla\beta_{k,l}^{v}$;

}

}

End

Figure 6.2 : The *MCSTA* Algorithm (Cont.)

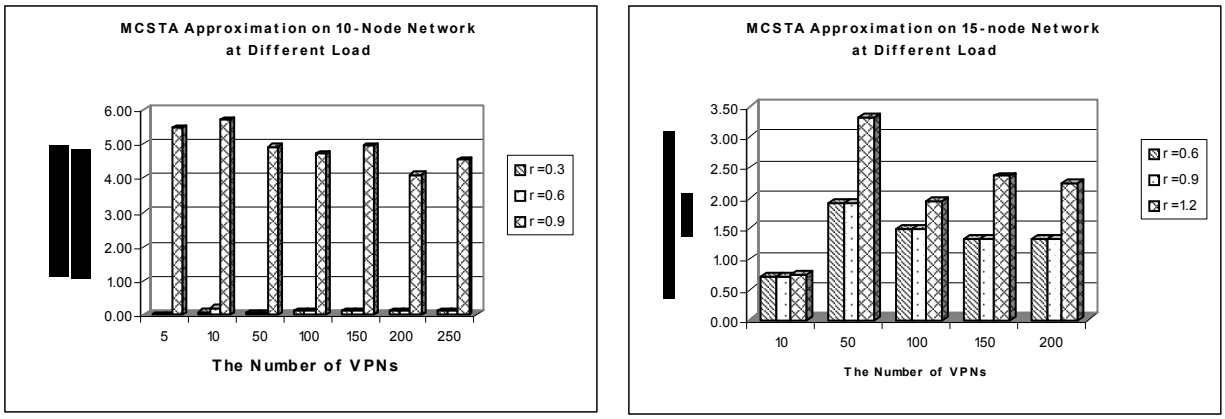
The demand $d_{m.k}^{v}$ is routed on a direct link with the total incremental cost of $\nabla\theta_{k,l}^{v}$, when there exists a direct link between ingress node *m* and egress node *k* which is on a minimum spanning tree $SP_{m,k}$. Otherwise, a minimum spanning tree route is preferred over a minimum cost route. Therefore, if $\varepsilon_k^v \le \tau_k^v$, a minimum spanning tree path $SP_{m,k}$ will be selected but if $\varepsilon_k^v > \tau_k^v$, demand $d_{m.k}^{v}$ will be routed over a minimum cost path $MCP_{m,k}$. The bandwidth usage $\beta_{k,l}^{v}$ over each link is then updated for each demand pair in D_k^v before a successive demand routing is continued. The complete steps of the *MCSTA* algorithm are described in Figure 6.1.

6.3. Performance Analysis of the *MCSTA* Algorithm

The goodness of the *MCSTA* algorithm is evaluated by comparing the solution approximated by the *MCSTA* algorithm with the optimal solution obtained from the problem formulation of Chapter 4 with the tree heuristics of Chapter 5. The comparison will also be made on a computational time which is measured in term of CPU real-time in seconds. The ILP solution is obtained by running CPLEX 7.1 InP solver on a Sun Blade1000 workstation with 750 MHz processor and 2 gigabytes of memory. The *MCSTA* algorithm runs on a standalone PC with Intel Pentium IV 2.6 GHz processor with hyper-threading technology and 768 Mbytes of memory. Different traffic scenarios including symmetric and asymmetric demand and load values are generated over several networks. In this study, the link capacity is selected according to the total traffic demands offered to the network to represent the correct load value. Complete numerical results of the study are listed in Appendix B. Sample results are selected to establish a point of discussion.

Table 6.1 illustrates the *MCSTA* cost in comparison with the optimal cost at different numbers of VPNs on the 10-node network. At 0.6 load, the *MCSTA* cost clearly be seen to closely approximate the optimal solution across different number of VPNs. A percentage of relative error less than 1 percent can be demonstrated. At a small number of VPNs, an exact solution can be found. When the network is highly loaded, the relative error ranges from 4 to 6 percent. Notice that, the relative error does not vary much with the number of VPNs. This is considered to be another desirable characteristic of the *MCSTA* algorithm in its capability to consistently produce a solution within a small relative error even for a large number of VPNs.

Figure 6.3 (a) and (b) show the percentage of relative error at various load levels on networks of 10 and 15 nodes when the demand is symmetric. Note that the extreme load values of 0.9 and 1.2 are deliberately applied to the cases of the 10 and 15-node networks. This is to



(a) (b)

Figure 6.3 : The *MCSTA* Approximation at Different Load Levels

evaluate the performance of the algorithm when the capacity of several network links is driven to a saturated point. At the high load, for the 10-node network, the relative error is at less than 6 percent, but a smaller error of less than 3.5 percent is observed for the 15-node network. At small to medium load, the *MCSTA* performs relatively well especially for a small number of VPNs. It can be said that *MCSTA* performance is closely related to the traffic load and network size. Table 6.2 shows the *MCSTA* approximation for the 55-node network carrying symmetric demand. At 0.6 load, the percentage of relative error is fairly small.

At 0.9 load, the relative error does increase but remain relatively small compared to cases of the 10 and 15-node networks. For a large and sparse, the *MCSTA* can give a close approximation even at a high load situation. However, for 5 VPNs, the *MCSTA* can not establish a feasible solution. It is not very often that over approximation of the *MCSTA* algorithm causes an infeasible set of route assignment. Under asymmetric demand, selected results at 0.9 load are summarized in Table 6.3. In almost all cases, the relative error is found to be within 4 percent for the 10-node network except for a single VPN case where the demand ordering does not have any

Load	# VPNs	Optimal Cost	MCSTA Cost	%Error	ILP CPU Time(sec)	MCSTA CPU Time (sec)
0.6	5	3,357	3,357	0.00	0.52	0.14
	10	6,624	6,639	0.23	8.5	1.50
	100	65,618	65,708	0.14	410	15.43
	200	133,679	133,869	0.14	2500	31.14
	250	165,917	166,153	0.14	4700	38.54
0.9	5	3,476	3,665	5.45	72	0.85
	10	6,770	7,156	5.70	120	1.92
	100	66,699	69,841	4.71	1800	19.39
	200	136,147	141,733	4.10	4600	39.78
	250	168,614	176,279	4.55	9300	50.12

Table 6.1 : *MCSTA* Approximation on the 10-node network with Symmetric Demand at 0.6 Load

Load	# VPNs	Optimal Cost	MCSTA Cost	%Error	ILP CPU Time(sec)	MCSTA CPU Time (sec)
0.6	1	2,909	2,909	0.00	2	0.14
	5	11,518	11,506	0.10	42	0.76
	10	25,439	25,431	0.03	24	1.62
	20	53,869	54,153	0.53	210	3.76
	30	79,214	79,070	0.18	1200	5.14
0.9	1	inf	inf	-	-	-
	5	11,711	inf	-	3600	-
	10	25,568	26,120	2.16	2200	1.70
	20	53,962	54,532	1.06	4700	3.87
	30	79,362	80,010	0.82	5000	5.53

Table 6.2 : *MCSTA* Approximation on the 55-node Network with Symmetric Demand

Network	# VPNs	Optimal Cost	MCSTA Cost	%Error	ILP CPU Time(sec)	MCSTA CPU Time (sec)
10	1	1,096	1,167	6.46	2.40	0.26
	50	57,901	60,011	3.64	3,600	16.65
	100	115,911	120,338	3.82	31,000+	34.51
	150	174,814	181,429	3.78	9,700	49.82
	200	231,839	240,214	3.61	31,000+	65.67
15	1	1,794	1,830	1.97	11.00	0.29
	50	83,979	85,070	1.30	2,100	15.81
	100	168,036	169,904	1.11	20,000	32.01
	150	128,832	130,268	1.11	8,700	23.50
55	1	inf	inf	-	-	-
	5	11,113	11,351	2.14	280	0.75
	10	20,183	20,365	0.90	2300	1.42
	20	43,605	44,295	1.58	3700	3.00
	30	71,858	72,097	0.33	57000	4.43

Table 6.3: *MCSTA* Approximation for Asymmetric Demand at 0.9 Load

effect on the route assignment of the *MCSTA*. For the 15 and 55-node networks, the relative error is less than 2 and 2.5 percent respectively. Thus, the *MCSTA* performs relatively well even when the demand is asymmetric. Once more, the computational time of the *MCSTA* increases linearly with the number of VPNs over the same network. Overall, the *MCSTA* performance is somewhat remarkable in being able to give a good approximation with a small relative error even at a large number of VPNs, within a short amount of computational time.

6.4. Variations of the *MCSTA* Algorithm

The aggregated bandwidth B_k^v of demand group D_k^v used in the *MCSTA* algorithm, to select a group to be routed, is an indication of the amount of network bandwidth or resources that demand group D_k^v will use. Demand group D_k^v which has a large sum of the aggregated bandwidth B_k^v is allowed to use a large capacity on all spine links. There are several variations on the *MCSTA* algorithm that can be developed by considering the resource usage level of each demand group. A resource usage index $\Delta_{m,k}^v$ of demand pair (m, k) of $v \in V$ VPN is defined as

$$\Delta_{m,k}^v = d_{m,k}^v \cdot \mathrm{A}_{m,k} \tag{6.9}$$

where, $A_{m,k}$ is the average path length (measured in hops) between node pair (m, k). The index $\Delta_{m,k}^v$ is another measure of the estimated resource consumption when a demand pair (m, k) is admitted into a network. Hence, when a demand group D_k^v is admitted, the amount of resource usage can be measured by Δ_k^v as

$$\Delta_k^v = \sum_{m \in M} \Delta_{m,k}^v \tag{6.10}$$

One variation of the original *MCSTA* algorithm is to use the Δ_k^v index to indicate the ordering of the demand group D_k^v. This approach is called the *MCSTA_APL* when the Δ_k^v index is measured using the average path length.

Another measure of resource usage is to apply a shortest path length instead of an average path length between node pair (m, k). Therefore $\Delta_{m,k}^v$ is then calculated by

$$\Delta_{m,k}^v = d_{m,k}^v \cdot S_{m,k} \tag{6.11}$$

where, $S_{m,k}$ is a shortest path length (measured in hops) between node pair (m, k). In this case, Δ_k^v index for demand group D_k^v can be derived as in equation 6.10. Hence, another variation of the *MCSTA* algorithm is to perform a route ordering based on Δ_k^v index which is measured by a shortest path length, and therefore, is called the *MCSTA_SPL* algorithm.

In both *MCSTA_APL* and *MCSTA_SPL* algorithms, a demand group D_k^v is first selected using Δ_k^v index. Then, each pair demand $d_{m,k}^v$ within the group will also be routed in the order based on $\Delta_{m,k}^v$ index. Hence, when $\Delta_{m,k}^v$ index is used to determine a route ordering, a large demand between node pair that is far apart is allowed to take the first best route.

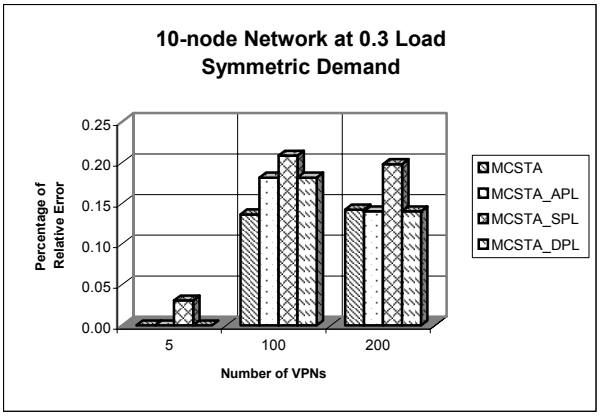
The last variation of the *MCSTA* algorithm is to preserve the use of the aggregated bandwidth B_k^v as a factor to determine ordering for demand group D_k^v, but a route order within the same group is indexed by $\Delta_{m,k}^v$ which is calculate based on an average path length. This approach is referred to as the *MCSTA_DPL* algorithm.

The performance of different variations of the *MCSTA* algorithm are compared with the original *MCSTA* algorithm for various network sizes, load values, and demand characteristics. The percentage of relative error of the approximate solution to the optimal solution (obtained

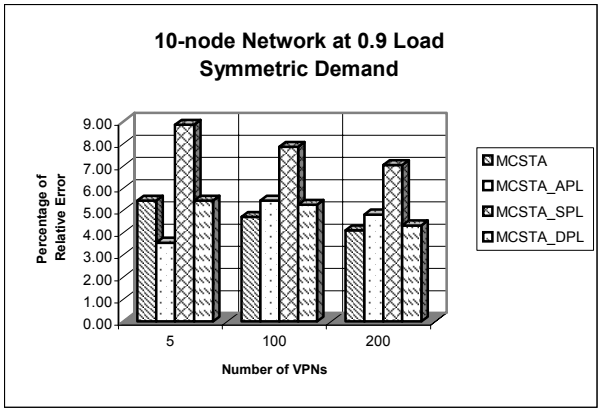
from solving ILP model based branch and bound technique) is used as a scale to compare all approximation algorithms. The complete numerical results for both symmetric and asymmetric demand are presented in Appendix C.

Figure 6.4 (a)-(f) shows a performance comparison of different algorithms under symmetric demand load. At a low load, all algorithms can give a good approximation within 2 percent of the relative error. Over the small network of 10 nodes at 0.3 load (shown in Figure 6.3 (a)), *MCSTA*, *MCSTA_APL* and *MCSTA_DPL* yield an exact solution with zero percent relative error for a small number of VPNs. Nonetheless, the relative error is likely to arise as the number of VPNs increases. The *MCSTA_SPL* seems to be the worst approximation among all and can not derive an exact solution even at a small number of VPNs. As the load increases from 0.3 to 0.9, the relative error increases. This is also the case for the 15 and 55-node networks at high and low load situations. Note that for the heavy load case, the error goes down as the network size increases. Across the cases considered, no one scheme is consistently the best with the *MCSTA* and *MCSTA_APL* typically performing the best.

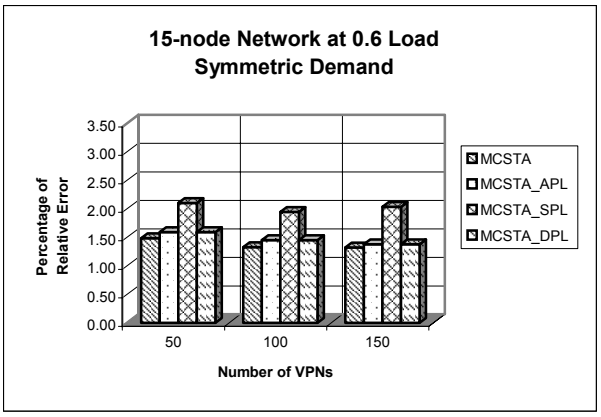
Unlike the *MCSTA_SPL*, the *MCSTA_APL* is reliably demonstrated to be able to establish a good error bound. It also can give an exact solution at a small number of VPNs. Even though, the *MCSTA_APL* generally provides a solution slightly farther from the optimum when compared to the *MCSTA*, it can be shown to outperform the *MCSTA* in several cases at the high load situation. Therefore, using a resource usage index based on average path length to determine the route ordering seems to work pretty well especially at the high load situation when demand is symmetric. By doing this, a node-pair separated by many hops away, carrying a large demand, is likely to use much bandwidth resources and will be the first to be packed onto a large pipe. This concept, therefore, has been illustrated in the *MCSTA_APL* cases. On the other hand, when the



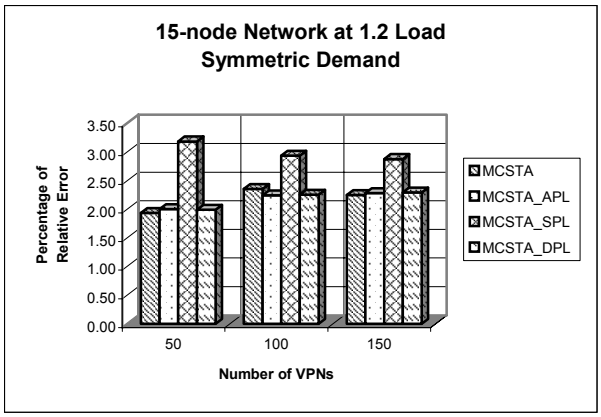
(a)



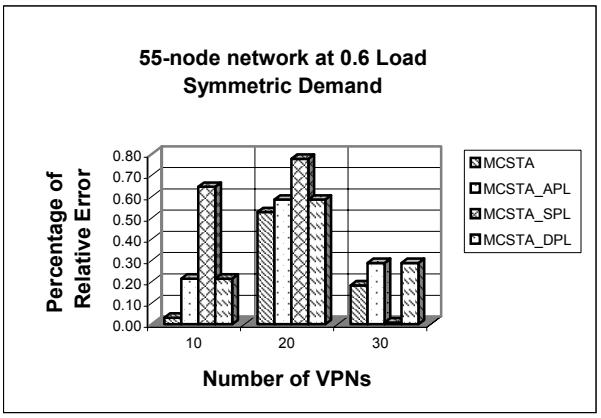
(b)



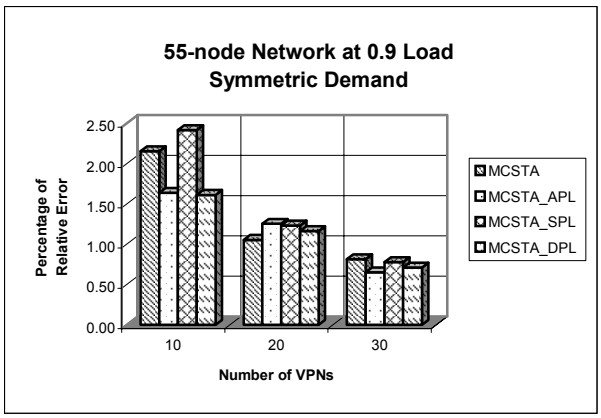
(c)



(d)

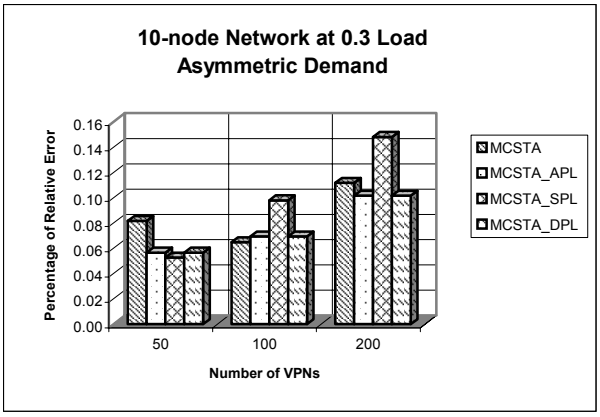


(e)

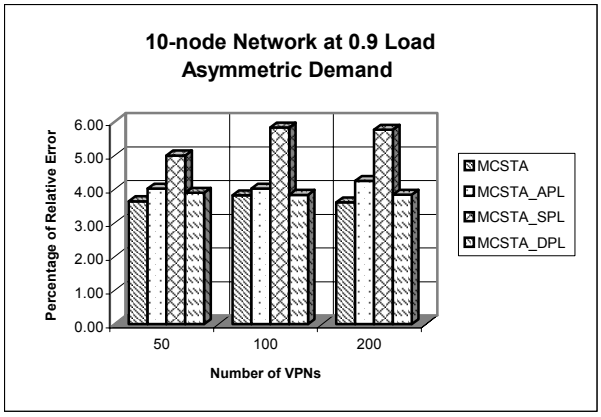


(f)

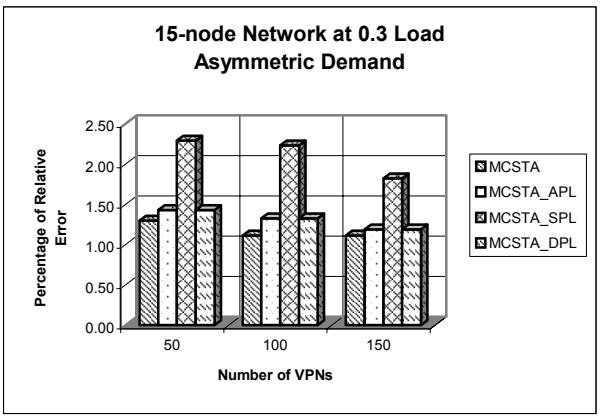
Figure 6.4 : Performance Comparison of Different Variations of *MCSTA* Algorithms (Symmetric Demand)



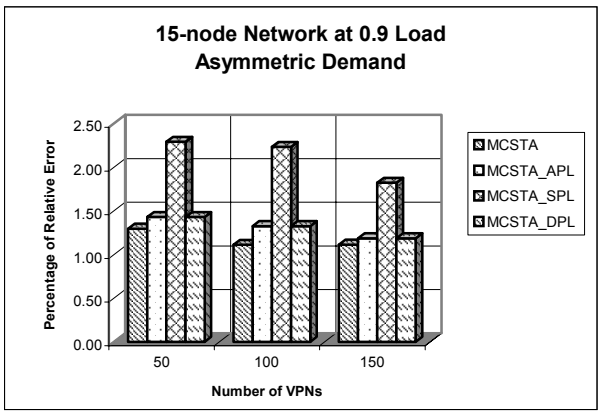
(a)



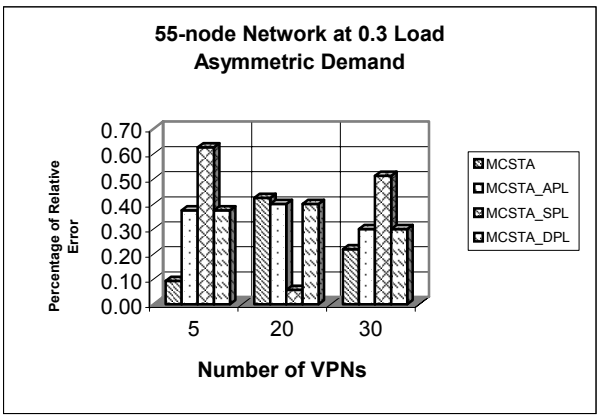
(b)



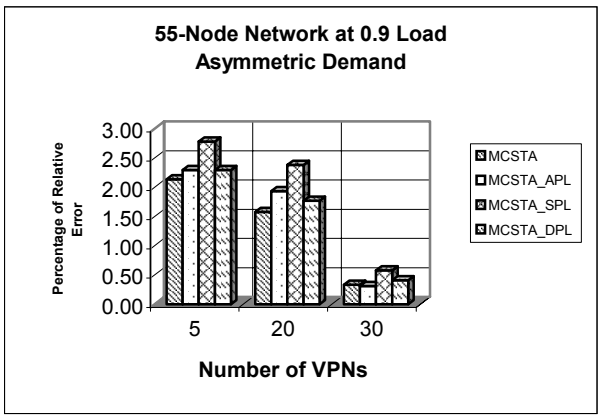
(c)



(d)



(e)



(f)

Figure 6.5 : Performance Comparison of Different Variations of *MCSTA* Algorithms (Asymmetric Demand)

resource usage index is calculated based on a shortest path in the *MCSTA_SPL*, a demand ordering pattern does not favor a saving in total bandwidth cost across all VPNs. However, for a large and sparse network of 55-node case, an average path length is relatively close to the shortest path length, therefore, the *MCSTA_SPL* performs fairly well in this scenario. Based on this observation, the shortest path, in many circumstances, is not a good measure to be used in determining a resource usage of each demand group. Combining the use of bandwidth aggregate and resource usage index based on average path length to determine the route ordering of a group demand and a pair-demand within the same group respectively, the *MCSTA_DPL* performs relative well in all most all cases but its performance still fall in between the *MCSTA* and *MCSTA_APL*.

For an asymmetric demand, a comparison of the different algorithms is shown in Figure 6.5 (a)-(f). When demand is asymmetric, traffic in both directions of a node-pair do not shares the same demand value, therefore, the load may not be balanced across all part of the network. The *MCSTA_SPL* shows somewhat similar performance as in symmetric demand case having the largest error in most cases. Under asymmetric demand, the *MCSTA_APL* does not seem to gain more advantage in term of bandwidth cost saving over the original *MCSTA* algorithm. The *MCSTA* algorithm seems to be quite robust and more predictable when compared to other approximation methods.

Disregarding the *MCSTA_SPL* approximation, a drawback of other approximations can be seen in the case of the 10-node network (shown in Figure 6.5 (a)-(b)). The error increases from less than 0.2 percent to about 4 percent range when the load increases to 0.9. This observation points out the fact that, at a high load situation, after a capacity on spine links of spanning trees has been completely occupied, the *MCSTA* algorithm does not efficiently route

the left-off demand through an optimal route. When one closely examines the steps in the *MCSTA* algorithm, in every iteration, the cost of routing a demand over a minimum-cost route given by a successive-shortest path computation is always compared with the cost of using the path passing through spine links. The path over spine links is always preferred and will be chosen first. When the capacity on a spine link has filled up, the route assignment will be determined solely by the successive shortest path computation which, by itself, can not establish a good gain in bandwidth aggregation. Therefore, the approximate solution turns out to be farther from the optimal when a network is highly loaded. In fact, this can be more clearly illustrated by example in the case of the 15-node network. Initially, the spine links on a minimum spanning tree are chosen for all edge nodes as shown in Figure 6.6 (a). Large demands between the edge

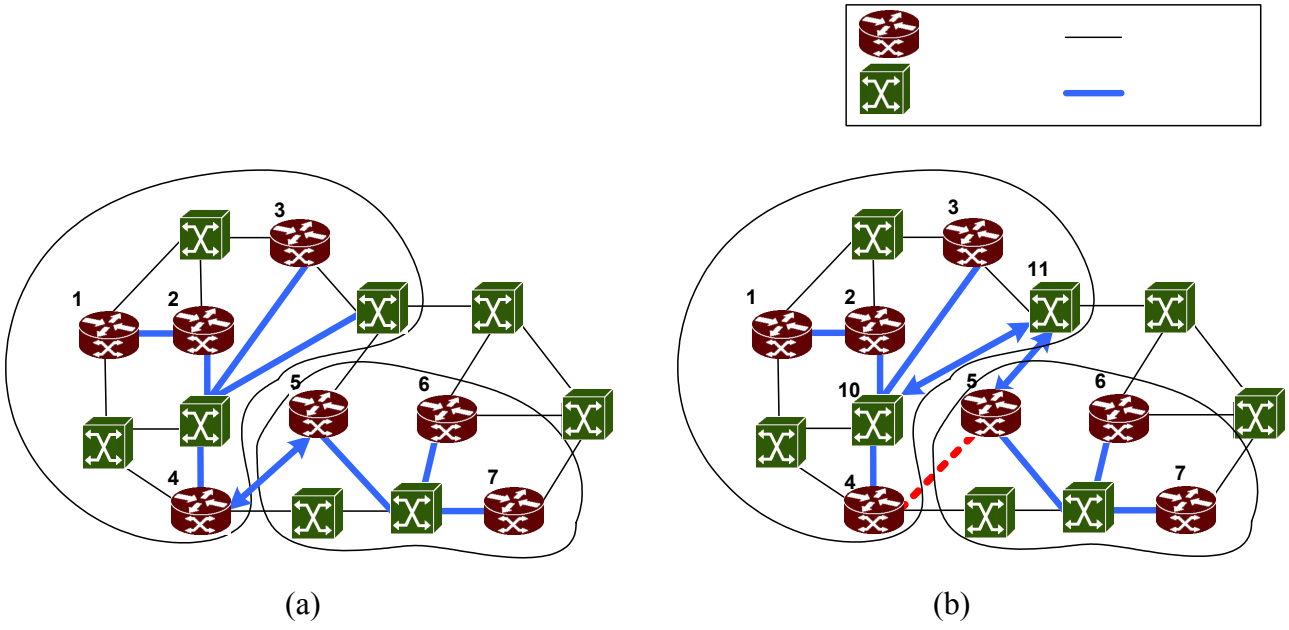


Figure 6.6 : Spine Links Selected over a Minimum Spanning Tree

nodes {1, 2, 3, 4} and {5, 6, 7} must cross link 4-5. At a high load scenario, when the capacity on link 4-5 is depleted, demand can not be routed over the minimum spanning trees since the tree is separated into two parts if link 4-5 is removed from the network. As a result, the *MCSTA*

will rely mostly on a successive shortest-path route which generally routes demand through a minimum-cost path based on incremental link cost. In fact, a new spanning tree should be reassigned. For example, when capacity on link 4-5 can not be assigned, a new spanning tree could be establish using links 5-11 and 10-11 instead of link 4-5.

Overall, the *MCSTA* algorithm has demonstrated good performance in approximating the optimal solution within 6 percent or less error. The key advantage of the *MCSTA* algorithm over a traditional standard optimization technique for the mixed integer programming problem lies in the fact that the *MCSTA* algorithm can give a close approximation within a polynomial computational time. Nonetheless, the weakness of the *MCSTA* algorithm is also based on the fact that spine links selected from a minimum spanning tree, used in the algorithm, is determined on undirected link. Therefore, when demand is asymmetric, spine links on a minimum spanning tree may not be optimum over a group of edge nodes. In practice, the capacity allocation is typically done separately for each directional link. As a result, if one considers improving the performance of the *MCSTA* algorithm by choosing new spine links when capacity over some spine links is depleted, directional characteristics of link should be taken into account.

To this extent, a VPNs design utlilizing sink-tree routing paths aims to aggregating a large demand on a small number of high-capacitated links. The bandwidth savings as well as cost reduction is achieved based on this perspective. However, when demand is concentrated on a few links, some part of the network may have link with almost none capacity available. As a result, network congestion is likely to occur, since the traffic load is not well balance across the network. To alleviate such congestion problem, network management may set aside a certain portion of the network bandwidth to absorb congestion as well as for other management purposes. This could easily be done by setting the utilization factor α_l on each $l \in L$ as

explained previously in Chapter 4. The utilization factor protects links from being overly subscribed and subject to potential congestion. In practice, a smaller value of α_l may be assigned to links connecting to core-routers than ones connecting to edge-routers.

6.5. Complexity of the *MCSTA* Algorithm

The *MCSTA* algorithm operates in three phases. First a set of minimum spanning tree routes is computed. Then, the groups of demands ending at the same egress node for all VPNs will be sorted in a decreasing order based on the total sum of required bandwidth of demand pair within each group. Within each group, a demand pair is also sorted in an increasing order of its required bandwidth. After that, the *MCSTA* algorithm will run iteratively for each demand pair by calculating an incremental link cost on every link and executing a successive shortest-path computation.

On a network of *N* nodes and *L* links, a minimum spanning tree could be computed using Prim or Kruskal's algorithm which has the worst case complexity of $\Theta(N^2)$ and $\Theta(L \cdot log\ L)$ respectively. Prim's algorithm is chosen for this purpose. In the sorting operation, the quicksort algorithm is employed. Quicksort is considered to be the fastest sorting algorithm in practice with an average running time of $\Theta(G \cdot log\ G)$, where G is the number of items to be sorted. However, it can be shown to have a worst case complexity of $\Theta(G^2)$. In *MCSTA* operation, the number of items to be sorted is equal to the number of demand groups for all VPNs. Given that there are V VPNs, each of which has a demand going to all $M \subset N$ edge nodes, there are M demand group per VPN. The total number of demand groups is then equal to $V \cdot M$ groups or $V \cdot N$ groups if all nodes are included. Therefore, the time bound for the sorting operation is $\Theta(V^2 \cdot N^2)$. In the successive shortest-path computation, a classic Dijkstra's shortest path

algorithm is applied over a network with modified link cost. Dijkstra's algorithm operates within $\Theta(N^2)$ time. Since, each demand pair will be sequentially routed in order until all demand are put in place, all iterations will be take the total time of $\Theta(K \cdot V \cdot N^2)$ if there are *K* demand pairs to be routed for each VPN. As a result, the worst case performance of the *MCSTA* algorithm is $\Theta(N^2)+\Theta(V^2 \cdot N^2)+\Theta(K \cdot V \cdot N^2)$ which is $\Theta(V^2 \cdot N^2)$ or $\Theta(K \cdot V \cdot N^2)$. This obviously exhibits the fact that the computational time of the *MCSTA* algorithm increases with the number of VPNs as well as the number of nodes in the network. It may seem that the computational time of the *MCSTA* algorithm is bounded by the sorting operation. However, in practice, the sorting is done over some repetitive values of demand bandwidth and takes rather a small amount of time. For this reason, the complexity of the *MCSTA* algorithm can be said to be bounded by $\Theta(K \cdot V \cdot N^2)$.

6.6. Summary

This chapter presents the *MCSTA* approximation algorithm and its variations to solve the multiple VPNs design problem utilizing sink-tree routing paths. One key attribute of the *MCSTA* algorithm is the ordering of a group demand, i.e. traffic pairs ending at the same egress node, based on its aggregated capacity. Another key criteria is a selection of the route on a minimum spanning tree links for large demand when cost is justified against a minimum-cost route. The *MCSTA* algorithm has proved to be able to closely approximate the optimal solution within a polynomial computational time.

7. VPNs Design Models for Multipoint Connections

Two types of traffic connections may be carried over a VPN including point-to-point connections and multipoint connections. In a traditional point-to-point application, commonly known as unicast connection, sender and receiver have a one-to-one relationship. Many emerging applications are multipoint applications where a single data stream is targeted to a group of receivers at various locations and vice versa. Examples of these applications are video conferencing, broadcast TV, groupware or computer-supported cooperative-work and etc. In multipoint connections, sender and receiver may have many-to-one, one-to-many or many-to-many relationship. More specifically, multipoint connections include broadcast (point-to-multipoint), merge (multipoint-to-point), composite (point-to multipoint and multipoint-to-point) and full multipoint (multipoint-to-multipoint) [69].

Although multipoint connections can be supported using multiple point-to-point connections, the resulting level of network resource utilization can expected to be low. In other words, when point-to-point paths are employed for a multipoint connection, the same information must be replicated for each point-to-point connection and therefore results in poor utilization of network bandwidth. Instead, multipoint connections should be routed over a distribution tree where only a single copy of information will be transmitted over any network link. In general, a distribution tree for multipoint connections utilizing minimum amount of network resources is preferred.

From a network design perspective, point-to-point as well as multipoint connections should be taken into account. Regardless of the point-to-point traffic, a network designer must efficiently map multipoint traffic to a candidate tree routes and wisely allocate bandwidth to each segment of a selected route corresponding to the traffic demand. For example, in a multipoint-to-point connection, if only one source can be active at a time, the bandwidth allocated to a segment from the merge point to the downstream point is the maximum of all source-transmitted rates.

As stated earlier, when the capacity cost is to be minimized, a shortest path algorithm is incapable of given an optimal distribution tree when bandwidth could be shared among group of sources and/or destinations. The shortest path algorithm aims to optimize cost on a pair-wise basis, and bandwidth saving is only a by product when paths converge. Instead, one should try to utilize a tree that exploits link-sharing as much as possible such that packets are only duplicated when paths diverge or converge (in case of point-to-multipoint and multipoint-to-point connections, respectively). Even though some traffic may be sent over a longer path than a shortest path, the total distribution cost of all multipoint connections could be minimized. For any arbitrary network, determining an optimal tree of minimum cost over a subset of nodes, such that a bandwidth constraint is satisfied, is, in fact, an NP-hard problem.

In this chapter, the VPN design model is extended to include multipoint connections. In particular, multipoint-to-point and point-to-multipoint connections are considered. The model, however, assumes that one tree is built for each multipoint-to-point or point-to-multipoint connection of each service class within the same VPN and hour-period respectively. For a large number of VPNs, the Minimum-Capacity Sink-Tree Assignment (*MCSTA*) algorithm will be applied with slight modification to solve the VPNs design problem for multipoint connections.

7.1. VPNs Design Model for Multipoint-to-Point Connections

In a multipoint-to-point connection, data from multiple sources is transferred to a single destination. Each source may send data to the destination node at an arbitrary rate, which is usually independent of those of other sources. Multipoint-to-point demand can be supported using a connection-oriented paradigm where an explicit tree path is being setting up for each multipoint-to-point group. However, by using this paradigm, the complete topology information and multipoint group membership and their bandwidth must be known in advance. It is important to note that the model does not address the dynamic characteristics of the multipoint demand where member may join and leave the network at any point in time.

The problem of multiple VPNs design for multipoint-to-point demand shares the same characteristics as the design for point-to-point demand utilizing sink-tree paths where a destined egress router is at the root of a tree and all ingress routers are at the leaves of a tree. Hence, the VPNs design model with bandwidth aggregation (*VPWBA*) presented in Chapter 4 can be applied for a multipoint demand with a modification in capacity assignment. For example, in a multipoint-to-point connection, if only one source can be active at any time, the bandwidth allocated to any link on a sink-tree path is a maximum of all source-transmitted rates. Given that the sources $d \in D_{v,s,k}$ within demand group $k \in K$ transmitted at the rate $B^{d}_{v,h,s,k}$ and sink-tree path $p \in P_k$ is selected for service class $s \in S$ of VPN $v \in V$ and hour period $h \in H$, the bandwidth assignment $\beta^{l}_{v,h,s,k}$ on link $l \in L$ for multipoint-to-point demand ending at $k \in K$ egress node can be derived as

$$B^{d}_{v,h,s,k} \cdot \sum_{p \in P_k} \gamma^{l}_{p,d} \cdot X^{p}_{v,h,s,k} \quad \leq \quad \beta^{l}_{v,h,s,k} \qquad (7.1)$$

Note that the $\beta^{l}_{v,h,s,k}$ has subscripts of $s \in S$, $v \in V$ and $h \in H$ to reflect the fact that bandwidth assignment is done separately for each service class within a VPN and an hour period, respectively. An equivalent bandwidth allocation is calculated for $\beta^{l}_{v,h,s,k}$ to find an actual link capacity allocation for this multipoint-to-point demand within service class $s \in S$ of VPN $v \in V$ and hour period $h \in H$.

$$EB^{l}_{v,h,s,k} = Eqv\left(\beta^{l}_{v,h,s,k}, T_s, QoS_s\right) \qquad (7.2)$$

The rest of the model remains the same as in *VPBWA*. Accordingly, the VPN design model for multipoint-to-point connection (*VPMTP*) can be stated as the following.

VPMTP : ***Minimize*** $\sum_{l \in L} \psi_l * Y_l$

Subject to :

(1) $\sum_{p \in P_k} X^{p}_{v,h,s,k} = 1$; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$

(2) $B^{d}_{v,h,s,k} \cdot \sum_{p \in P_k} \gamma^{l}_{p,d} \cdot X^{p}_{v,h,s,k} \le \beta^{l}_{v,h,s,k}$
; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$, $d \in D_{v,s,k}$

(2) $EB^{l}_{v,h,s,k} = Eqv\left(\beta^{l}_{v,h,s,k}, T_s, QoS_s\right)$
; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$, $l \in L$

(3) $\sum_{s \in S} \sum_{k \in K} EB^{l}_{v,h,s,k} \le U^{l}_{v,h}$; for all $v \in V$, $h \in H$, $l \in L$

(4) $\sum_{v \in V} U^{l}_{v,h} \le Y_l$; for all $h \in H$, $l \in L$

(5) $Y_l \le \alpha_l \cdot C_l$; for all $l \in L$

(6) $X^{p}_{v,h,s,k} \in \{0,1\}$; for all $v \in V$, $h \in H$, $s \in S$, $k \in K$, $p \in P_k$

(7) $Y_l \ge 0$; for all $l \in L$

The *VPMTP* model assumes that multiple distribution trees are used for all multipoint-to-point demand within one VPN. Particularly, one distribution tree is intended for each multipoint-to-point demand ending at one egress router of the same service class of a VPN. When demand is varied across multi-hour periods, a different distribution tree may be used for each hour period.

7.2. VPNs Design Model for Broadcast Connections

Numerous emerging multimedia applications use broadcast communications to distribute large bandwidth high-quality information to multiple destinations simultaneously. Samples of these applications are video/TV on demand, distance learning systems, newspaper publication and health care, etc. Therefore, broadcast demand soon will become a dominant portion of all traffic. Since broadcast demand has a potential to consume a large amount of bandwidth as well as require a guarantee on a minimum bandwidth allocation such that a degradation will not harm the quality of its transmission. Using multiple point-to-point routes for broadcast demand could be cost inefficient because a large amount of traffic must be replicated. Hence, constructing a broadcast tree spanning all recipients is a far better strategy in practice. Since only a single replication of information is transmitted on any link on a distribution tree, a large bandwidth savings could be realized. Typically, a source will broadcast at a certain rate but each recipient has different capability in receiving the traffic. As a result, a broadcast traffic may be splitting out at different rates at the distribution points.

Here, the *VPWBA* design model is extended for broadcast demand. Two choices of broadcast-traffic layout are possible. First, a set of multiple point-to-multipoint trees (or source-based trees) could be used for all broadcast demand within one VPN. Each point-to-multipoint tree carries broadcast traffic for each participating source within one VPN. Another is to employ

a single distribution tree for all broadcast traffic from all participating sources within one VPN. Again, a distribution tree is intended for each service class and different distribution trees may be used over multi-hour periods.

7.2.1. Multiple Point-to-Multipoint Trees

When multiple point-to-multipoint trees paths (or a source-based tree paths) are used for all broadcast demand, the *VPWBA* design model could, again, be applied with a minor modification. In this case, a group of demand within each VPN must be known in advance. Each source will declare its transmission rate and all recipients intended to receive such information will declare their maximum receiving rates. Different recipients may have different capabilities in receiving broadcast traffic which, for example, may depend on receiver's access link capacity, processing capability, buffer size and so on. Therefore, broadcast traffic will transmit to any recipient at the rate which is a minimum between a source transmission rate and the recipient receiving rate. Let D^v represents a broadcast demand matrix of VPN $v \in V$, which declares a transmission rate from a source (an ingress node) to its corresponding recipient (an egress node). D^v is a $M \; x \; M$ matrix which $d^v_{r,m}$ is a directional transmission rate from an ingress node $r \in R$ ($R \subseteq M$) to egress node $m \in M$ of VPN $v \in V$. Then, D^v_r is a row vector of a broadcast demand (in term of transmission rate) from source $r \in R$ of VPN $v \in V$.

$$D^v_r = [\, d^v_{r,1}, \quad d^v_{r,2}, \quad d^v_{r,3}, \quad d^v_{r,4}, \cdots, d^v_{r,m} \,] \tag{7.3}$$

where $d^v_{r,r} = 0$ for all $r \in R$.

Each broadcast demand D_r^v from each source will together be routed as a single entity on a source-based tree. For broadcast demand, the *VPWBA* design model can be rewritten such that the tree path $p \in P_r$ represents a source-based tree or a tree rooted at a source node $r \in R$ and all recipients are at its leave nodes. From this perspective, a sink-tree path can still be applied, but, to route a demand in its reverse direction emerging from a root node (or a source node). Therefore, one tree path in $p \in P_r$ is selected for each broadcast demand from any source node $r \in R$ and path selection constraint can be written as

$$\sum_{p \in P_r} X^p_{v,h,s,r} = 1 \qquad (7.4)$$

Typically, broadcast traffic flow from a source will be split out to a smaller flow when a tree path diverges. Based on a transmission rate from source to each recipient, the amount of bandwidth allocated on any link over a distribution tree is the maximum of all transmission rates from a source to all of its recipients. Given the source $r \in R$, the broadcast traffic to all intended recipients in $d \in D_{v,s,r}$ at the transmit rate of $B^d_{v,h,s,r}$ for service class $s \in S$ of VPN $v \in V$ and hour period $h \in H$, the bandwidth assignment $\beta^l_{v,h,s,r}$ on link $l \in L$ for broadcast demand from source $r \in R$ can be derived from

$$B^d_{v,h,s,r} \cdot \sum_{p \in P_r} \gamma^l_{p,d} \cdot X^p_{v,h,s,r} \leq \beta^l_{v,h,s,r} \qquad (7.5)$$

Again, the $\beta^l_{v,h,s,r}$ has subscripts of $s \in S$, $v \in V$ and $h \in H$ to reflect the fact that a bandwidth assignment is done separately for each service class within a VPN and an hour period, respectively. Then, an equivalent bandwidth allocation is calculated for $\beta^l_{v,h,s,r}$ to find the actual link capacity allocation for this multipoint-to-point demand within service class $s \in S$ of VPN $v \in V$ and hour period $h \in H$.

$$EB^{l}_{v,h,s,r} \quad = \quad Eqv\left(\beta^{l}_{v,h,s,r}, T_s, QoS_s\right) \qquad (7.6)$$

The rest of the model remains the same as in *VPBWA*. Accordingly, the VPN design model for a broadcast demand over multiple point-to-multipoint broadcast tree paths (*VPMBT*) can be written as the following.

VPMBT : ***Minimize*** $\sum_{l \in L} \psi_l * Y_l$

Subject to :

(1) $\sum_{p \in P_r} X^{p}_{v,h,s,r} = 1$; for all $v \in V$, $h \in H$, $s \in S$, $r \in R$

(2) $B^{d}_{v,h,s,r} \cdot \sum_{p \in P_r} \gamma^{l}_{p,d} \cdot X^{p}_{v,h,s,r} \leq \beta^{l}_{v,h,s,r}$
; for all $v \in V$, $h \in H$, $s \in S$, $r \in R$, $d \in D_{v,s,r}$

(2) $EB^{l}_{v,h,s,r} = Eqv\left(\beta^{l}_{v,h,s,r}, T_s, QoS_s\right)$
; for all $v \in V$, $h \in H$, $s \in S$, $r \in R$, $l \in L$

(3) $\sum_{s \in S} \sum_{r \in R} EB^{l}_{v,h,s,r} \leq U^{l}_{v,h}$; for all $v \in V$, $h \in H$, $l \in L$

(4) $\sum_{v \in V} U^{l}_{v,h} \leq Y_l$; for all $h \in H$, $l \in L$

(5) $Y_l \leq \alpha_l \cdot C_l$; for all $l \in L$

(6) $X^{p}_{v,h,s,r} \in \{0,1\}$; for all $v \in V$, $h \in H$, $s \in S$, $r \in R$, $p \in P_r$

(7) $Y_l \geq 0$; for all $l \in L$

The *VPMBT* model assumes that multiple source-based trees are used for each broadcast demand within one VPN. Specifically, one distribution tree is intended for each broadcast

demand originated from an ingress node within the same service class of a VPN. When demand is varied across a multi-hour period, a different source-based tree may be used for each hour period. One could be aware of a similarity of *VPMTP* and *VPMBT* design models. The major difference lies on the fact that *VPMTP* makes use of a sink-tree ending at an egress node while *VPMBT* makes use of a source-based tree rooted at an ingress node. Hence, the same set of candidate tree path given by a heuristics path selection algorithm described in Chapter 5 could be used in both *VPMTP* and *VPMBT* design model. In spite of this, when a set of candidate tree paths is used in *VPMBT* design model, the same tree path is used to route a traffic in a reverse direction rooting at the source node instead of the destination node.

7.2.2. A Single Broadcast Tree

A broadcast demand from multiple sources within one VPN could be transmitted using a single broadcast tree spanning over all nodes within the group. In here, the cast of the VPN design model for broadcast demand based on a single broadcast tree is denoted *VPSBT*. When a single broadcast tree is being used for all broadcast demand, the *VPMBT* design model could be employed with a modification in the path selection constraint. In particular, the model forces the use of a single tree path for all demand within the same service class and VPN. Therefore, the path selection criteria in the *VPSBT* design model can be written as

$$\sum_{p \in P} X^{p}_{v,h,s} = 1 \qquad (7.7)$$

Note that subscript $r \in R$ is dropped from routing variable $X^{p}_{v,h,s}$ since the same tree is selected for all demand emerging from all source nodes. Therefore, the model yields one distribution tree

for each service class within one VPN and hour period. Different distribution trees may be used over a multi-hour period.

In the VPSBT design, the same path set is used for all demand groups. In fact, a path from a candidate tree set will be derived for all sources to all destinations. For this reason, when a heuristic path selection algorithm computes for a set of candidate tree paths, a hop-count limitation constraint must be imposed on every source-destination pair to ensure that any traffic demand transmitted over a broadcast tree will not violate a maximum delay requirement. The *VPSBT* design problem can be defined as follows.

VPSBT : ***Minimize*** $\sum_{l \in L} \psi_l * Y_l$

Subject to :

(1) $\sum_{p \in P} X^p_{v,h,s} = 1$; for all $v \in V$, $h \in H$, $s \in S$

(2) $B^d_{v,h,s,r} \cdot \sum_{p \in P} \gamma^l_{p,r,d} \cdot X^p_{v,h,s} \leq \beta^l_{v,h,s,r}$
; for all $v \in V$, $h \in H$, $s \in S, r \in R, d \in D_{v,s,r}$

(2) $EB^l_{v,h,s} = Eqv\left(\sum_{r \in R} \beta^l_{v,h,s,r}, T_s, QoS_s\right)$
; for all $v \in V$, $h \in H$, $s \in S$, $l \in L$

(3) $\sum_{s \in S} EB^l_{v,h,s} \leq U^l_{v,h}$; for all $v \in V$, $h \in H$, $l \in L$

(4) $\sum_{v \in V} U^l_{v,h} \leq Y_l$; for all $h \in H$, $l \in L$

(5) $Y_l \leq \alpha_l \cdot C_l$; for all $l \in L$

(6) $X^p_{v,h,s} \in \{0,1\}$; for all $v \in V$, $h \in H$, $s \in S$, $p \in P_r$

(7) $Y_l \geq 0$; for all $l \in L$

7.3. Analysis of VPNs Design for Multipoint Connections

VPNs design models for multipoint connections are employed to simultaneously find an optimum layout as well as capacity assignment of multiple VPNs over a given network infrastructure. The design problems of *VPMTP, VPMBT* and *VPSBT* were implemented using AMPL and solved by he CPLEX 7.1 InP solver implementing branch and bound solution technique. Multipoint demand for different numbers of VPNs is generated at two selected load values of 0.3 and 0.9 to represent a low and high load scenario, respectively. The results of multipoint demand routed over tree paths are compared with when a point-to-point path is used.

Table 7.1 shows the optimum cost of the *VPMTP* design of multipoint-to-point demand routed over multiple sink-tree paths. At different numbers of VPNs and load values, the cost of using sink-tree paths is obviously shown to be much smaller than when point-to-point paths are being used. The percentage of cost savings over point-to-point paths for the 10, 15, and 55-node networks varied between 20 to 30 percent, 30 to 35 percent and 16 to 35 percent respectively. A larger cost savings has been realized especially when a network is highly loaded. For example, when load increases from 0.3 to 0.9, the total cost savings increases from 22.50 to 24.64 percent in case of 1 VPN over the 10-node network, and from 33.94 to 35.07 percent in case of 10 VPNs over the 55-node network. When multipoint-to-point demand is routed over point-to-point paths, a larger cost is presented as network load increases because traffic demand is likely to be replicated at many links possibly introducing congestion at several links. Therefore, some demand must be sent over diverted

Load	Network	#VPN	*VPMTP* Design	Point-to-Point	% Cost Savings
0.3	10	1	572	738	22.50
		100	45,094	64,539	30.13
		200	90,562	130,157	30.42
	15	1	685	1,059	35.31
		100	65,856	95,004	30.68
		200	134,540	195,248	31.09
	55	1	2,355	3,588	34.36
		10	15,631	23,660	33.94
		20	42,077	50,534	16.73
0.9	10	1	614	815	24.64
		100	45,104	65,364	31.00
		200	90,777	130,918	30.66
	15	1	742	1,116	33.50
		100	65,853	95,004	30.68
		200	134,534	195,248	31.10
	55	1	inf	inf	non
		10	15,701	24,180	35.07
		20	42,311	51,496	17.84

Table 7.1 : Multipoint-to-point Demand over Multiple Sink Trees

Load	Network	#VPN	*VPMBT* Design	Point-to-Point	% Cost Savings
0.3	10	1	500	738	32.27
		100	45,164	64,539	30.02
		200	90,497	130,157	30.47
	15	1	645	1,059	39.07
		100	65,573	95,004	30.98
		200	134,292	195,248	31.22
	55	1	2,029	3,588	43.46
		10	15,683	23,660	33.72
		20	42,853	50,534	15.20
0.9	10	1	534	815	34.43
		100	45,272	65,364	30.74
		200	90,777	130,918	30.66
	15	1	706	1,116	36.77
		100	65,576	95,004	30.98
		200	134,299	195,248	31.22
	55	1	inf	inf	non
		10	15,683	24,180	35.14
		20	42,894	51,496	16.70

Table 7.2 : Broadcast Demand over Multiple Point-to-Multipoint Trees

ρ	Network	*VPSBT* Design	*VPMBT* Design		Point-to-Point	
		Cost	Cost	% Cost Saving	Cost	%Cost Saving
0.3	10	382	500	23.62	738	48.27
	15	502	645	22.25	1,059	52.62
	55	1,890	2,029	6.82	3,588	47.31
0.9	10	507	534	5.18	815	37.82
	15	529	706	25.10	1,116	52.64
	55	1,890	inf	non	inf	non

Table 7.3 : Broadcast Demand over a single Multipoint-to-Multipoint Tree

paths and use more resources than the optimum routes. However, in case of the 15-node network, the cost increment is not significant since several alternate paths of equivalent cost are available across all egress nodes. Thus, the percentage of cost saving only slightly increases at an increased load.

For broadcast demand, the *VPMBT* design model is applied. Multiple source-based trees are selected for the broadcast demand of each VPN. As before, the comparison is made between *VPMBT* optimal cost and the cost of routing demand through point-to-point paths as shown in Table 7.2. The *VPMBT* design yields a smaller cost over a point-to-point path and the total cost saving falls between 15 to 43 percent. Similar results can be observed at a larger load value. An increase in the cost savings can be observed except for the 15-node network case where the load has a small effect on the cost savings. For instance, when the load increases from 0.3 to 0.9, the cost savings increases from 32.27 to 34.43 percent in the case of 1 VPN over the 10-node network, and 33.72 to 35.14 percent in the case of 10 VPNs over the 55-node network

Table 7.3 shown the result of the *VPSBT* design when all broadcast demand within a VPN is aggregated over a single broadcast tree. It is clear demonstrated that the *VPSBT* design provides a good gain in bandwidth multiplexing and yields a smaller cost as compared to the *VPMBT* design and when point-to-point paths are employed. Overall, the *VPSBT* design results

in approximately 5 to 25 percent savings in cost compared to the *VPMBT* design. The cost savings is due to the fact that when a single broadcast tree is employed, all broadcast demand from all sources can be multiplexed over a common links and therefore a high degree of bandwidth efficiency could be attained. The cost benefit of *VPSBT* design is more pronounced when compared with point-to-point paths and observed to range between 37 to 52 percent. These results further substantiate the applicability of the *VPSBT* design for broadcast demand over VPN. In contrast, utilizing point-to-point paths for broadcast demands is shown to poorly use network resources and very likely to introduce more congestion within the network.

7.4. *MCSTA* Approximation for Multipoint Connections

When dealing with a large number of VPNs, it may not be practical to solve the VPN design problem using a standard optimization approach especially when multipoint demand is concerned. The large amount of computational resources and time may limit applicability of *VPMTP, VPMBT* and *VPSBT* design model since, in general, the problem of multipoint demand is more complex than that of a point-to-point demand. In this section, the *MCSTA* approximation will be applied to give cost estimation for *VPMTP* and *VPMBT* design model when multiple sink trees or source-based trees are considered. By nature, the *MCSTA* can directly be applied to multipoint demand cases with a minor modification in bandwidth assignment.

For multipoint-to-point demand, let $\beta_{k,l}^{v}$ represent a bandwidth assignment of demand D_k^v over link $l \in L$. For each demand pair $d_{m,k}^{v}$ in D_k^v, an incremental cost $(\nabla \theta_{k,l}^{v})$ of carrying a demand $d_{m,k}^{v}$ is calculated on all links $l \in L$.

$$\nabla \theta_{k,l}^{v} = \{ eqv(max\{ \beta_{k,l}^{v} , d_{m,k}^{v} \}) - eqv(\beta_{k,l}^{v}) \} \cdot \psi_l \qquad (7.8)$$

In other words, an incremental cost at any link for demand $d^v_{m,k}$ is the amount of additional cost when the maximum between current bandwidth assignment $\beta^v_{k,l}$ on link $l \in L$ and the demand $d^v_{m,k}$ will be put on that link. The rest of the *MCSTA* algorithm remains the same.

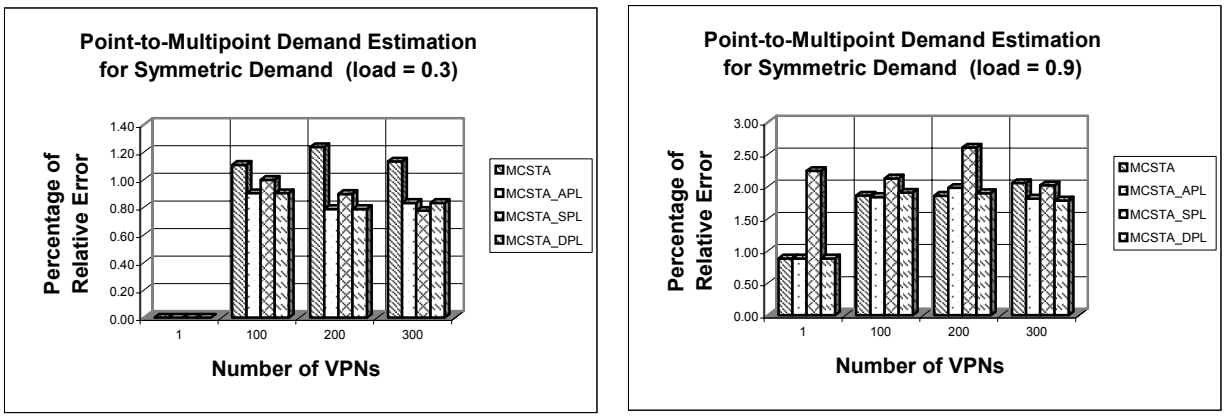
In case of broadcast demand, the demand D^v_r represent a matrix of transmission rate $d^v_{r,m}$ emerging from a source $r \in R$ destined to $m \in M$ and, therefore, an incremental cost $(\nabla \theta^v_{r,l})$ of carrying a demand $d^v_{r,m}$ on link $l \in L$ can be calculated as

$$\nabla \theta^v_{r,l} = \{ eqv(max\{ \beta^v_{r,l} , d^v_{r,m}\}) - eqv(\beta^v_{r,l}) \} \cdot \psi_l \qquad (7.9)$$

where, $\beta^v_{r,l}$ represent a bandwidth assignment of demand D^v_r over link $l \in L$. As before an incremental cost at any link for demand $d^v_{r,m}$ is the amount of additional cost when the maximum between current bandwidth assignment $\beta^v_{r,l}$ on link $l \in L$ and demand $d^v_{r,m}$ will be put on that link. Other parts of the *MCSTA* algorithm can be applied likewise.

The *MCSTA* algorithm and its variations are evaluated numerically for the case of multipoint demand. The goodness of the algorithms was determined by comparing an approximation with the optimum solution given by branch and bound approach. Complete numerical results of *MCSTA* approximation for multipoint demand can be found in Appendix D. First, symmetric demand is considered as shown in Figure 7.1 where the percentage of relative error of various *MCSTA* approximations for multipoint-to-point demand is compared at different numbers of VPNs. At 0.3 load, a relative error is shown to be less than 1.2 percent and, for a single VPNs, the exact solution is given. The relative error increases at high load but is within 2.5 percent approximately. The *MCSTA* approximation yields results inferior to its variations at

light load. However, at a 0.9 load, the *MCSTA* seems to give a better approximation across various number of VPNs.



(a) *(b)*

Figure 7.1 : Multipoint-to-point Demand over Sink Trees on the 10-node network (Symmetric Demand)

Asymmetric demand cases are shown in Figure 7.2. Overall, the *MCSTA* can give a close approximation within 5 percent relative error even at highly loaded cases. The performance of the *MCSTA* has proven to be consistent and predictable in many cases. For a small and dense network as in the 10-node network, the relative error is less than 1.2 percent. On the other hand, *MCST_SPL* approximation yields a larger relative error than the other approximations for the 10 and 15-node networks. For example, the relative error of the *MCSTA_SPL* can be as high as 7.32 and 10.13 percent for the 10 and 15-node networks at 0.9 load. For the 55-node network, the results from *MCSTA_SPL* approximation are comparable to other approximations. Only, for a case of 5 VPNs at 0.9 load, the *MCSTA_SPL* give the best approximation with relative error of less than 1 percent.

The approximation of the *MCSTA* algorithm for broadcast demand is shown in Figure 7.3. In this case, the *MCSTA* algorithm yields a good approximation within 6 percent of relative

error and in several cases gives an exact solution. It is shown that for the 10-node network at 0.3 load, the *MCSTA* algorithm give an inferior approximation among all. Nevertheless, at 0.9 load, it is shown to best approximate the optimum solution. It also demonstrates a good performance in case of the 15-node network at both low and high load situation. For the 55-node network, the *MCSTA* does not exhibit a good performance in comparison with other approximations and the *MCST_SPL* is shown to be the best across all number of VPNs. However, the error is relatively small (< 3%) for the 55-node network.

7.5. Conclusion

In this chapter, the VPN design model is extended for multipoint demand traffic including multipoint-to-point and point-to-multipoint connections. The VPNs design model with bandwidth aggregation (*VPWBA*) presented in Chapter 4 can be applied for multipoint demand with some modification in capacity and route assignment. Specifically, three VPNs design models are constructed including (1) the model for multipoint-to-point connection utilizing sink trees (*VPMTP*), (2) the model for broadcast demand utilizing multiple source-based trees (*VPMBT*), and (3) the model for broadcast demand utilizing a single broadcast tree (*VPSBT*). The total capacity cost derived from *VPMTP*, *VPMBT* and *VPSBT* models is shown to be much smaller than when multipoint demand is routed over point-to-point paths where replication of information causes unnecessary use of network resources. Furthermore, the cost savings is shown to be as high as 50 percent when a single broadcast tree is employed. These results clearly illustrate the advantage of utilizing a tree path for multipoint connection.

With these VPN design models, a multipoint connection is supported within a connection-oriented paradigm where an explicit tree path is being setting up for each multipoint-

to-point group. The use of such models for a multipoint connection requires that the complete topology information and multipoint group membership and their bandwidth must be known in advance. Moreover, the model does not address the dynamic characteristics of the multipoint demand where member may join and leave the network at any point in time.

Lastly, the *MCSTA* algorithm is extended to account for multipoint-to-point and broadcast demand with a minor modification in bandwidth assignment computation. The *MCSTA* algorithm demonstrates a good performance in closely approximating the optimum solution within 6 percent of relative error while having a polynomial computational time.

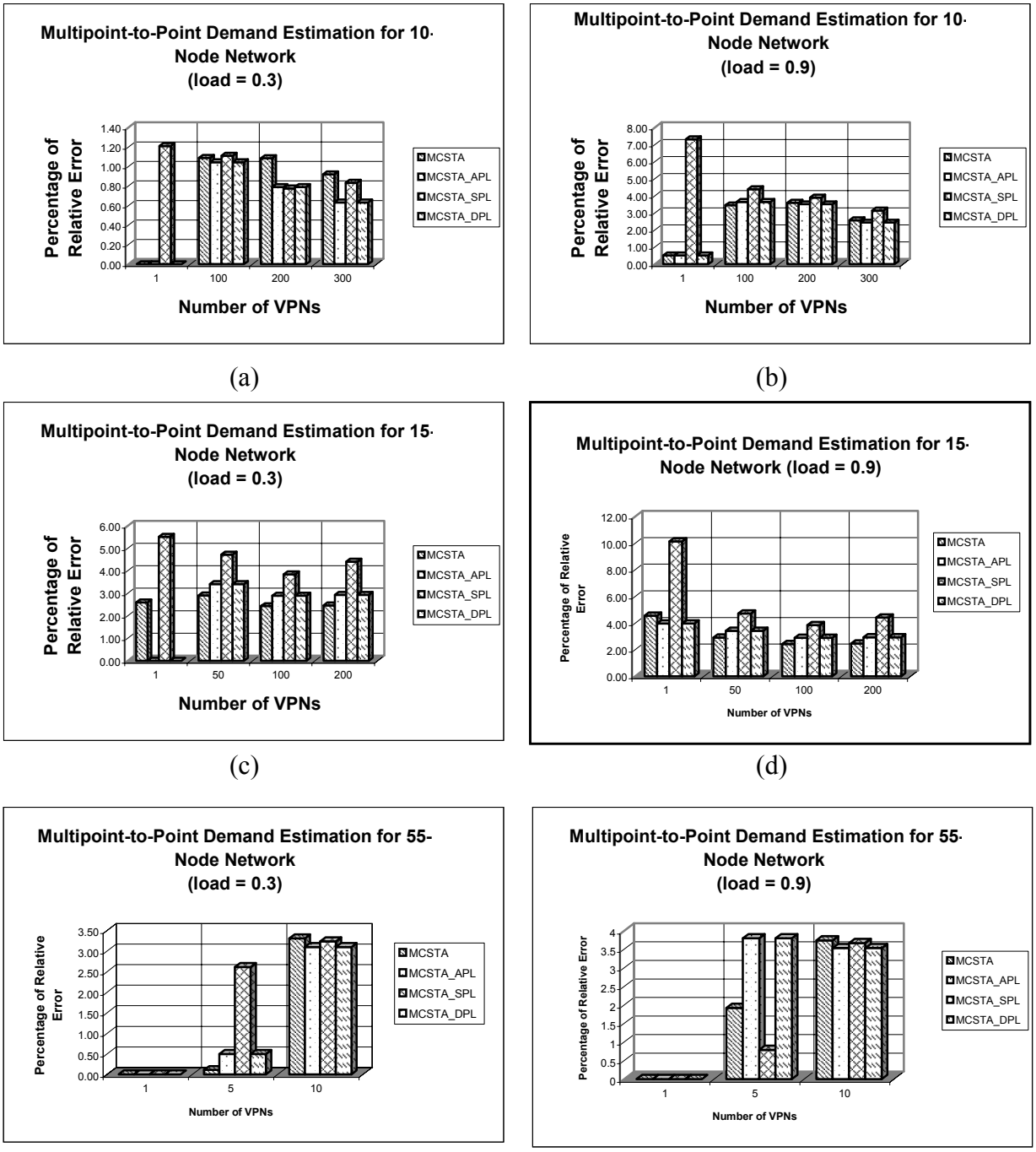


Figure 7.2: *MCSTA* Approximations for Multipoint-to-Point Demand

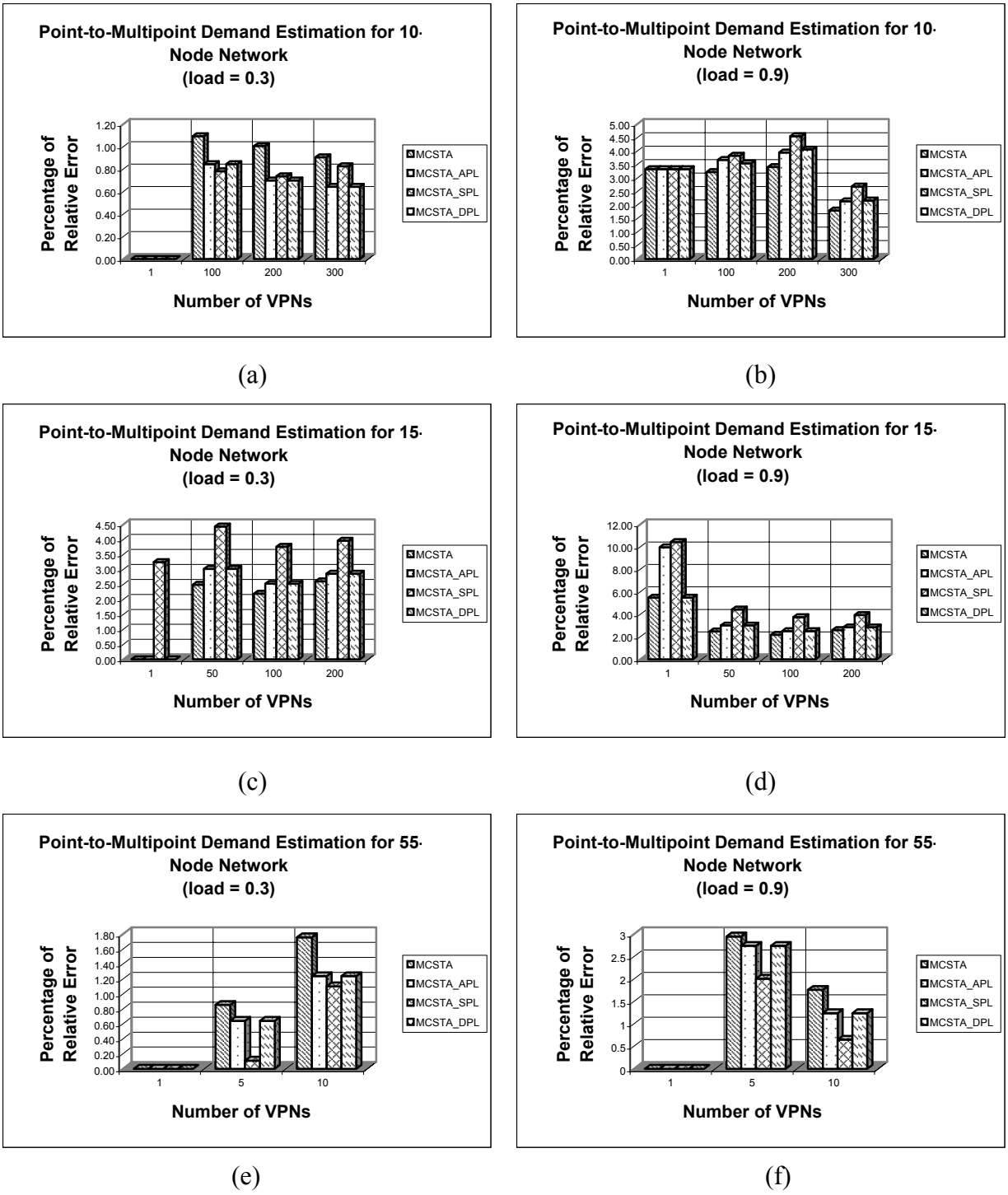


Figure 7.3: MCSTA Approximation for Broadcast Demand

8. Research Summary

In this chapter, the research is summarized on the VPN design model utilizing sink-tree paths. Then research contributions are summarized along with future possible directions.

8.1. Research Summary

This thesis proposes mathematical formulations for the problem of VPN design in order to simultaneously find optimal VPNs logical topologies and their dimensions over a MPLS-like infrastructure network to carry multi-service, multi-hour traffic. The proposed model utilizes pre-computed sink-tree paths (multipoint-to-point paths) over which VPN traffic is routed within a core network. In the model, different levels of bandwidth aggregation/multiplexing occur across different service classes and routes within one VPN, but not across different VPNs. Numerical studies illustrate the fact that, by routing VPNs traffic over multiple logical sink trees, the number of label switch paths can be reduced. Moreover, when bandwidth aggregation is considered in a sink-tree design, demand is routed over a tree that is different from a shortest-path tree such that link capacity can be shared among multiple connections, and, therefore, the total capacity cost is reduced.

A thorough analysis of problem solutions reveals a notable fact that the optimal trees agree on using many common links when possible. The thesis, therefore, presents a tree selection heuristics aiming to scale the VPN design problem by choosing a small-but-good candidate set of feasible sink-tree paths. A proposed heuristic introduces a new selection criteria which limits the number of links used in a tree. The size-limit sink-tree set is chosen by ranking trees based on the

number of links used in the tree, given that a hop-count limit is not violated. Numerical results obtained clearly show the benefit of the tree selection heuristic in reducing the candidate set of sink-tree paths to be searched over and allowing the optimal solution to be obtained within a reasonable time for realistic-size network. Nonetheless, when a large number of VPNs are considered to be laid out simultaneously, a standard optimization approach may not be practical considering that VPN planning and management may be done quite frequently over a certain time period. Thus, a heuristics-based approach, *Minimum-Capacity Sink-Tree Assignment* (*MCSTA*), is devised to approximate the optimal bandwidth and sink-tree route assignment for multiple VPNs. One key attribute of the *MCSTA* algorithm is the ordering of a group demand (traffic pairs ending at the same egress node) based on its aggregated capacity. Another key criteria is to use a route on a minimum spanning tree for a large traffic demand when the cost is justified against a minimum-cost route. In the *MCSTA*, a series of successive shortest-path route computation are done with incremental capacity link cost recomputed for each demand pair. In addition, variations of the *MCSTA* algorithm are considered including the *MCSTA_APL*, *MCSTA_SPL* and *MCSTA_DPL* algorithms which employ a resource utilization index as criteria in ordering of a group demand. The resource utilization index is a measure of the amount of network resources consumed by a traffic demand when it is being admitted to the network. The *MCSTA_APL* uses an average path length while *MCSTA_SPL* uses the shortest path length between a node-pair to calculate the resource utilization index. The group demand which will consume the largest amount of network resources will be routed first. In the *MCSTA_DPL*, the aggregated capacity is used to determine a group ordering but a pair-demand to be routed will be ordered based on the resource utilization index. The *MCSTA* algorithm and its variations were

shown to be able to closely approximate the optimal solution within a polynomial computational time.

Lastly, the VPN design model is extended for multipoint traffic demands including multipoint-to-point and point-to-multipoint connections. Specifically, three VPNs design models are constructed including (1) a design model for multipoint-to-point connection utilizing sink trees (*VPMTP*), (2) a design model for broadcast demand utilizing multiple source-based trees (*VPMBT*), and (3) a design model for broadcast demand utilizing a single broadcast tree (*VPSBT*). Numerical study illustrates clear advantage of utilizing a tree path for multipoint connection in bandwidth savings over a traditional point-to-point path. When a large number of VPNs are concerned, the *MCSTA* algorithm can be slightly modified to include multipoint connections. It can be shown that *MCSTA* algorithm can establish a good approximation with a small relative error while operating on a polynomial time scale.

8.2. Research Contributions

The major contributions of this dissertation are as follows

- The design of multiple VPNs over MPLS-like infrastructure network, using sink-tree routing, is formulated as a mixed integer programming problem to simultaneously find VPNs logical topologies and their dimension to carry multi-service, multi-hour VPNs traffic.

- A heuristic path selection algorithm is proposed to scale the VPN design problem by choosing a small-but-good candidate set of feasible sink-tree paths to solve the

optimization problem over. The proposed heuristic introduces a new selection criteria which limits the number of links used in a tree.

- The minimum-Capacity Sink-Tree Assignment (*MCSTA*) algorithm is devised to approximate the optimal bandwidth and sink-tree route assignment for multiple VPNS in a polynomial computational time.

- Variations of the *MCSTA* algorithm are considered including *MCSTA_APL*, *MCSTA_SPL* and *MCSTA_DPL* algorithms which employ a resource utilization index as criteria in a group demand ordering.

- The VPN design model is extended for multipoint traffic demands including multipoint-to-point and point-to-multipoint connections. Multiple broadcast trees as well as a single broadcast tree are considered for point-to-multipoint connections.

8.3. Future Works

In the VPN design utilizing sink-tree paths, when capacity cost is sought to minimize, traffic demands are concentrated on a few links over tree paths which may introduce over utilization of capacity at a small number of links and could lead to network congestion problem, even though some links may have some capacity available. One solution is to use a link utilization factor to limit the capacity usage at all links especially links between core routers. This, however, may cause a difficulty in selecting a practical value of link utilization to be assigned to every link since a single value should

not be used to all links. Another practical solution is to introduce a load balancing objective to the proposed VPN design model such that capacity cost is minimized while the maximum link utilization are also minimized across an entire network. Multi-objective optimization techniques should be applied.

The *MCSTA* algorithm has been shown to give a close approximation to the VPN design problem. However, at high load scenario, capacity over selected spine links may be exhausted and the algorithm rely mainly on a minimum cost route computation on adjusted incremental link cost which does not well promote the bandwidth aggregation as well as total capacity saving. As a result, the MCSTA algorithm may give a solution farther from the optimum. One way to improve the algorithm, especially for a heavy load scenario, is to reselect new spine links over the topology where links having unavailable capacity are removed. In practice, the link capacity allocation is typically done separately for each directional link and it is very likely that capacity is exhausted only in one direction not both especially when traffic demand is asymmetric. Therefore, directional characteristics of link should be taking into account in a spine link selection.

The proposed VPN design model and the *MCSTA* algorithm may be extended for the case where some existing demand are already in place over existing topology and to find the minimum cost routing plan to carry additional traffic demand. One may consider adding extra capacity on existing and/or adding new capacitated links which the cost of adding acquiring additional capacity as well as the cost of link creation must be added to the design model. This may be appeared as a penalty cost function which allows the use of additional capacity on the link with no left-over capacity but at a higher cost.

Lastly, the proposed VPN design may be extended for WDM infrastructure where a sink-tree path creation is possible. For WDM network, a switching cost at a merging/splitting point and a fixed set-up cost must be accounted for.

Appendix A: Performance of Path Selection Heuristics

I. Effect of Increasing the Number of Tree Paths

No. Tree Paths	Optimal Cost	CPU Time (sec)
50	2,553	0.13
100	2,520	0.26
150	2,520	0.26
180	2,504	0.33
200	2,504	0.38
352	2,504	0.94

Table A.1 : Symmetric Demand over the 10-node network (1 VPN)

No. Tree Paths	Optimal Cost	CPU Time (sec)
50	7,965	130
100	7,478	120
150	7,470	290
180	7,384	300
200	7,384	560
250	7,354	1000
300	7,354	1200
352	7,354	2300

Table A.2 : Symmetric Demand over the 10-node network (4 VPNs)

No. Tree Paths	Optimal Cost	CPU Time (sec)
50	7,668	1.40
100	7,346	1.80
500	7,073	5.80
1000	6,995	5.80
1500	6,958	7.30
3000	6,932	9.40
4000	6,932	12.00
5000	6,932	15.00

Table A.3 : Symmetric Demand over the 15-Node Network (Hop Count Factor = 5)

No. Tree Paths	Optimal Cost	CPU Time (sec)
100	7,203	2.20
500	6,987	4.30
1000	6,932	7.60
1200	6,932	7.50
1500	6,932	10.00
1727	6,932	14.00

Table A.4 : Symmetric Demand over the 15-Node Network (Hop Count Factor = 2)

No. Tree Paths	Optimal Cost	CPU Time (sec)
50	32,465	4,500
100	31,570	2,400
200	31,194	5,000
400	31,017	3,300
600	30,972	4,000
800	30,972	4,900
1000	30,972	5,500
1300	30,936	17,000
1500	30,936	38,000

Table A.5 : Symmetric Demand over the 55-Node Network

No. Tree Paths	Optimal Cost	CPU Time (sec)
50	26,528	3,500
200	25,499	3,500
300	25,351	3,100
400	25,008	6,900
700	25,008	7,900
1100	24,975	19,000
1200	24,975	23,000
1300	24,975	37,000

Table A.6 : Asymmetric Demand over the 55-Node Network

II. Effect of Increasing the Number of VPNs

	# VPNs	1	5	10	50	100	150	200	250	300
Optimum Cost	Set-1	632	3,357	6,624	33,354	65,618	100,714	133,679	165,917	197,045
	Set-2	622	3,265	6,620	33,808	67,423	99,354	132,544	165,441	200,588
	Set-3	693	3,435	6,792	33,505	65,511	100,516	133,490	165,445	199,964
CPU Time (sec)	Set-1	0.33	3.3	7.5	110	380	1100	2400	4200	6900
	Set-2	0.26	2.1	5.7	130	430	1200	2200	3900	8500
	Set-3	0.39	3.1	5.9	84	390	1600	2300	5200	6800

Table A.7 : Symmetric Demand over the 10-node network at 0.3 Load

Network	# VPNs	1	5	10	50	100	150	200	250	300
10	ρ=0.3	632	3,357	6,624	33,354	65,618	100,714	133,679	165,917	197,045
	ρ=0.6	657	3,357	6,624	33,354	65,618	100,714	133,679	165,917	197,045
	ρ=0.9	705	3,476	6,770	34,005	66,699	102,420	136,147	168,614	non
15	ρ=0.3	1,049	4,898	9,922	50,915	99,110	150,365	199,305	non	non
	ρ=0.6	1,049	4,898	9,922	50,915	99,110	150,365	199,305	non	non
	ρ=0.9	1,051	4,898	9,922	50,915	99,110	150,365	non	non	non
	ρ=1.2	1,053	4,898	9,931	50,915	99,110	150,367	non	non	non

Table A.8 : Optimal Cost for Symmetric Demand over the 10 and 15-Node Networks at Various Load

Network	# VPNs	1	5	10	50	100	150	200	250	300
10	ρ=0.3	0.33	3.3	7.5	110	380	1100	2400	4200	6900
	ρ=0.6	0.52	3.3	8.5	110	410	1100	2500	4700	8000
	ρ=0.9	0.64	120	72	560	1800	2100	4600	9300	non
15	ρ=0.3	2.1	18	40	530	4200	12000	23000	non	non
	ρ=0.6	2.6	17	84	980	4400	11000	24000	non	non
	ρ=0.9	2.4	19	40	570	2500	11000	non	non	non
	ρ=1.2	2.3	16	3400	2100	5600	17000	non	non	non

Table A.9 : CPU Time (seconds) for Symmetric Demand over the 10 and 15-Node Networks at Various Load

	# VPNs	1	5	10	20	30
Optimum Cost	ρ=0.3	2,846	11,518	25,439	53,869	79,214
	ρ=0.6	2,909	11,518	25,439	53,869	79,214
	ρ=0.8	inf	11,546	25,441	53,871	79,214
	ρ=0.9	inf	11,711	25,568	53,962	79,362
	ρ=0.97	inf	11,897	25,760	54,057	non
CPU Time (sec)	ρ=0.3	1.5	11	26	210	1900
	ρ=0.6	2	42	24	210	1200
	ρ=0.8	inf	51	50	240	1100
	ρ=0.9	inf	3600	2200	4700	5000
	ρ=0.97	inf	12000	3600	9800	non

Table A.10 : Symmetric Demand over the 55-Node Network at Various Load

	# VPNs	1	50	100	150	200	250	300
Optimum Cost	ρ=0.3	1,035	56,961	113,727	171,339	227,185	283,888	338,706
	ρ=0.9	1,096	57,901	115,911	174,814	231,839	289,133	non
CPU Time (sec)	ρ=0.3	0.61	520	2300	6100	14000	25000	45000
	ρ=0.9	2.4	3600	31000+	9700	31000+	35000	non

Table A.11 : Asymmetric Demand over the 10-node network at Various Load

	# VPNs	1	50	100	150	200
Optimum Cost	ρ=0.3	1,787	83,979	168,036	128,833	non
	ρ=0.9	1,794	83,979	168,036	128,832	non
CPU Time (sec)	ρ=0.3	1	50	100	150	200
	ρ=0.9	9.4	2000	11000	8700	non

Table A.12 : Asymmetric Demand over the 15-Node Network at Various Load

	# VPNs	1	5	10	20	30
Optimum Cost	ρ=0.3	2,980	10,961	20,139	43,397	71,663
	ρ=0.9	inf	11,113	20,183	43,605	71,858
CPU Time (sec)	ρ=0.3	1.1	20	140	670	12000
	ρ=0.9	inf	280	2300	3700	57000

Table A.13 : Asymmetric Demand over the 55-Node Network at Various Load

Appendix B: Performance MCST Algorithms

I. Symmetric Demand Scenario

ρ = 0.3			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	632	632	0.00
5	3,357	3,357	0.00
10	6,624	6,632	0.11
50	33,354	33,382	0.08
100	65,618	65,708	0.14
150	100,714	100,839	0.12
200	133,679	133,869	0.14
250	165,917	166,153	0.14
300	197,045	197,299	0.13

ρ = 0.3		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	0.33	0.26
5	3.30	0.62
10	7.50	1.31
50	110.00	7.10
100	380.00	14.48
150	1100.00	22.03
200	2400.00	29.28
250	4200.00	36.00
300	6900.00	43.29

Table B.1 : Symmetric Demand over the 10-node Network at 0.3 load

ρ = 0.6			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	657	661	0.59
5	3,357	3,357	0.00
10	6,624	6,639	0.23
50	33,354	33,382	0.08
100	65,618	65,708	0.14
150	100,714	100,839	0.12
200	133,679	133,869	0.14
250	165,917	166,153	0.14
300	197,045	197,299	0.13

ρ = 0.6		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	0.52	0.14
5	3.3	0.64
10	8.5	1.50
50	110	8.09
100	410	15.43
150	1100	23.00
200	2500	31.14
250	4700	38.54
300	8000	45.65

Table B.2 : Symmetric Demand over the 10-node Network at 0.6 load

ρ = 0.9			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	705	non	non
5	3,476	3,665	5.45
10	6,770	7,156	5.70
50	34,005	35,676	4.92
100	66,699	69,841	4.71
150	102,420	107,474	4.93
200	136,147	141,733	4.10
250	168,614	176,279	4.55
300	non	211,284	-

ρ = 0.9		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	0.64	non
5	72	0.85
10	120	1.92
50	560	10.21
100	1800	19.39
150	2100	30.45
200	4600	39.78
250	9300	50.12
300	non	64.29

Table B.3 : Symmetric Demand over the 10-node Network at 0.9 load

ρ = 0.6			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	1,049	1,076	2.57
5	4,898	4,932	0.69
10	9,922	10,113	1.92
50	50,915	51,669	1.48
100	99,110	100,425	1.33
150	150,365	152,346	1.32
200	199,305	201,630	1.17

ρ = 0.6		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	2.6	0.12
5	17	0.84
10	84	1.67
50	980	9.20
100	4400	18.10
150	11000	25.53
200	24000	36.14

Table B.4 : Symmetric Demand over the 15-node Network at 0.6 load

ρ = 0.9			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	1,051	1,078	2.52
5	4,898	4,932	0.69
10	9,922	10,113	1.92
50	50,915	51,669	1.48
100	99,110	100,425	1.33
150	150,365	152,346	1.32
200	non	201,630	-

ρ = 0.9		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	2.4	0.14
5	19	0.76
10	40	1.65
50	570	8.89
100	2500	18.32
150	11000	26.76
200	non	36.85

Table B.5 : Symmetric Demand over the 15-node Network at 0.9 load

ρ = 1.2			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	1,053	1,053	0.00
5	4,898	4,933	0.73
10	9,931	10,260	3.32
50	50,915	51,900	1.93
100	99,110	101,444	2.35
150	150,367	153,748	2.25
200	non	203,248	-

ρ = 1.2		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	2.3	0.15
5	16	0.84
10	3400	1.92
50	2100	9.40
100	5600	18.87
150	17000	29.45
200	non	40.39

Table B.6 : Symmetric Demand over the 15-node Network at 1.2 load

ρ = 0.6			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	2,909	2,909	0.00
5	11,518	11,506	0.10
10	25,439	25,431	0.03
20	53,869	54,153	0.53
30	79,214	79,070	0.18

ρ = 0.6		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	2	0.14
5	42	0.76
10	24	1.62
20	210	3.76
30	1200	5.14

Table B.7 : Symmetric Demand over the 55-node Network at 0.6 load

ρ = 0.8			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	non	3,285	-
5	11,546	11,636	0.78
10	25,441	25,431	0.04
20	53,871	54,153	0.52
30	79,214	79,070	0.18

ρ = 0.8		
# VPNs	ILP CPU Time (sec)	MCSTA CPU Time (sec)
1	non	0.17
5	51	0.79
10	50	1.60
20	240	3.75
30	1100	5.26

Table B.8 : Symmetric Demand over the 55-node Network at 0.8 load

ρ = 0.9			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	non	3,347	-
5	11,711	infeasible	-
10	25,568	26,120	2.16
20	53,962	54,532	1.06
30	79,362	80,010	0.82

ρ = 0.9		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	non	0.20
5	3600	non
10	2200	1.70
20	4700	3.87
30	5000	5.53

Table B.9 : Symmetric Demand over the 55-node Network at 0.9 load

II. Asymmetric Demand Scenario

ρ = 0.3			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	1,035	1,035	0.00
50	56,961	57,007	0.08
100	113,727	113,801	0.06
150	171,339	171,461	0.07
200	227,185	227,439	0.11
250	283,888	284,201	0.11
300	338,706	339,040	0.10

ρ = 0.3		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	0.61	0.18
50	520	14.01
100	2,300	27.50
150	6,100	41.01
200	14,000	55.71
250	25,000	66.15
300	45,000	80.39

Table B.10 :Asymmetric Demand over the 10-node Network at 0.3 load

ρ = 0.9			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	1,096	1,167	6.46
50	57,901	60,011	3.64
100	115,911	120,338	3.82
150	174,814	181,429	3.78
200	231,839	240,214	3.61
250	289,133	298,887	3.37
300	non	357,190	non

ρ = 0.9		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	2.40	0.26
50	3,600	16.65
100	31,000	34.51
150	9,700	49.82
200	31,000+	65.67
250	45,000	80.98
300	non	96.64

Table B.11 : Asymmetric Demand over the 10-node Network at 0.9 load

ρ = 0.3			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	1,787	1,817	1.71
50	83,979	85,070	1.30
100	168,036	169,904	1.11
150	128,833	130,268	1.11
200	non	340,234	non

ρ = 0.3		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	9.40	0.23
50	2,000	15.40
100	11,000	29.96
150	8,700	21.53
200	non	58.14

Table B.12 : Asymmetric Demand over the 15-node Network at 0.3 load

ρ = 0.9			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	1,794	1,830	1.97
50	83,979	85,070	1.30
100	168,036	169,904	1.11
150	128,832	130,268	1.11
200	non	340,234	non

ρ = 0.9		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	11.00	0.29
50	2,100	15.81
100	20,000	32.01
150	8,700	23.50
200	non	63.03

Table B.13 : Asymmetric Demand over the 15-node Network at 0.9 load

ρ = 0.3			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	2,980	2,980	0.00
5	10,961	10,971	0.09
10	20,139	19,915	1.11
20	43,397	43,213	0.42
30	71,663	71,820	0.22

ρ = 0.3		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	1.10	0.15
5	20	0.70
10	140	1.37
20	670	2.73
30	12000	4.62

Table B.14 : Asymmetric Demand over the 55-node Network at 0.3 load

ρ = 0.9			
# VPNs	Optimal Cost	MCSTA Cost	%Error
1	inf	inf	-
5	11,113	11,351	2.14
10	20,183	20,365	0.90
20	43,605	44,295	1.58
30	71,858	72,097	0.33

ρ = 0.9		
# VPNs	ILP CPU Time(sec)	MCSTA CPU Time (sec)
1	-	0.17
5	280	0.75
10	2300	1.42
20	3700	3.00
30	57000	4.43

Table B.15 : Asymmetric Demand over the 55-node Network at 0.9 load

Appendix C: Performance Comparison of MCST Algorithms and Its Variations

I. Symmetric Demand Scenario

ρ = 0.3									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	632	632	0.00	632	0.00	632	0.00	632	0.00
50	33,354	33,382	0.08	33,400	0.14	33,415	0.18	33,400	0.14
100	65,618	65,708	0.14	65,738	0.18	65,755	0.21	65,738	0.18
150	100,714	100,839	0.12	100,878	0.16	100,912	0.20	100,878	0.16
200	133,679	133,869	0.14	133,866	0.14	133,943	0.20	133,866	0.14
250	165,917	166,153	0.14	166,169	0.15	166,230	0.19	166,169	0.15
300	197,045	197,299	0.13	197,313	0.14	197,377	0.17	197,313	0.14

Table C.1 : Symmetric Demand over the 10-node Network at 0.3 Load

ρ = 0.9									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	705	inf	-	Inf	-	inf	-	inf	-
50	34,005	35,676	4.92	36,041	5.99	36,667	7.83	35,807	5.30
100	66,699	69,841	4.71	70,333	5.45	71,941	7.86	70,207	5.26
150	102,420	107,474	4.93	107,883	5.33	109,842	7.25	107,861	5.31
200	136,147	141,733	4.10	142,707	4.82	145,736	7.04	142,038	4.33
250	168,614	176,279	4.55	177,684	5.38	181,152	7.44	176,870	4.90
300	non	211,284	-	212,391	-	216,333	-	211,714	-

Table C.2 : Symmetric Demand over the 10-node Network at 0.9 Load

ρ = 0.6									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	1,049	1,076	2.57	1,076	2.57	1,092	4.10	1,076	2.57
50	50,915	51,669	1.48	51,725	1.59	51,985	2.10	51,725	1.59
100	99,110	100,425	1.33	100,545	1.45	101,035	1.94	100,545	1.45
150	150,365	152,346	1.32	152,436	1.38	153,422	2.03	152,436	1.38
200	199,305	201,630	1.17	201,806	1.25	202,690	1.70	201,806	1.25

Table C.3 : Symmetric Demand over the 15-node Network at 0.6 Load

ρ = 1.2									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	1,053	1,053	0.00	1,053	0.00	1,239	17.65	1,053	0.00
50	50,915	51,900	1.93	51,936	2.00	52,537	3.18	51,929	1.99
100	99,110	101,444	2.35	101,334	2.24	102,018	2.93	101,347	2.26
150	150,367	153,748	2.25	153,799	2.28	154,693	2.88	153,818	2.30
200	non	203,248	-	203,626	-	204,620	-	203,395	-

Table C.4 : Symmetric Demand over the 15-node Network at 1.2 Load

ρ = 0.6									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	2,909	2,909	0.00	2,909	0.00	2,927	0.65	2,909	0.00
5	11,518	11,506	0.10	11,515	0.02	11,478	0.35	11,515	0.02
10	25,439	25,431	0.03	25,494	0.21	25,603	0.65	25,494	0.21
20	53,869	54,153	0.53	54,184	0.59	54,287	0.78	54,184	0.59
30	79,214	79,070	0.18	78,986	0.29	79,220	0.01	78,986	0.29

Table C.5 : Symmetric Demand over the 55-node Network at 0.6 Load

ρ = 0.9									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	inf	inf	-	inf	-	inf	-	inf	-
5	11,711	inf	-	inf	-	inf	-	inf	-
10	25,568	26,120	2.16	25,988	1.64	26,186	2.42	25,981	1.62
20	53,962	54,532	1.06	54,640	1.26	54,628	1.23	54,590	1.16
30	79,362	80,010	0.82	79,879	0.65	79,982	0.78	79,927	0.71

Table C.6 : Symmetric Demand over the 55-node Network at 0.9 Load

II. Asymmetric Demand Scenario

ρ = 0.3									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	1,035	1,035	0.00	1,035	0.00	1,035	0.00	1,035	0.00
50	56,961	57,007	0.08	56,993	0.06	56,991	0.05	56,993	0.06
100	113,727	113,801	0.06	113,806	0.07	113,839	0.10	113,806	0.07
150	171,339	171,461	0.07	171,456	0.07	171,515	0.10	171,456	0.07
200	227,185	227,439	0.11	227,416	0.10	227,522	0.15	227,416	0.10
250	283,888	284,201	0.11	284,144	0.09	284,235	0.12	284,144	0.09
300	338,706	339,040	0.10	339,042	0.10	339,181	0.14	339,042	0.10

Table C.7 : Asymmetric Demand over the 10-node Network at 0.3 Load

ρ = 0.9									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	1,096	1,167	6.46	1,144	4.33	1,167	6.43	1,167	6.46
50	57,901	60,011	3.64	60,229	4.02	60,796	5.00	60,157	3.90
100	115,911	120,338	3.82	120,569	4.02	122,666	5.83	120,355	3.83
150	174,814	181,429	3.78	182,484	4.39	184,939	5.79	181,951	4.08
200	231,839	240,214	3.61	241,691	4.25	245,197	5.76	240,736	3.84
250	289,133	298,887	3.37	300,398	3.90	304,973	5.48	299,652	3.64
300	non	357,190	-	359,659	-	364,155	-	358,354	-

Table C.8 : Asymmetric Demand over the 10-node Network at 0.9 Load

ρ = 0.3									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	1,787	1,817	1.71	1,817	1.71	1,818	1.78	1,817	1.71
50	83,979	85,070	1.30	85,181	1.43	85,904	2.29	85,181	1.43
100	168,036	169,904	1.11	170,267	1.33	171,790	2.23	170,267	1.33
150	128,833	130,268	1.11	130,363	1.19	131,181	1.82	130,363	1.19
200	non	340,234	-	340,510	-	342,768	-	340,510	-

Table C.9 : Asymmetric Demand over the 15-node Network at 0.3 Load

ρ = 0.9									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	1,794	1,830	1.97	1,830	1.97	1,869	4.14	1,830	1.97
50	83,979	85,070	1.30	85,181	1.43	85,904	2.29	85,181	1.43
100	168,036	169,904	1.11	170,267	1.33	171,790	2.23	170,267	1.33
150	128,832	130,268	1.11	130,363	1.19	131,181	1.82	130,363	1.19
200	non	340,234	-	340,510	-	342,768	-	340,510	-

Table C.10 : Asymmetric Demand over the 15-node Network at 0.9 Load

ρ = 0.3									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	2,980	2,980	0.00	2,980	0.00	2,980	0.00	2,980	0.00
5	10,961	10,971	0.09	10,920	0.37	11,029	0.63	10,920	0.37
10	20,139	19,915	1.11	19,877	1.30	19,877	1.30	19,877	1.30
20	43,397	43,213	0.42	43,223	0.40	43,372	0.06	43,223	0.40
30	71,663	71,820	0.22	71,877	0.30	72,030	0.51	71,877	0.30

Table C.11 : Asymmetric Demand over the 55-node Network at 0.3 Load

ρ = 0.9									
# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	inf	inf	-	inf	-	inf	-	inf	-
50	11,113	11,351	2.14	11,369	2.30	11,423	2.78	11,369	2.30
100	20,183	20,365	0.90	20,340	0.78	20,380	0.98	20,340	0.78
150	43,605	44,295	1.58	44,448	1.93	44,646	2.39	44,378	1.77
200	71,858	72,097	0.33	72,080	0.31	72,276	0.58	72,153	0.41

Table C.12 : Asymmetric Demand over the 55-node Network at 0.9 Load

Appendix D : VPN Design for Multipoint Connections

I. Multipoint-to-Point Demand

# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	483	483	0.00	483	0.00	483	0.00	483	0.00
50	25,635	25,938	1.18	25,823	0.73	25,889	0.99	25,823	0.73
100	50,440	50,999	1.11	50,896	0.90	50,944	1.00	50,896	0.90
150	77,570	78,483	1.18	78,163	0.76	78,271	0.90	78,163	0.76
200	103,171	104,447	1.24	103,984	0.79	104,097	0.90	103,984	0.79
250	128,074	129,488	1.10	129,020	0.74	129,130	0.82	129,020	0.74
300	151,908	153,630	1.13	153,173	0.83	153,085	0.77	153,173	0.83

Table D.1 : Multipoint-to-point Demand over Sink Trees on the 10-Node Network at 0.3 Load (Symmetric Demand)

# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
1	584	590	0.88	590	0.88	598	2.24	590	0.88
50	25,683	26,252	2.22	26,275	2.31	26,353	2.61	26,277	2.31
100	50,437	51,376	1.86	51,365	1.84	51,510	2.13	51,398	1.90
150	77,580	79,137	2.01	79,008	1.84	79,190	2.08	79,140	2.01
200	103,371	105,291	1.86	105,417	1.98	106,066	2.61	105,332	1.90
250	128,129	130,924	2.18	130,660	1.98	131,217	2.41	130,771	2.06
300	152,079	155,202	2.05	154,835	1.81	155,151	2.02	154,792	1.78

Table D.2 : Multipoint-to-point Demand over Sink Trees on the 10-Node Network at 0.9 Load (Symmetric Demand)

ρ	# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
0.3	1	572	572	0.00	572	0.00	579	1.21	572	0.00
	100	45,094	45,585	1.09	45,565	1.04	45,595	1.11	45,565	1.04
	200	90,562	91,546	1.09	91,279	0.79	91,266	0.78	91,279	0.79
	300	134,736	135,975	0.92	135,588	0.63	135,861	0.83	135,588	0.63
0.9	1	614	617	0.52	614	0.00	659	7.32	617	0.52
	100	45,104	46,659	3.45	46,910	4.00	47,085	4.39	46,748	3.64
	200	90,777	94,036	3.59	93,752	3.28	94,318	3.90	93,963	3.51
	300	134,835	138,306	2.57	138,290	2.56	139,063	3.14	138,120	2.44

Table D.3 : Multipoint-to-point Demand over Sink Trees on the 10-Node Network (Asymmetric Demand)

ρ	# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
0.3	1	685	703	2.60	685	0.00	723	5.51	685	0.00
	50	33,589	34,567	2.91	34,738	3.42	35,175	4.72	34,738	3.42
	100	65,856	67,460	2.44	67,764	2.90	68,390	3.85	67,764	2.90
	200	134,540	137,855	2.46	138,485	2.93	140,475	4.41	138,485	2.93
0.9	1	742	776	4.56	772	3.97	817	10.13	772	3.97
	50	33,589	34,567	2.91	34,738	3.42	35,175	4.72	34,738	3.42
	100	65,853	67,460	2.44	67,764	2.90	68,390	3.85	67,764	2.90
	200	134,534	137,855	2.47	138,485	2.94	140,475	4.42	138,485	2.94

Table D.4 : Multipoint-to-point Demand over Sink Trees on the 15-Node Network (Asymmetric Demand)

ρ	# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
0.3	1	2,355	2,355	0.00	2,355	0.00	2,355	0.00	2,355	0.00
	5	8,333	8,324	0.11	8,291	0.50	8,550	2.62	8,291	0.50
	10	15,631	15,114	3.31	15,146	3.10	15,125	3.24	15,146	3.10
0.9	1	inf	inf	non	inf	non		non	inf	non
	5	8,618	8,453	1.92	8,291	3.80	8,550	0.78	8,291	3.80
	10	15,701	15,114	3.74	15,146	3.53	15,125	3.67	15,146	3.53

Table D.5 : Multipoint-to-point Demand over Sink Trees on the 55-Node Network (Asymmetric Demand)

II. Point-to-Multipoint (Broadcast) Demand

ρ	# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
0.3	1	500	500	0.00	500	0.00	500	0.00	500	0.00
	100	45,164	45,658	1.09	45,545	0.84	45,516	0.78	45,545	0.84
	200	90,497	91,407	1.01	91,130	0.70	91,164	0.74	91,130	0.70
	300	135,672	136,900	0.91	136,544	0.64	136,791	0.83	136,544	0.64
0.9	1	534	552	3.33	552	3.33	552	3.33	552	3.33
	100	45,272	46,735	3.23	46,938	3.68	47,008	3.83	46,880	3.55
	200	90,777	93,876	3.41	94,361	3.95	94,910	4.55	94,466	4.06
	300	136,472	138,933	1.80	139,403	2.15	140,148	2.69	139,433	2.17

Table D.6 : Broadcast Demand over Multiple Point-to-Multipoint Trees on the 10-Node Network (Asymmetric Demand)

ρ	# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
0.3	1	645	645	0.00	645	0.00	666	3.25	645	0.00
	50	33,101	33,925	2.49	34,101	3.02	34,569	4.43	34,101	3.02
	100	65,573	67,013	2.20	67,227	2.52	68,036	3.76	67,227	2.52
	200	134,292	137,792	2.61	138,136	2.86	139,609	3.96	138,136	2.86
0.9	1	706	745	5.50	776	9.99	779	10.43	745	5.50
	50	33,101	33,925	2.49	34,101	3.02	34,569	4.43	34,101	3.02
	100	65,576	67,013	2.19	67,227	2.52	68,036	3.75	67,227	2.52
	200	134,299	137,792	2.60	138,136	2.86	139,609	3.95	138,136	2.86

Table D.7 : Broadcast Demand over Multiple Point-to-Multipoint Trees on the 15-Node Network (Asymmetric Demand)

ρ	# VPNs	Optimal Cost	MCST	% Error	MCST APL	% Error	MCST SPL	% Error	MCST DPL	% Error
0.3	1	2,029	2,029	0.00	2,029	0.00	2,029	0.00	2,029	0.00
	5	8,351	8,279	0.85	8,297	0.64	8,360	0.11	8,297	0.64
	10	15,683	15,407	1.76	15,489	1.24	15,510	1.10	15,489	1.24
0.9	1	inf	inf	non	inf	non	inf	non	inf	non
	5	8,531	8,279	2.95	8,297	2.74	8,360	2.01	8,297	2.74
	10	15,683	15,407	1.76	15,489	1.24	15,581	0.65	15,489	1.24

Table D.8 : Broadcast Demand over Multiple Point-to-Multipoint Trees on the 55-Node Network (Asymmetric Demand)

BIBLIOGRAPHY

[1] R. G. Addie, J. L. Burgin, and S. L. Sutherland, "B-ISDN Protocol Architecture," IEEE GLOBECOM'88, pp. 22.6.1-22.6.5, 1988.

[2] S. Ahn, R. P. Tsang, S. Tong, and D. H. Du, "Virtual Path Layout Design on ATM Networks," IEEE INFOCOM'94, pp. 192-200, 1994.

[3] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, "LDP Specification," Internet Draft, draft-ietf-mpls-ldp-06.txt, October, 1999.

[4] N. G. Aneroussis and A. A. Lazar, "Virtual Path Control for ATM Networks with Call Level Quality of Service Guarantees," IEEE INFOCOM'96, pp. 312-319, 1996.

[5] G. R. Ash, R. H. Cardwell, and R. P. Murray, "Design and Optimization of Networks with Dynamic Routing," *Bell System Technical Journal*, vol. 60, pp. 1787-1820, 1981.

[6] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," RFC 2702, September, 1999.

[7] D. Awduche and Y. Rekhter, "Multiprotocol lambda switching: combining MPLS traffic engineering control with optical crossconnects," *IEEE Communications Magazine*, vol. 39, no. 3, pp. 111-116, March, 2001.

[8] D. O. Awduche, L. Berger, D. Gan, T. Li, G. Swallow, and V. Srinivasan, "Extensions to RSVP for LSP Tunnels," Internet Draft, draft-ietf-mpls-rsvp-lsp-tunnel-04.txt, September, 1999.

[9] A. Banerjee, J. P. Drake, B. Turner, K. Kompella, and Y. Rekhter, "Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements," *IEEE Communications Magazine*, vol. 39, no. 1, pp. 144-150, January, 2001.

[10] T. Bauschert, "Multihour Design of Multi-Hop Virtual Path Based Wide-Area ATM Networks," *15th International Telecommunications Conference*, 1997.

[11] H. C. Berkowitz, "Requirements Taxonomy for Virtual Private Networks," Internet Draft, draft-berkowitz-vpn-tax-01.txt, October, 1999.

[12] L. T. M. Berry, "A Mathermatical Model for Optimizing Telephone Networks," Ph. D. Dissertation, University of Adelaide, 1971.

[13] M. E. Beshai and R. Horn, "Toll Network Dimensioning under End-to-end Grade of Service Constraints," *Network Planning Symposium*, vol. 1, pp. 101-108, 1980.

[14] J. Burgin, "Broadband ISDN Resource Management," *Computer Networks and ISDN Systems*, vol. 20, pp. 323-331, 1990.

[15] J. Burgin and D. Dorman, "Broadband ISDN Resource Management: The Role of Virual Paths," *IEEE Communications Magazine*, vol. 29, no. 9, pp. 44-48, September, 1991.

[16] H. Cameron and J. Régnier, "Dynamic Routing for Intercity Telephone Networks," *International Teletraffic Congress*, vol. 10, pp. 3.2.1-3.2.8, June, 1983.

[17] E. Cavallero, U. Mocci, C. Scoglio, and A. Tonietti, "Optimization of Virtual Path/Virtual Circuit Management in ATM Networks," *Proc.of Networks'92*, pp. 153-158, May, 1992.

[18] A. Cayley, "On the mathematical theory of isomers," *Philosophical Magazine*, no. 67, pp. 444, 1874.

[19] K.-T. Cheng and F. Y.-S. Lin, "On the Joint Virtual Path Assignment and Virtual Circuit Routing Problem in ATM Networks," IEEE Globecom'94, pp. 777-782, December, 1994.

[20] K.-T. Cheng and F. Y.-S. Lin, "Virutal Path Assignment and Virtual Circuit Routing in ATM Networks," IEEE GLOBECOM'93, pp. 436-441, November, 1993.

[21] J. Chlamtac, A. Farago, and T. Zhang, "How to Establish and Utilize Virtual Paths in ATM Networks," Proc. of the ICC'93, pp. 1368-1372, May, 1993.

[22] J. Chlamtac, A. Farago, and T. Zhang, "Optimizing the System of Virtual Paths," *IEEE/ACM Transaction on Networking*, vol. 2, no. 6, pp. 581-587, December, 1994.

[23] C. T. Chou, "Traffic engineering for MPLS-based virtual private networks," *Computer Networks*, vol. 44, pp. 319-333, 2004.

[24] N. Christofides, *Graph Theory and Algorithmic Approach*, London, Academic Press Inc., 1986.

[25] M. De Prycker, "ATM Switching on Demand," *IEEE Network*, vol. 6, no. 2, pp. 25-28, March, 1992.

[26] Z. Dziong, *ATM Network Resource Management*, Mc Graw-Hill, 1997.

[27] Z. Dziong, B. Shukhman, and L. G. Mason, "Estimation of Aggregrate Effective Bandwidth for Traffic Admission in ATM Networks," *IEEE INFOCOM'95*, Boston, pp. 810-818, April, 1995.

[28] M. Eisenberg, "Engineering Traffic Network for More Than One Busy Hour," *Bell System Technical Journal*, vol. 56, pp. 1-20, 1977.

[29] M. Eisenberg, "Multi-Hour Engineering in Alternate-Route Networks," *International Teletraffic Congress*, vol. 8, pp. 132.1-132.6, November, 1976.

[30] W. B. Elsner, "A Decent Algorithm for the Multi-Hour Sizing of Traffic Networks," *Bell System Technical Journal*, vol. 56, pp. 1405-1430, 1977.

[31] A. Farago, S. Blaabjerg, L. Ast, G. Gordos, and T. Henk, "A New Degree of Freedom in ATM Network Dimensioning: Optimizing the Logical Configuration," *IEEE Journal of Selected Areas in Communications*, vol. 13, no. 7, pp. 1199-1205, September, 1995.

[32] M. Gerla, J. A. S. Monteiro, and R. Pazos, "Topology Design and Bandwidth Allocation in ATM Nets," *IEEE Journal of Selected Areas in Communications*, vol. 7, no. 8, pp. 1253-1262, October, 1989.

[33] A. Girard, *Routing and Dimensioning in Circuit-Switched Networks*, Reading, Massachusetts, Addison-Wesley, 1990.

[34] K. M. Girish, B. Zhou, and J.-Q. Hu, "Formulation of the Traffic Engineering Problems in MPLS based IP Networks," 5th IEEE Symposium on Computers and Communications., Los Alamitos, CA, USA, pp. 214-219, 2000.

[35] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis, "A Framework for IP Based Virtual Private Networks," Internet Draft, draft-gleeson-vpn-framework-03.txt, November, 1999.

[36] G. Gopal, C. Kim, and A. Weinrib, "Algorithms for Reconfigurable Networks," *13th International Teletraffic Congress*, Copenhagen, Denmark, pp. 341-347, June, 1991.

[37] G. Gopal, C. Kim, and A. Weinrib, "Dynamic Network Configuration Management," *IEEE International Conference on Communications (ICC'90)*, pp. 295-301, 1990.

[38] W. D. Grover, *Mesh-based survivable transport networks: Options and strategies for optical, MPLS, SONET, and ATM networking*, 1st ed, Prentice Hall, 2003.

[39] R. Guerin, H. Ahmadi, and M. Naghshineh, "Equivalent Capacity and Its Application to Bandwidth Allocation in High-Speed Networks," 7th ITC Seminar, Morristown, NJ, October, 1990.

[40] F. Guillemin and I. Hamchaoui, "Some Traffic Issues in the Design of Virtual Private Networks over ATM," *International Journal of Communication Systems*, vol. 9, no. 4, pp. 197-212, July-August, 1996.

[41] R. Huberman, S. Hurtubise, S. A. Le Nir, and T. Drwiega, "Multihour Dimensioning for a Dynamic Routed Network," *International Teletraffic Congress*, vol. 11, September, 1985.

[42] J. Y. Hui, et al.,, "A Layered Broadband Switching Architecture with Physical or Virtual Path Configurations," *IEEE Journal of Selected Areas in Communications*, vol. 9, no. 9, pp. 1416-1426, December, 1991.

[43] ITU-T, "Text of Draft Recommendation I.326: Functional Architecture of Transport Networks Based on ATM," July, 1995.

[44] B. Jamoussi, "Constraint-Based LSP Setup using LDP," Internet Draft, draft-ietf-mpls-cr-ldp-03.txt, September, 1999.

[45] M. R. G. a. D. S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-Completeness*, San Francisco, W.H. Freeman and Company, 1979.

[46] S. Katz, "Trunk Engineering of Nonhierarchical Networks," *International Teletraffic Congress*, vol. 6, pp. 142.1-142.8, 1971.

[47] R. Kawamura, K.-I. Sato, and I. Tokizawa, "Self-Healing ATM Networks Based on Virtual Path Concept," *IEEE Journal of Selected Areas in Communications*, vol. 12, no. 1, pp. 120-127, 1994.

[48] F. P. Kelly, "Routing in circuit-switched networks: Optimization, shadow prices and decentralization," in *Adv. Appl. Prob.*, pp. 112-144, 1988.

[49] S. Kent and R. Atkinson, "Security architecture for the Internet Protocol," RFC 2401, November, 1998.

[50] S. Kim, "An Optimal Establishment of Virtual Path Connections for ATM Networks," *IEEE INFOCOM'95*, Boston, pp. 72-79, April, 1995.

[51] L. Kleinrock, *Queueing Systems*, vol. Vol. II: Computer Applications, New York, Wiley-Interscience, 1976.

[52] J. E. Knepley, "Minimum Cost Design for Circuit Switched Networks," AD-A014 101, Defense Communications Agency Systems, Engineering Facility, Reston, Virginia, July, 1973.

[53] A. Kumar, Rastogi, R., Silberschatz, A. and Yener, B., "Algorithms for provisioning virtual private networks in the hose model," *IEEE/ACM transactions on networking*, vol. 10, no. 4, August, 2002.

[54] A. Kumar, Rastogi, R., Silberschatz, A. and Yener, B., "Algorithms for provisioning virtual private networks in the hose model," Bell Labs, Tech. Memo., 2000.

[55] M.-J. Lee and J. R. Yee, "A Design Algorithm for Reconfigurable ATM Networks," *IEEE INFOCOM'93*, pp. 144-151, 1993.

[56] W. W. Lee and J. W. Mark, "Capacity Allocation in Statistical Multiplexing of ATM Sources," *IEEE/ACM Transactions on Networking*, vol. 3, no. 2, April, 1995.

[57] K. Lindberger, "Dimensioning and Design Methods for Integrated ATM Networks," *14th International Teletraffic Congress*, Antibes, France, pp. 897-906, June, 1994.

[58] K. Lindberger, "Some Ideas on GOS and Call Scale Link-by-link Dimensioning," Technical Report COST 242 TD (92) 018, March, 1992.

[59] M. Logothetis and S. Shioda, "Centralized Virtual Path Bandwidth Allocation Scheme for ATM Network," *IEICE Transaction on Communication*, vol. E75-B, no. 10, pp. 1071-1080, 1992.

[60] E. Mannie and e. al., "Generalized Multi-Protocol Label Switching Architecture," Internet Draft, draft-ietf-ccamp-gmpls-architecture-07.txt, May, 2003.

[61] M. Meddeb, A. Girard, and C. Rosenberg, "The Impact of Tree Selection on the Design of Networks with Multipoint Connections," IEEE GLOBECOM' 97, pp. 1891-1897, 1997.

[62] M. Meddeb, A. Girard, and C. Rosenberg, "Tree Selection Heuristic for Network Design with Point-to-Multipoint Communications," 16th International Teletraffic Congress, pp. 645-656, 1999.

[63] D. Medhi, "Models for Network Design, Servicing and Monitoring of ATM Networks based on the Virtual Path Concept," *Computer Networks and ISDN Systems*, vol. 29, no. 3, pp. 373-386, 1997.

[64] D. Medhi, "Multi-Hour, Multi-Traffic Class Network Design for Virtual Path-Based Dynamically Reconfigurable Wide-Area ATM Networks," *IEEE/ACM Transactions on Networking*, vol. 3, no. 6, December, 1995.

[65] S. Mitra and D. Ghosh, "Configuring Express Pipes in Emerging Telecommunications Networks," *Telecommunication Systems 1*, 1993.

[66] S. Mitra, J. A. Morrison, and K. G. Ramkrishnan, "ATM Network Design and Optimization: A Multirate Loss Network Framework," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 531-543, 1996.

[67] E. Oki and N. Yamanaka, "An Optimum Logical ATM Network Design Method of Guaranteeing Multimedia QoS Requirements," IEEE Globecom'95, pp. 263-269, 1995.

[68] M. D. Pazos, J. A. Suruagy Monteiro, and M. Gerla, "Topological Design of Multiservice ATM Networks," Proceeding of SBT/IEEE ITS'94, Rio de Janeiro, Brazil, pp. 385-389, 1994.

[69] M. Pióro, Myslek, A., Jüttner, A, Harmatos, J. and Szentesi, Á. "Topological Design of MPLS Networks," GLOBECOM'01, San Antonio, Texas, pp. 12-16, November, 2001.

[70] C. W. Pratt, "The Concept of Marginal Overflow in Alternate Routing," *Australian Telecommunication Research*, vol. 1-2, pp. 76-82, 1967.

[71] Y. Rapp, "Planning of Junction Networks in a Multi-exchange Area. I:General Principles," *Ericsson Technics*, vol. 20, pp. 77-130, 1964.

[72] Y. Rapp, "Planning of junction networks in a multi-exchange area. II:extensions of the principles and applications," *Ericsson Technics*, vol. 21, pp. 187-240, 1965.

[73] Y. Rekhter and E. Rosen, "Carrying Label Information in BGP-4," Internet Draft, draft-ietf-mpls-bgp4-mpls-03.txt, September, 1999.

[74] J. Riordan, *An Introduction of Combinatorial Analysis*, New York, Wiley, 1958.

[75] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," RFC 3031, January, 2001.

[76] B. Ryu, H. Ohsaki, M. Murata, and H. Miyahara, "Design Method for Virtual Path Based ATM Networks with Multiple Traffic Classes," ICCC'95, pp. 336-341, 1995.

[77] F. Safaei and I. Ouveysi, "Optimal Capacity Allocation for Support of ATM Services over SDH Unidirectional Rings," Australian Telecommunication Network & Application Conference (ATNAC'95), Sydney, pp. 475-484, December, 1995.

[78] H. Saito, Y. Miyao, and M. Yoshida, "Traffic Engineering using Multiple multipoint-to-point LSPs," IEEE INFOCOM 2000, pp. 894-901, March, 2000.

[79] K.-I. Sato, S. Ohta, and I. Tokizawa, "Broad-Band ATM Network Architecture Based on Virtual Paths," *IEEE Transactions on Communications*, vol. 38, pp. 1212-1222, 1990.

[80] C. J. Truitt, "Traffic Engineering Techniques for Determining Trunk Requirements in Alternate Routing Trunk Networks," *Bell System Technical Journal*, vol. 33, pp. 277-302, 1954.

[81] P. Winter, "Steiner Problem in Networks: A Survey," *Networks*, vol. 17, pp. 129-167, 1987.

[82] N. Xiao, F. F. Wu, and S. Lun, "Dynamic Bandwidth Allocation Using Infinitesimal Perturbation Analysis," IEEE INFOCOM'94, pp. 383-389, 1994.