Improving Resource Management in Multi-Protocol Label Switched Traffic Engineered Networks

A Thesis

Submitted to the Faculty

of

Drexel University

by

Sukrit Dasgupta

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy in Computer Engineering

2008

Dedicated To

My Parents, *Baba* and *Ma* ...

For all their love and support, endless encouragement and teaching me how to be strong during testing times. I know you always watch over me.

My Grandparents, *Dadubhai* and *Didai* ...

For their unending love and encouragement.

My Sister, *Tukai* ...

For all her energy, love and encouragement to go for it.

My Beloved Wife, *Mimi* ...

For all her patience, love and support.

Acknowledgements

The number people who contributed to my education at Drexel is immense and I cannot thank everyone enough for just being the person they were.

Dr. Jaudelice C. de Oliveira, my thesis advisor and Jean-Philippe Vasseur, my collaborator at Cisco have been my mentors in every sense. They have guided and helped me through my toughest times and made sure I faced everything in the right spirit. From them I have learnt to do the hardest of things and face the stiffest of situations without any doubt in my mind. In them I have found two great teachers, colleagues and friends.

I thank my thesis committee chair Dr. Steven 'Darth' Weber and committee member Dr. Ali Shokoufandeh for being some of the best teachers I have ever learnt from. Their attitudes towards life and problem solving has been a strong motivation to push forward during long hours that seemed to have no end. I thank committee member Dr. Harish Sethu who never stopped encouraging me during the long hours we have discussed research and teaching. I also thank committee member Dr. Nagarajan 'Cobra' Kandasamy for always being his helpful and spirited self and patiently being a part of my committee. The professors at Drexel: Dr. Cohen and Dr. Petropulu for all the knowledge that I have gained through them, Dr. Prabhu, Dr. Ferrone, Dr. Venkat and Dr. Valliers for helping me when I needed it.

The terrific staff at the ECE Department who have helped me through everything and been great friends: Wayne, Stacey, Tanita, Dolores, Chad, Kathy, Brian and Jeremy. The great friends that made the experience most memorable and fun. My lab members: Rainer, Anbu Elancheziyan, Zhen Zhao, Josh Goldberg, Fei Bao and Joydeep Tripathi. My good friends at Drexel: Sankalp, Ananth, Nikhil, Sagar, Madhu, Uma, Prakash, Kalyan, Alankar, Murali, Brian and Nicholas. The nicest people at University of Girona: Dr Marzo, Fer, Luis-Fer, Lluis, Jorge and Juan.

Last but never the least I want to thank my family without whose love and support, this thesis would have been impossible. My parents, whose encouragement was endless. My grandparents whose love and support showed me through the tough times. My wife and sister for bringing all the lady luck I needed in my life as well as for their love and patience.

Table of Contents

List of Tables

List of Figures

Abstract

Improving Resource Management in Multi-Protocol Label Switched
Traffic Engineered Networks
Sukrit Dasgupta
Jaudelice C. de Oliveira, Ph.D.

Over the years, the Internet has emerged as an indispensable platform for information exchange. As availability increases, development of new applications generate enormous volumes of traffic. Such growth continually taxes service provider resources. A common and effective resource management option deployed by several service providers is Multi-Protocol Label Switched (MPLS) based Traffic Engineering (TE).

This dissertation proposes new MPLS based TE mechanisms capable of dealing with traffic changes, such as growth and shifts. Specifically, new techniques for dynamic bandwidth allocation and routing are proposed and developed through simulations under failure and non-failure scenarios. Issues related to inter-domain deployment are also studied and finally, an experimental testbed setup is proposed and implemented for realistic small scale testing.

A new traffic engineering technique involving the coupling of dynamic bandwidth allocation with rerouting to find the best path for the current traffic is proposed. Realistic topologies and traffic profiles are used for detailed analysis and comparisons with existing techniques. Performance analysis is also undertaken in an International network scenario carrying a mix of voice and data traffic across several timezones. Several key issues are highlighted after studying underlying network dynamics such as signaling overhead, router load, traffic path quality, etc. Keeping these issues in mind, a new trend-based bandwidth reservation mechanism is proposed. The problem of inter-domain TE is analyzed next. Existing inter-domain path computation approaches, signaling and path setup issues are studied, quantified and compared.

Lastly, the functional prototype of a testbed architecture consisting of Cisco routers and Linux boxes is presented. A new Java based API that has been developed to configure the testbed and deploy new mechanisms is also discussed.

1. Introduction

Communication services have increasingly become an indispensable tool in most social and economic activities of our daily lives. Over the years, telecommunication networks have faced explosive traffic growth as the popularity of the Internet and its ease of availability has steadily increased. Internet service providers are facing growing demands for supporting network sensitive applications such as Voice over IP, gaming, streaming media, corporate connectivity and more. These applications prove to be very resource intensive and cannot cope with poor network performance (high delays or low available bandwidth) or network instability (congestion and jitter). To add to this, routine events such as planned maintenance or unexpected equipment failure cause routing changes that may lead to transient service disruptions or persistent performance problems. With an increasing diversity of new applications having stringent demands emerging rapidly, it has become crucial to engineer a network capable of using its resources effectively. Networks should be able to quickly adapt to changes such as flash traffic, traffic shifts, misconfiguration or network element failure. To meet these challenges, the Internet Engineering Task Force (IETF) formed the Traffic Engineering Working Group and in 2002 formally coined the term "Traffic Engineering" as "that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks" [1].

1.1 Traffic Engineering

Traffic Engineering (TE) is more formally defined in [1] as the science of providing *efficient* and *reliable* network operations and simultaneously *optimizing* network resource utilization and traffic performance. This translates to three important objectives:

1. Network operations should be reactive to congestion and failure to become reliable and effecient.
2. User satisfaction and adherence to Quality of Service (QoS) requirements must be met by optimizing traffic performance.
3. Network resources such as available bandwidth, processing cycles, etc. must be consumed in an optimum fashion.

To meet these goals, [1] also outlines the class of problems relevant to Traffic Engineering that are faced when developing such functions. Simply stated, they are:

1. Explicit formulation of the problem that TE is trying to solve.
2. Identifying the requirements on the solution space.
3. Specification of the desirable features of a good solution.
4. Effective measurement of relevant network state parameters.
5. Effective characterization and evaluation of system level and network level performance under a variety of scenarios.
6. Optimization of network performance by translating solutions into network configurations which may also involve resource management control, routing control and/or capacity augmentation.

1.2 Background

The Traffic Engineering Working Group at the IETF did not mandate the use of specific technologies to implement effective TE. It was thus left to service providers to determine which technology would best address their objectives. Two approaches emerged; Internet Protocol (IP) and Multi-Protocol Label Switching (MPLS) based.

1.2.1 Internet Protocol

IP is a network protocol that is responsible for the routing of a data packet from the source to the destination where both the source and the destination are assigned unique IP addresses. Routing of traffic in a Service Providers' IP network within a routing domain uses an Interior Gateway Protocol (IGP). Almost all large IP networks deploy a link state IGP such as "Open Shortest Path First" (OSPF) [2] or "Intermediate System-Intermediate System" (IS-IS) [3]. Using static metrics/weights assigned to physical links by the service provider, both protocols compute the Shortest Path Tree (SPT) from a router to every reachable destination within the routing domain using the Dijkstra's Shortest Path First algorithm (SPF) [4]. At each hop, routers forward packets on particular interfaces based on the destination and the information stored in the forwarding table. Traffic flow in an IP based network can therefore be controlled by tuning the link metrics/weights used by the IGP.

1.2.2 Multi-Protocol Label Switching

MultiProtocol Label Switching (MPLS) was designed and standardized by the IETF to provide a platform for Traffic Engineering. IETF RFC 3031 [5] describes the basics of MPLS architecture. RFC 2702 [6] introduces the requirements for TE, whereas RFC 3209 [7] and RFC 3210 [8] provide necessary extensions to the Resource Reservation Protocol (RSVP) [9] for setting up traffic tunnel instances knows as Label Switched Paths (LSPs). Overview and principles of Internet TE are discussed in RFC 3272 [1] and the applicability statement for MPLS TE is given in RFC 3346 [10].

In the MPLS architecture, a short fixed-length label (called the shim header) is inserted between Layer 2 and Layer 3 of the protocol stack of a packet. The packet is then forwarded in the MPLS domain based on this label instead of the IP header. The label assignments are based on the concept of Forwarding Equivalent Class (FEC). According to it, packets belonging to the same FEC are assigned the same label and generally traverse through the same path across the MPLS network. An FEC may consist of packets that have common ingress and egress nodes or a combination of same service class and same ingress or egress nodes, etc. A path traversed by an FEC is called a Label Switched Path (LSP) or a Traffic Engineered-Label Switched Path (TE-LSP). The Label Distribution Protocol (LDP) [11] and an extension to RSVP [9] are used to establish, maintain (refresh) and tear-down LSPs.

Ingress and egress routers in an MPLS domain are called the Label Edge Routers (LER), while the routers that forward packets within the MPLS domain and also perform label swapping are called Label Switched Routers (LSR). At every LSR, forwarding decisions are taken based on the label of the incoming packet. Using this approach, traffic forwarding can be customized to meet Traffic Engineering objectives.

1.3 Research Objectives

This research addresses several intra- and inter-domain resource management issues faced by service providers today. In particular, the objectives are:

1. Identify system performance measures that accurately capture and quantify the important aspects of a network.

2. Investigate and quantify the performance of current traffic engineering technologies, namely IP and MPLS.

3. Investigate the problem of dynamic bandwidth allocation in MPLS-TE networks.

4. Design and develop a Dynamic TE technique to efficiently manage available bandwidth as well as meet QoS constraints. The proposed method couples dynamic bandwidth allocation and routing of TE-LSPs on the shortest path.

5. Observe and analyze the performance of the proposed Dynamic TE in the International network scenario with mixed voice and data traffic.

6. Analyze in detail the underlying signaling issues that are concomitant with the deployment of a TE mechanism such as Dynamic TE.

7. Design and develop a trend based bandwidth provisioning mechanism that keeps track of traffic history and current traffic trend and uses their information. The mechanism should also reduce signaling and meet QoS requirements of traffic.

8. Investigate inter-domain traffic engineering issues. Analyze and compare inter-domain path computation and path setup issues that arise when setting up TE-LSPs across multiple domains.

9. Design and develop a testbed that will provide a realistic platform for small scale testing of new TE mechanisms.

1.4 Thesis Outline

The thesis is organized as follows. Chapter 2 formalizes the system parameters and identifies the important performance measures that are used for performance analysis of the mechanisms presented in later chapters. It also discusses the relationship between the various performance measures and justifies the need to study them. Chapter 3 presents a performance comparison to motivate the use of MPLS as the platform for proposing new TE mechanisms. It identifies key performance measures to analyze the behavior of IP based and MPLS based TE and quantifies them for comparison. In Chapter 4, the Dynamic Traffic Engineering mechanism is proposed. This mechanism couples dynamic bandwidth allocation with dynamic routing. Comparisons are undertaken with Static TE where bandwidth reservations are based on peak traffic. Simulations are run for failure and non-failure scenarios. Policy based Dynamic TE to customize the mechanism as per the requirements of a service provider are then studied. The effect of different traffic profiles and timezones is studied next. Chapter 4 also analyzes in detail the affect of RSVP-TE [7] and OSPF-TE [12] signaling arising due to Dynamic TE. Based on the performance analysis of Dynamic TE, a Trend Based Bandwidth Provisioning Mechanism is proposed and its performance analyzed. Chapter 5 first discusses the issues with inter-domain path computation, signaling and setup. It then presents performance analysis of the two inter-domain path computation technologies that exist today. In Chapter 6, the architecture for an experimental router based MPLS testbed is presented. The motivation is to develop a platform that will allow the small scale yet realistic testing and deployment of new TE mechanisms. This chapter also discusses a software Java API that allows rapid deployment of new TE mechanisms on the testbed. Finally, Chapter 7 discusses the summary, future directions and concludes the thesis.

2. System Entities and Performance Metrics

In this chapter, we present some of the notations used to define the system. Important performance metrics that are used throughout the thesis are also defined and their significance discussed in Section 2.2. In Section 2.3, the relationships between the most significant metrics are presented.

2.1 Defining the System

In this section, we define the notation for a network as well as describe the functionality that exists on the routers. All elements in the network can easily be mapped to elements in a graph. For this thesis, the following correlation can be kept in mind: physical links in a network are the same as edges in a graph, routers are the same as vertices, and TE-LSPs are the same as flows. Therefore, these terms are interchangeable in this thesis.

2.1.1 Network Model

The network topology is denoted as $\mathcal{N} = (\mathcal{R}, \mathcal{L}, \mathcal{K})$. Here $\mathcal{R}$ denotes the set of all nodes, $\mathcal{L}$ denotes the set of all edges and $\mathcal{K}$ denotes the set of all flows. Vector $\mathbf{c} = (c_l, l \in \mathcal{L})$ denotes the physical link capacities, $\mathbf{a^{phy}} = (a_l^{phy}, l \in \mathcal{L})$ denotes the instantaneous physical available capacity of the links and $\mathbf{w} = (w_l, l \in \mathcal{L}, w_l \in \mathbb{Z}^+)$ denotes the weights assigned to the edges. Vector $\boldsymbol{\lambda^{max}} = (\lambda_k^{max}, k \in \mathcal{K})$ and $\boldsymbol{\lambda^t} = (\lambda_k^t, k \in \mathcal{K})$ denote the set of peak and instantaneous reservation values for the TE-LSPs respectively.

2.1.2 Functionality on a Router

A router is the fundamental network equipment. It is highly intricate and complicated and comprises of several functional entities. For accurate performance analysis of the mechanisms discussed in this thesis, the significant and relevant aspects of router functionality have been abstracted and presented in this section.

1. **Topology Database.** Every router r contains an entity called the topology database $\mathcal{D}^r = (\mathcal{R}^r, \mathcal{L}^r)$. Here, $\mathcal{R}^r \subseteq \mathcal{R}$ and $\mathcal{L}^r \subseteq \mathcal{L}$ denote the set of all routers and links whose information is present in the topology database of node r. It is also referred to as the Traffic Engineering Database (abbreviated as TED or TE-DB in the literature and this thesis).

2. **Path Computation Entity.** The path computation entity on the router is responsible for computing paths given a source, destination and bandwidth. This entity returns the set of links that satisfy the bandwidth requirement and forms the shortest such path from the source to the destination. The set of paths for the TE-LSPs is denoted as $\mathcal{P}$. The vector $\boldsymbol{p_k} = (l|l \in \mathcal{L})$ where $k \in \mathcal{K}$ denotes the edges that form the path for TE-LSP k. The path cost is the sum of edge weights that form it:

$$w(\boldsymbol{p_k}) = \sum_{i=1}^{|\boldsymbol{p_k}|} w_l$$

The shortest path cost $\delta(u, v, \lambda_k)$ corresponding to a TE-LSP k with request λ_k from source u to destination v is be represented as:

$$\delta(u, v, \lambda_k) = min \begin{cases} w(p) : u \rightsquigarrow^{p} v & \text{if there is a path from } u \text{ to } v \text{ that can fit } \lambda_i \\ \infty & \text{otherwise} \end{cases}$$

For path computation, the router uses the topology database $\mathcal{D}^r$ that exists on it. The shortest path that meets a bandwidth requirement of λ_k is computed using the Constrained Shortest Path First (CSPF) algorithm. This involves the pruning of links where available bandwidth is less than the request bandwidth followed by the use of Dijkstra's Shortest Path algorithm on the resulting subgraph. With every path computation, there is an associated processing overhead incurred on the node's processing unit. We use d_{proc} to denote the total processing delay arising from path computation.

3. **TE-LSP Path Setup.** Setting up TE-LSP k on a path $\boldsymbol{p_k}$ involves the reservation of bandwidth on the links that form the path. Vector $\boldsymbol{a^{phy}}$ gets updated as $\boldsymbol{a^{phy}} = (a_l^{phy} - \lambda | l \in \mathcal{L}, l \in \boldsymbol{p_k})$ since the corresponding amount of physical bandwidth is not available in the network after reservation. Figure 2.1 shows the steps involved in setting up a TE-LSP.

   If TE-LSP k is being setup for the first time from head end LSR r, then $\boldsymbol{a}^r$ is also updated as $\boldsymbol{a}^r = (a_l^r - \lambda | l \in \mathcal{L}, l \in \boldsymbol{p_k})$. In other situations such as a reroute due to a link failure or if a better path is found, $\boldsymbol{a}^r$ is not updated.

   There is significant delay associated with TE-LSP path setup. The head-end router r sends out the RSVP-PATH [7] packet along the computed path. On receiving this packet, the destination router then responds with the RSVP-RESV [7] packet that reserves the requested bandwidth on all the links in the path. The total delay incurred is proportional to the length of the path and is denoted by $d_{setup} * \boldsymbol{p_k}$. Processing of the PATH and RESV packets incur a processing delay of d_{rsvp}^{proc} on each mid-point router leading to a total delay of $d_{rsvp}^{proc} * 2(\boldsymbol{p_k} + 1)$.

4. **Call Admission Control (CAC).** Every router contains a link management

module that keeps track of the actual physical bandwidth available on the adjacent links. This module is also responsible for admission of TE-LSPs that are either being setup from the same router or other routers. Since $\boldsymbol{a}^r$ is not always updated on every change to $\boldsymbol{a^{phy}}$, there are situations when a TE-LSP cannot be setup on a path. This happens when a head-end LSR computes a path based on outdated bandwidth information in $\boldsymbol{a}^r$. In such situations, the link management module causes a CAC failure/rejection. Let F_l^{CAC} denote such a failure situation on link l, it can be represented as:

$$F_l^{CAC} = \textbf{true} \text{ if } \begin{cases} \lambda_k \leq a_l^r, \text{where } a_l^r \in \boldsymbol{a}^r \\ a_l \leq \lambda_k, \text{where } a_l^{phy} \in \boldsymbol{a^{phy}} \\ l \in \boldsymbol{p_k} \end{cases}$$

On a CAC failure, the routers adjacent to the corresponding link flood the network with the current available bandwidth on the link. On receiving this information, all the routers update their respective topology databases, path computation takes place for the concerned TE-LSP again and the head-end router tries to set it up again.

5. **Routing Protocol Signaling.** The underlying routing protocol (Interior Gateway Protocol or IGP) considered is Open Shortest Path First (OSPF) with Traffic Engineering (TE) Extensions. OSPF-TE employs Link State Advertisements (LSAs) to disseminate network information. LSAs are flooded on three occasions:

   (a) **Connectivity Change.** Whenever a router or a link is added or removed from the network, a new LSA is flooded by the routers that are adjacent to it. Link and router failure situations are most common causes of con-

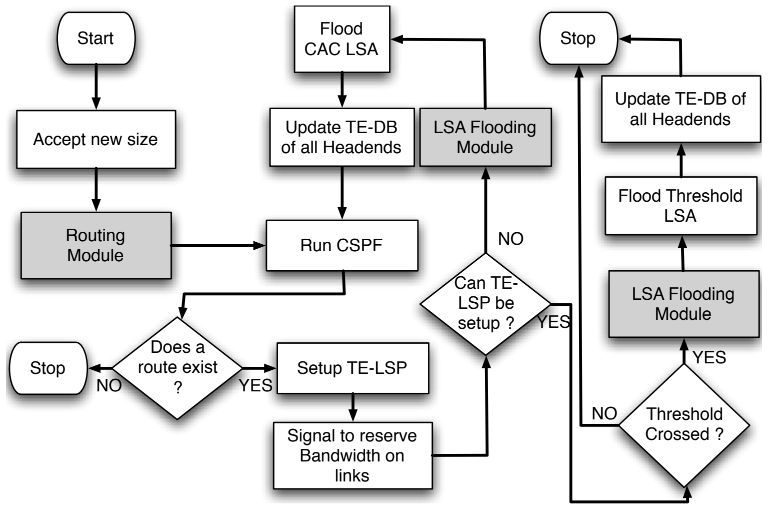


Figure 2.1: Sequence of steps involved to setup or reroute a TE-LSP.

nectivity change, they result in an LSA flood corresponding to their failure or restoration. On a link failure, the following updates take place:

$\mathcal{L} = \mathcal{L} - l$ (the link is removed from the set of all working links)

$\boldsymbol{a^{phy}} = \boldsymbol{a^{phy}} - a_l^{phy}$ (no available bandwidth associated)

$\boldsymbol{e^r} = \boldsymbol{e^r} - e_l^r$ and

$\boldsymbol{a^r} = \boldsymbol{a^r} - a_l^r$ (removed from the topology database of every router r)

On a router failure, the links adjacent to it also go down. The following updates take place for all links l adjacent to the failing router r.

$\mathcal{L} = \mathcal{L} - l$ (the link is removed from the set of all working links)

$\boldsymbol{a^{phy}} = \boldsymbol{a^{phy}} - a_l^{phy}$ (no available bandwidth associated)

$\boldsymbol{v^r} = \boldsymbol{v^r} - v_l^r$

$\boldsymbol{e^r} = \boldsymbol{e^r} - e_l^r$ and

$\boldsymbol{a^r} = \boldsymbol{a^r} - a_l^r$ (removed from the topology database of every router r)

(b) **Change in Available Bandwidth.** A pair of a set of fractions $\boldsymbol{\alpha^{up}} = \{\alpha_i^{up} | \alpha_0^{up}, \alpha_1^{up} \ldots \alpha_n^{up}\}$ and $\boldsymbol{\alpha^{down}} = \{\alpha_i^{down} | \alpha_0^{down}, \alpha_1^{down} \ldots \alpha_n^{down}\}$ are used for flooding an LSA whenever the physical available bandwidth a_l^{phy} on a link increases or decreases and crosses the threshold $c_l * \alpha_i^{up/down}$.

(c) **Periodic Flooding.** Every router periodically sends out an LSA corresponding to the links that it is adjacent to. This keeps the routers in the network updated and reduces the chances of setup failures or suboptimal path computations.

2.2 Performance Meaures

This section presents those performance measures that are common to all mechanisms discussed in this thesis. In addition to these, performance measures specific to corresponding mechanisms are presented in their respective sections.

2.2.1 Motivation

In a complex system like a TE network, there are several entities whose performance are inter-dependent on each other. It is very important to capture the performance of every entity in detail to observe this inter-dependence and finally design TE mechanisms keeping these aspects in mind. The common aspects of a network that need to be studied for all new mechanisms are Path Quality, Router State, Signaling, Network Utilization and MPLS Tunnel Reservation. Optimizing any one of these aspects either has an effect on the remaining or in some way is related to the optimization of another one. During the design of TE mechanisms, knobs are put in place to tune the performance according to requirements and available resources. Performance measures that capture the several facets are studied through detailed simulations.

2.2.2 Path Quality

1. **Number of TE-LSPs Not on Their Shortest Path.** If a TE-LSP is setup on the shortest path, it corresponds to traffic following favorable links, thus resulting in less end to end delay and better adherence to QoS constraints. Since providers can generate more revenue from traffic following favorable paths, a higher number of TE-LSPs on their shortest path is favorable.

2. **Path Cost Ratio.** Path Cost Ratio q_k^t for a TE-LSP k at time t is the ratio of current path cost to the shortest path cost for the corresponding constraint. The shortest path cost is calculated using a 'greenfield scenario' where no flows exist in the network. Again, two perspectives to this metric can be analyzed. The distribution of the average path cost ratio across all TE-LSPs and the distribution of the average path cost ratio across total traffic.

3. **Fraction of Times a TE-LSP Follows the Same Path on Resizing.** This metric captures the fraction of times a TE-LSP was on the same path after resizing to a new size. Staying on the same path reduces jitter that arises when traffic is forwarded onto new paths with different delays.

2.2.3 Router State

1. **Number of TE-LSPs per Router.** A router has to maintain state for every TE-LSP that passes through it. This involves memory allocation and processing overhead to main that state. It is ideal for routers to be lightly loaded in terms of the number of TE-LSPs passing through them.

2. **Maximum Router Load.** The number of TE-LSPs passing a node varies with time. The maximum number is of interest since this allows one to keep in mind the worst-case load in the network and its possible influence on the

network in the case of router failure. This metric allows us to study how the maximum number of TE-LSPs passing through a node can increase and shift in the network when new mechanisms are deployed.

3. **TE-LSP Setup Request.** A need to reroute a TE-LSP arises either when a new mechanism requires it, route reoptimization is required or when failures occur. Nodes in the network periodically receive requests for setup and tear down for these events. This metric captures the maximum number of such requests a node has received over the period of simulation time. The motivation for studying this quantity is to find the maximum stress experienced by a node in terms of processing such requests and is of great interest to measure the network stability in term of traffic shift resulting from route changes.

2.2.4 Signaling

1. **CAC Failures.** As mentioned previously, CAC failures take place due to outdated topology databases on routers. Once a midpoint routers experiences a CAC failure, significant delay is introduced in the setup of a TE-LSP. It is thus important to keep the number of CAC Failures down to a minimum.

2. **Available Bandwidth Thresholds.** When TE-LSPs are being rerouted in the network, reserved bandwidth changes in the links and the availability of bandwidth changes. This causes the underlying routing protocol to flood LSAs in the network. A high degree of signaling causes a lot of noise and is undesirable.

3. **TE-LSP Resizing.** Chapter 4 discusses the dynamic resizing of MPLS Tunnels. In this respect, it is important to keep the number of times a tunnel reservation is resized. Resizing involves the complete process of path computa-

tion and path setup using RSVP. This could in turn also result in underlying routing protocol signaling.

2.2.5 Network Utilization

1. **Link Reserved Bandwidth.** The average and maximum reservation experienced by links show the available bandwidth of a network. For analysis, this metric is studied as a distribution across all links and compared for different TE mechanisms.

2. **Link Utilization.** Link Utilization is the ratio of the total traffic passing the link and the link capacity. A distribution of the average utilization is analyzed over all the links.

2.2.6 Tunnel Reservation

1. **Fraction of Underbooking and Overbooking.** There may be many instances where the traffic flowing through a TE-LSP is either less or more than the corresponding reserved size. A situation of underbooking arises when the amount of traffic flowing through a TE-LSP is more than its current reserved size. The fraction of underbooking f_u^t and overbooking f_o^t at any instance t for TE-LSP k are correspondingly expressed as

$$\mathrm{f}_{u(t)}^k = (bps_k^t - \lambda_k^t)/\lambda_k^t \tag{2.1}$$

$$\mathrm{f}_{o(t)}^k = (\lambda_k^t - bps_k^t)/\lambda_k^t \tag{2.2}$$

2.3 Relation between the Measures

In this section, the relationship between the significant metrics is discussed. It is important to know how the behavior of a particular metric is related to another. Analysis here will help during the design phase of TE mechanisms.

2.3.1 Path Quality and Number/Size of TE-LSPs

Consider the residual network in Figure 2.2(a) where all the links are of same capacity c_l and a flow of constant size λ ($\lambda < c_l$) is to be setup from the source (node 1) to the sink (node 7). Using Dinic's algorithm we form a layered network $l_1$in Fig 2.2(c) and setup λ as an augmenting flow.

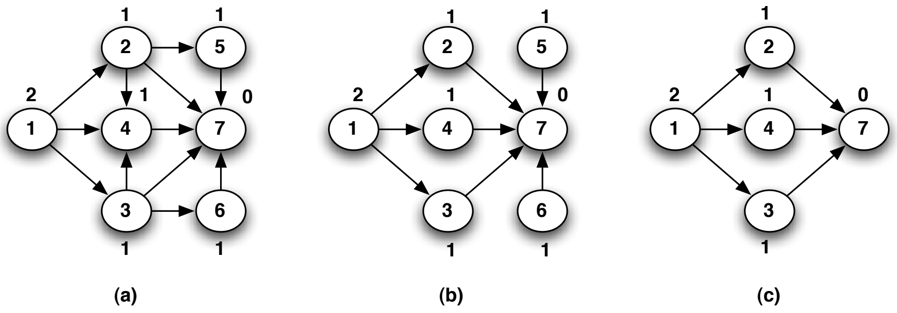


Figure 2.2: Forming layered networks: (a) residual network; (b) corresponding layered network; (c) layered network after deleting redundant arcs.

If similar flows from a different source to the same sink were to be setup, eventually, the layered network l_1 for source 1 would change since the residual network has changed. It has been shown that whenever a layered network changes when using the Dinic's algorithm, the distance label of the source node strictly increases [4]. This shows that the new flow being setup from node 1 is *not* on its shortest path.

To have the same layered network l_1 when the second flow is being setup from source 1, the minimum augmenting flow should have at least value $\lambda \leq c_l - n * \lambda$ where n is the number of flows setup from other sources through the arcs of the layered network l_1. Thus, it can be seen that the possibility of a flow not being on the shortest path is dependent on the number of flows setup before it and on the size of the flows.

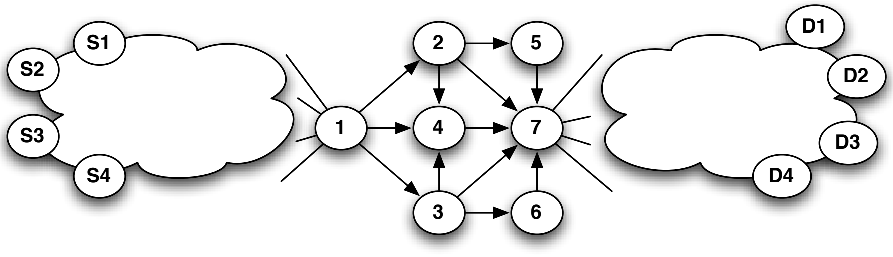


Figure 2.3: Scaling the network scenario.

This case can be generalized as shown in Figure 2.3. The network in Figure 2.2 can be a part of a much larger network and still result in the situation above.

2.3.2 Router Load and Number/Size of TE-LSPs

Using the example scenario from the previous section the following can be deduced. If $\lambda \ll c_l - n * \lambda$ then several augmenting flows can be fit before any arc gets saturated. This will lead to an increase of the number of flows passing through a node. Conversely, if $c_l - \sum_i^K \lambda_k < \lambda_i$, a new layered network will be formed and the corresponding flow will be on a longer path. As more flows of larger sizes need to be setup, many layered networks will be formed reducing the number of flows that pass through a node.

2.3.3 Call Admission Failure and Number/Size of TE-LSPs Present

In an empty network, when a TE-LSP k of size λ is setup on a path $\boldsymbol{p_k}$, available bandwidth in the network is changed: $\boldsymbol{a^{phy}} = (c_l - \lambda | l \in \mathcal{L}, l \in \boldsymbol{p_k})$ the topology database of router r is updated: $\boldsymbol{a}^r = (a_l^r - \lambda | l \in \mathcal{L}, l \in \boldsymbol{p_k})$. If another TE-LSP g is setup from router m on path $\boldsymbol{p_g}$, only $\boldsymbol{a^{phy}}$ and $\boldsymbol{a^m}$ are updated. If $\boldsymbol{p_g} \cap \boldsymbol{p_k} \neq \phi$ the topology database of router r will now be out of sync. For every link $i \in \boldsymbol{p_g} \cap \boldsymbol{p_k}$ there is a difference $\Delta_i^r = a_i^r - a_i^{phy}$ with respect to link i in the topology database of router r. Similarly, if there are n TE-LSPs setup from different head-end LSRs, and if each has at least one common link with TE-LSP k, then in the worst case, $\Delta_i^r = n * \lambda$.

CAC failures on a link i for setup request of λ occur when $a_i^r - \Delta_i^r < \lambda$. Thus it can be seen that the chances of a CAC failure occurring is proportional to the number of flows and/or the size of flows present in the network.

3. Traffic Engineering with MPLS and IP

Unexpected traffic variations in networks prove to be one of the most daunting management issues service providers face today. They can be caused due several reasons and often lead to deteriorating Quality of Service (QoS). To address this issue, service providers use Traffic Engineering (TE) techniques. IP and MPLS based TE react differently to the new arrival of traffic. IP uses the *'Shortest Path First' (SPF)* algorithm to route all traffic on the shortest available path. MPLS being based on Constraint-Based routing, uses the CSPF [13] algorithm to route the traffic on the shortest available path that *fits* it. With IP, this may result in situations where links are over-utilized due to common links that lie on the shortest path for several source-destination pairs. In this chapter, we bring this behavior to light and motivate the use of MPLS for proposing new TE techniques.

3.1 Introduction

Various traffic engineering techniques have been developed over the past few years for Internet Protocol (IP), based on Interior Gateway Protocol (IGP) link cost computation [14; 15], and Multi-Protocol Label Switching (MPLS) [6], based on constraint based routing using traffic engineered label switched paths. Driven by network characteristics such as topology, traffic matrix, and traffic profile, there are circumstances where in steady state, both traffic engineering approaches achieve similar degree of network optimization. There are however several common scenarios of network congestion or “hot-spots” where prevailing practices of IP traffic engineering fall short since dynamic adjustment of the IGP link metric is not a viable option. MPLS Traffic Engineering when combined with distributed head-end Constraint Shortest Path

First (CSPF) [13] computation provides the ability to forward traffic along paths dynamically determined by the network state. This approach achieves similar network optimization and in particular avoids congestion spots that would likely compromise QoS. While several network recovery techniques for IP and MPLS based networks have been extensively studied.

3.1.1 Motivation

The goal is to quantify the behavior of IP and MPLS based TE approaches when faced with unexpected traffic variations that lead to network overloading. With this objective in mind, and in order to simulate realistic network overloading, link failure events are introduced in the network topologies studied to cause traffic shifting and variations. We emphasize that failure scenarios are introduced merely to cause realistic traffic variations in the topologies under study.

3.1.2 Related Work

Routing of traffic in Service Providers' IP networks within a routing domain uses an Interior Gateway Protocol (IGP). Almost all large IP networks deploy a link state IGP such as "Open Shortest Path First" (OSPF) [2] or "Intermediate System-Intermediate System" (IS-IS) [3]. Using the static metrics assigned to physical links by the service provider, both protocols compute the Shortest Path Tree (SPT) from a router to every reachable destination within the routing domain using the Dijkstra Shortest path first algorithm (SPF) [4]. Each router maintains a view of the network topology and set of reachable IP prefixes using a Link State Database (LSDB). The LSDB is populated with Link State Packets (abbreviated as LSP with IS-IS) or Link State Advertisements (abbreviated as LSA with OSPF) that are originated at each router and flooded across the routing domain or area in a reliable fashion. Each time the network topology changes (e.g. network element failure, addition of a new link or node, addition/deletion of reachable IP prefixes), the LSDB is updated and each router recomputes its routing table. Having an approximate knowledge of the traffic matrix (traffic demand), off-line link metric computation algorithms can be used such that specific performance objectives, such as minimizing the maximum link utilization, path cost bounds, etc. are met and the traffic's Service Level Agreements (SLAs) are adhered to. Such techniques have been proposed by researchers in [15; 16; 17] where the goals are to find a *static* optimal link weight setting taking into account traffic variations and failures. Although good results can be obtained in steady state (provided that the traffic matrix is known), it is significantly more challenging to compute an efficient set of link metrics that satisfies the objective requirements for every failure case.When the state of a network changes, providers resort to changing link metrics to alleviate the situation and prevent large traffic shifts. This action, though intuitive, has the following shortcomings:

Global Impact. Each link metric change in the network has a global impact on every router present in the domain: new LSAs are flooded, the LSBDs are re-synchronized and the routing tables are recomputed. This also affects the routers that are routing flows not impacted by the congestion spot.

Micro-Loops. Each link metric change leads to traffic reroutes. One of the side-effects is the potential formation of temporary micro-loops caused by the non-synchronized updates of the router's Routing/Forwarding Information Base (RIB/FIB). The micro-loops potentially cause packet drops for the rerouted flows and additional delays for flows not impacted by the original congestion. Although heuristics exist to limit the number of such micro-loops, they cannot always be avoided.

Lack of Granularity. Changing the link metric does not provide a high granularity and control on the set of flows that are subject to reroutes.

Slow Reactive Process. A link metric change involves the network operator or an off-line management system and is thus a relatively slow process.
Although several techniques are being designed to remove or limit some of these drawbacks (e.g. use of Forwarding Information Base (FIB) ordering to avoid micro-loops during convergence [18], improvements in terms of LSA flooding handling and route tables computation), advantages of off-line reactive adjustment of the link metrics remain marginal.

3.1.3 Contribution

The contribution of this work is the identification and quantification of metrics that allow a comparison of IP and MPLS Traffic Engineering techniques. Based on the analysis of the performance results obtained, we motivate the use of MPLS as the technological platform to carry out Traffic Engineering research for the rest of the thesis. Results from this work have been published in [19]

3.2 MPLS Based TE

MPLS relies on the label switching technique. In MPLS, the packets are assigned a label and the routers determine the outgoing interface for the packets based on their label. The forwarding decision does not rely on the destination IP address. The fundamental idea of MPLS is to forward traffic across a network along a Traffic-Engineered Label Switched Path (TE-LSP). This path is determined by taking into account the network topology and traffic constraints such as bandwidth requirements, delay and jitter bounds, etc. Considering the network topology and the traffic constraints mentioned above, there are two basic ways in which TE-LSPs can be computed: off-line and online. The off-line approach usually provides a better degree of optimality compared to the on-line approach which involves each router computing the path of its

TE-LSPs using an algorithm such as CSPF. It must be noted that the difference in terms of optimality between the two path computation techniques is topology dependent. A shortcoming of the off-line approach lies with its inability to react quickly to topology and traffic changes in the network. In this chapter, the on-line approach (the one largely deployed in present-day TE-enabled networks) is considered.

Although there is a wide range of proposed CSPF algorithms significantly differing from each other, in its simplest form, it consists of selecting the shortest path that satisfies the TE-LSP constraints (such as the bandwidth) according to a metric (IGP or Traffic Engineering metric). CSPF uses the Traffic Engineering Database (TED) [12; 20] to prune links not satisfying the given constraint and then runs Djikstra's Shortest Path First algorithm on the resulting subgraph to find the shortest path. The SPF uses static link metrics assigned to the links for path calculation. After a path has been determined, a resource reservation protocol such as RSVP-TE [7] signals the TE-LSP and traffic is then forwarded onto the TE-LSP.

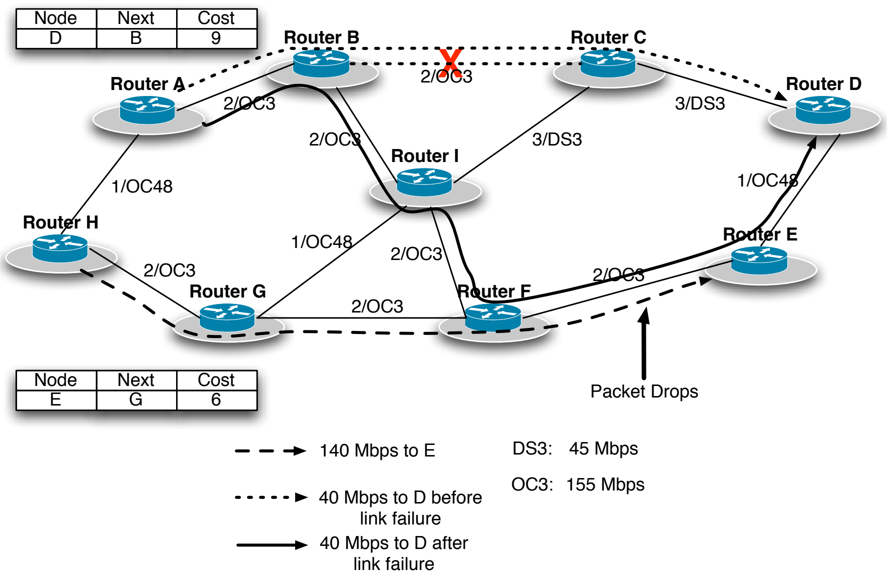


Figure 3.1: Routing of traffic using IP during a link failure.

A simple scenario of network link failure and the change in route is illustrated in Figure 3.1 and Figure 3.2. There are two flows of 140Mbps from Router C to Router E and 40Mbps from Router A to Router D. IP routes the traffic on the shortest path corresponding to the destination. A failure of link Router B-Router C triggers the re-computation of the routing tables on every router followed by a traffic reroute.

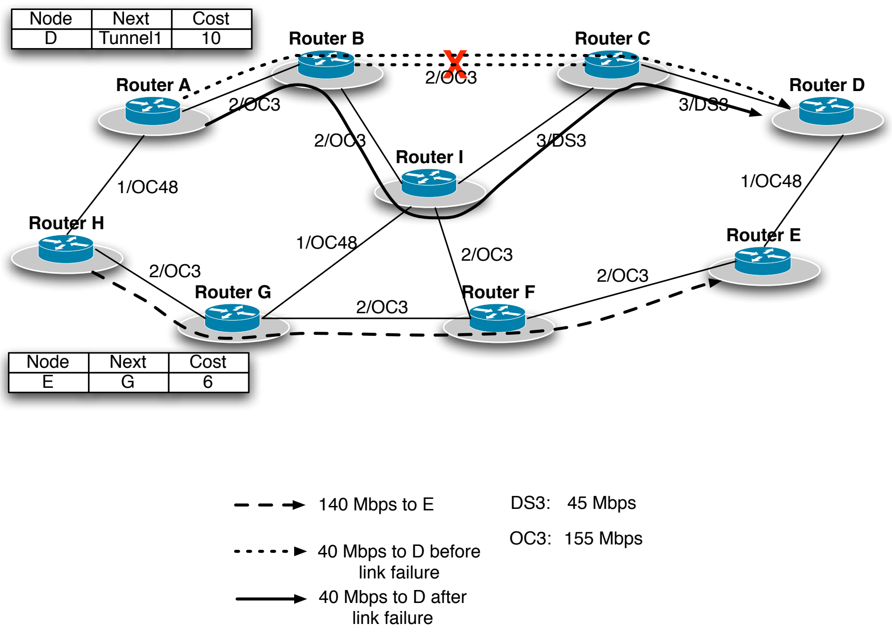


Figure 3.2: Routing of traffic using MPLS-TE during a link failure.

Since IP always routes traffic onto the shortest path and because the capacity of a OC3 link is 155Mbps, congestion takes place on link Router F-Router E as the existing traffic on that link is not taken into account. In the same scenario, since MPLS-TE is constraint based, the existing traffic on link Router F-Router E is considered. Constraint based routing then routes the traffic on links that can accommodate it. Although the path computed by the constraint based routing algorithm is longer, it is the shortest path that can fit the traffic.

3.3 Simulation Results

To fairly compare the performance of a network that employs Traffic Engineering based on IP and one that deploys MPLS Traffic Engineering, the same set of conditions for testing need to be imposed.

3.3.1 Simulation Setup

In order to have a common initial common ground for IP and MPLS-TE, the traffic is scaled such that the networks used for comparison are in steady state. In our study, all TE-LSPs follow the IGP shortest path in the steady state. In the case of IP, this simply translates to all flows following their shortest path. Four realistic topologies are considered, named MESH, SYM, ISP1 and ISP2 and detailed in Table 3.1. Details of TE-LSPs setup in the network is shown in Table 3.2.

Table 3.1: Details of the topologies used in simulations

Type	Network Description							
	Name	# of nodes	# of links	OC3 links	1 Gbps links	OC48 links	OC96 links	OC192 links
Core	MESH	83	167	0	0	132	0	35
Edge	SYM	66	127	40	80	0	0	7
Core	ISP1	158	280	65	0	92	0	123
Core	ISP2	88	168	0	0	49	9	110

Link failures are then induced in each network in order to simulate traffic variations. Every link can independently fail uniformly with mean time 30 minutes. Similarly, a failed link is restored independently and uniformly with mean time 15

Table 3.2: Details of the TE-LSPs used in simulations

Type	Name	TE-LSP Description		
		[0,20) Mbps	[20,50) Mbps	(50,1500] Mbps
Core	MESH	5105	1701	0
Edge	SYM	2655	885	0
Core	ISP1	1916	280	346
Core	ISP2	1510	149	188

minutes. When a link failure occurs, the traffic traversing it is rerouted. In the case of IP, the shortest path algorithm computes the next shortest path considering the failure. In the case of MPLS, each TE-LSP affected by the failure is rerouted along the shortest path satisfying the constraints (e.g. bandwidth). Inter-failure and inter-restoration times of all links are the same for simulations involving IP and MPLS-TE to prevent the results from being biased. The traffic profile used is shown in Figure 3.3

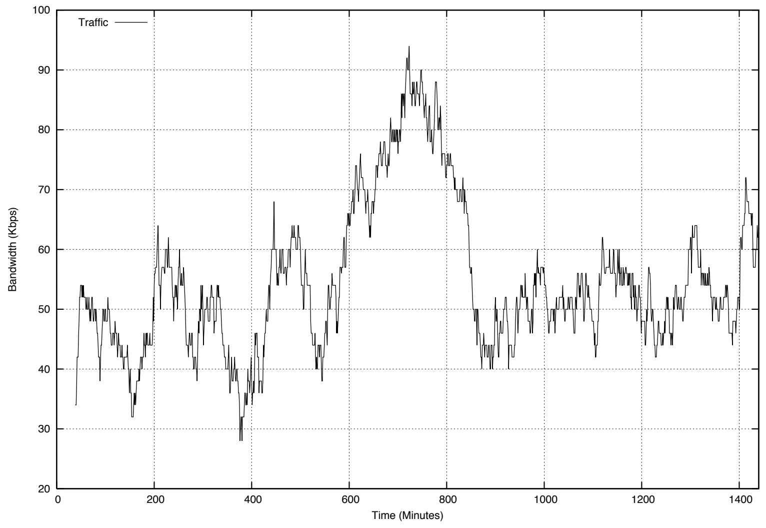


Figure 3.3: Traffic profile used in the simulations.

3.3.2 Performance Metrics

Two metrics are observed and analyzed. Link Utilization and Path Cost Ratio, both of which are discussed in Chapter 2.

Two perspectives of Link Utilization are observed, the time varying distribution of instantaneous link utilization across all the links and the distribution of the maximum link utilization experienced over the whole simulation period. Link utilization serves as an indicator of congestion and queuing delays and the maximum utilization experienced by a link is particularly important since it characterizes the worst case scenario.

The second metric quantifies the path quality of traffic. IP always keeps traffic on the shortest path whereas MPLS-TE by virtue of being constraint based may not always follow the shortest path possible corresponding to the constraint. MPLS-TE reroutes traffic along non IGP shortest paths in order to avoid congestion. The average increase in path cost is thus interesting since it effectively measures the relative increase in the length of such paths and characterizes changes in end to end delay.

3.3.3 Performance Analysis

1. **Link Utilization**

   Observation. Figure 3.4 shows the distribution of the maximum link utilization of all the links over the simulation period and the time varying behavior of the utilization of the links. It shows that with traffic shifts caused by link failure, 20% of the links experienced a larger maximum utilization with IP than with MPLS-TE. Also, with IP, almost 14% of the links at some point crossed a utilization of 100% out of which more than 5% of the links crossed a utilization of 150%. Figure 3.5 shows that 30% of the links experience a higher utilization under failures with IP than with MPLS-TE. Also, with IP, 24% of the links at

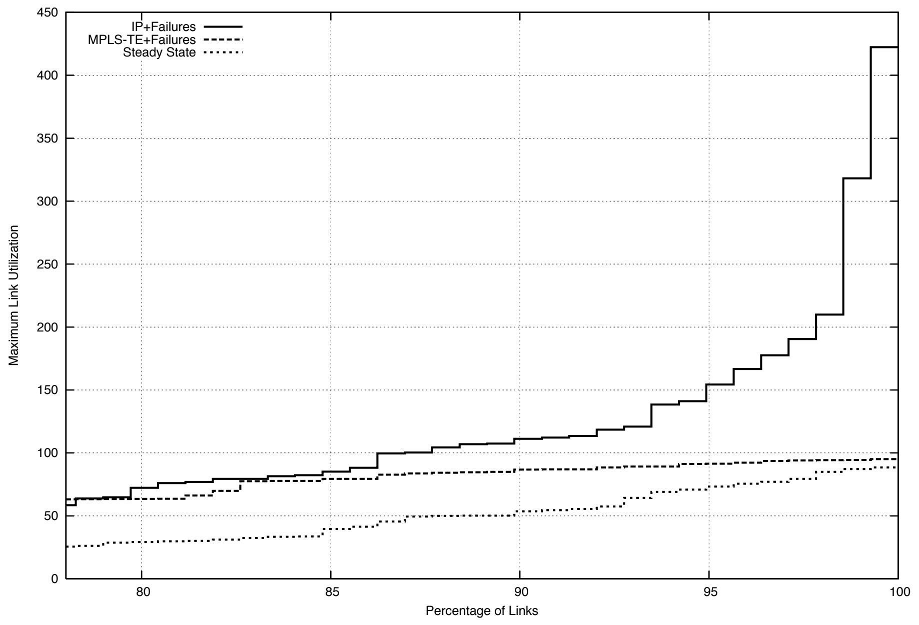


Figure 3.4: Distribution of maximum link utilization: MESH.

some point, experience a utilization of more than 100%. In such a situation, the links get highly congested and traversing packets start to experience non-linear delays and jitter, potentially leading to packet drop.

Hypothesis. This behavior can be attributed to IP being destination based and thus always selecting the shortest path. In some situations, the shortest path to a destination may have several common links lying on the shortest paths to other destinations. MPLS-TE, by its nature of being constraint based, computes TE-LSPs so as to avoid traffic congestion by selecting paths such that the required bandwidth is available. While this behavior is expected, to the best of our knowledge, this metric had not been quantified in previous studies, and it reveals a very unique look at the network state when the two TE techniques are employed.

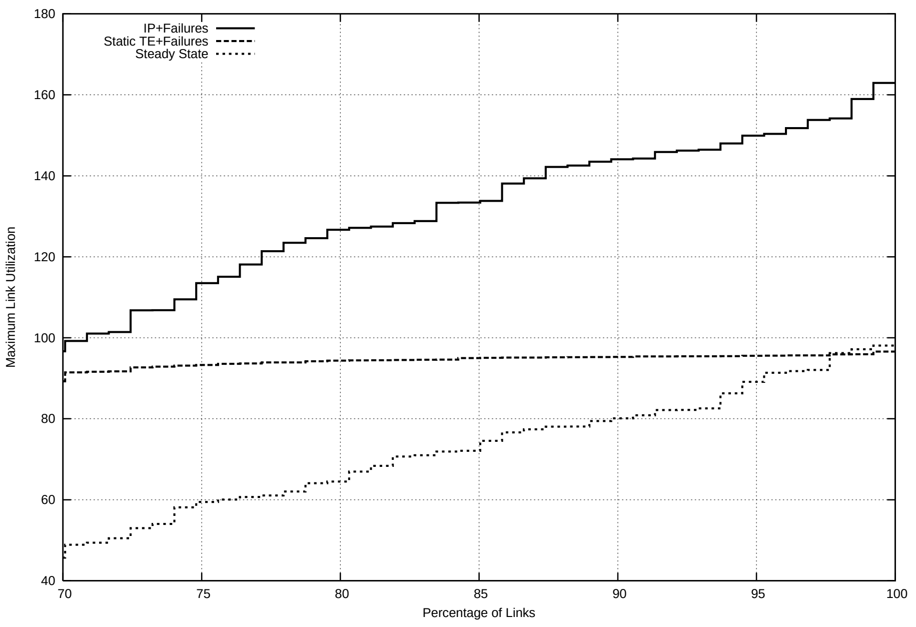


Figure 3.5: Distribution of maximum link utilization: SYM.

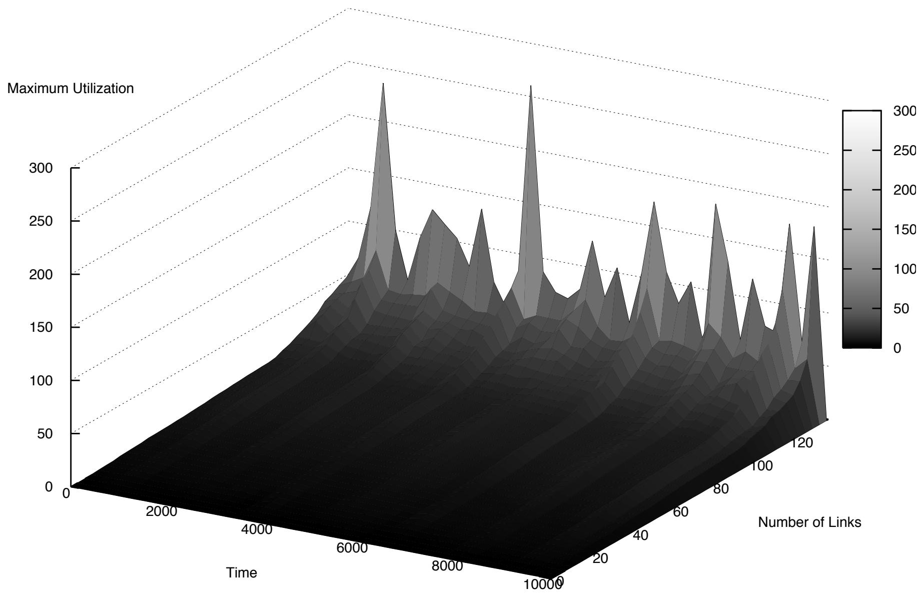


Figure 3.6: Distribution of link utilization with time for failures under IP: MESH.

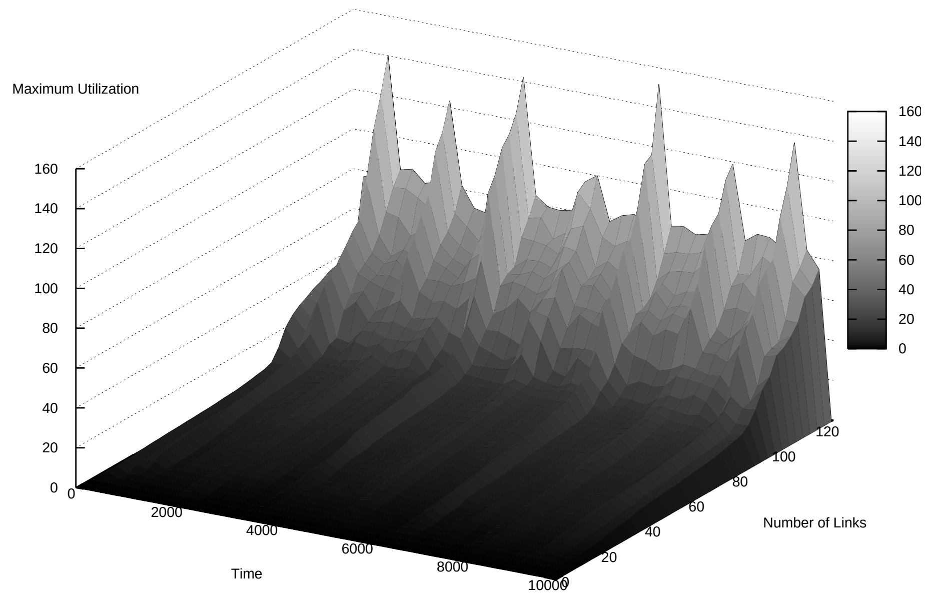


Figure 3.7: Distribution of link utilization with time for failures under IP: SYM.

Observation. Figures 3.6, 3.8, 3.7 and 3.9 show the trend of link utilization with time when traffic shifts resulting from link failures take place in MESH and SYM respectively. Sudden spikes and traffic shifts arising from failure of highly utilized links are captured. Figure 3.6 and 3.7 show several instances in time where the utilization of links crossed 100% with IP. Figure 3.8 and 3.9 show that MPLS-TE results in more links being utilized.

Hypothesis. MPLS computes paths that satisfy bandwidth constraints. This leads to more links being utilized when link failures start to take place as CSPF uses links that have available bandwidth. It should be noted that with MPLS-TE, the utilization of links mostly stay below 90%. Figures 3.6, 3.8, 3.7 and 3.9 together, clearly illustrate that traffic is better balanced with MPLS-TE where more links are at a higher utilization. This shows that traffic has been spread over the network. This is in contrast to IP where a few links are heavily congested and many are not utilized. Figure 3.10(a) and 3.10(b) highlight (in dark) the “hotspots” that arise in the network. A “hotspot” is defined to be a link that at some point experienced utilization greater than 90%. There are 19 such links for this network with IP TE in MESH and 40 such links in SYM.

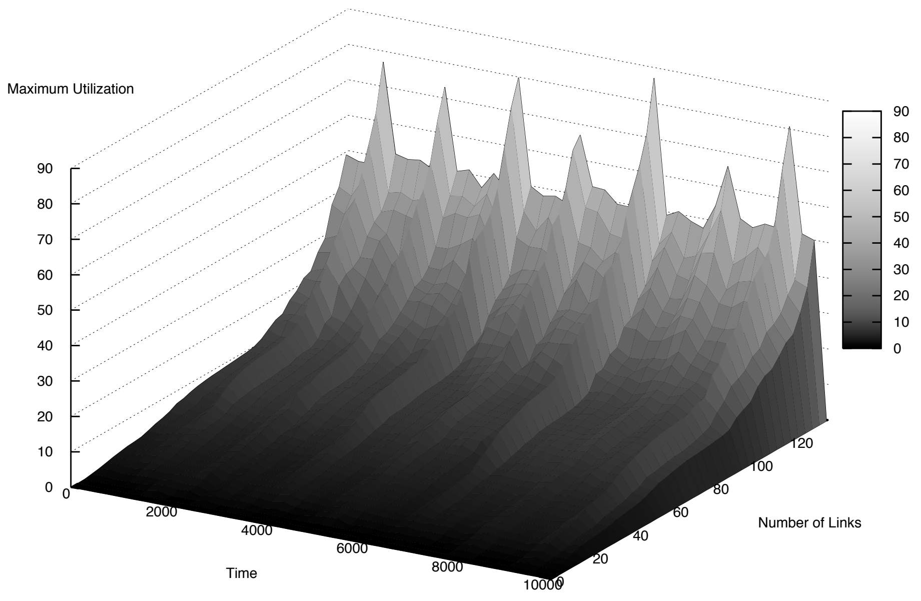


Figure 3.8: Distribution of link utilization with time for failures under MPLS: MESH.

2. **Path Cost** This section analyzes the behavior of path costs when IP and MPLS-TE reroute traffic after the occurrence of link failures. The link weight signifies some proportionality of either the link capacity or the link speed. An increase in path cost is always less desirable since it signifies either a longer delay or traversal over less desirable links.

   Observation. Figure 3.11 shows the distribution of the increase in path cost as a fraction of the IGP shortest path cost for TE-LSPs. Figure 3.12 shows

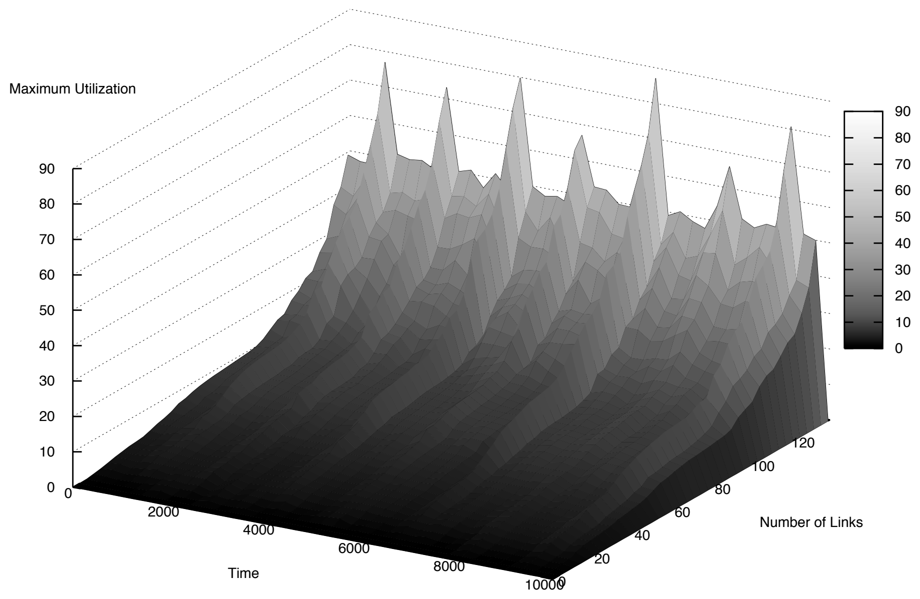
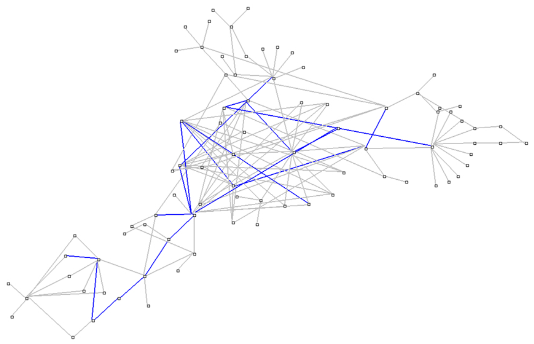
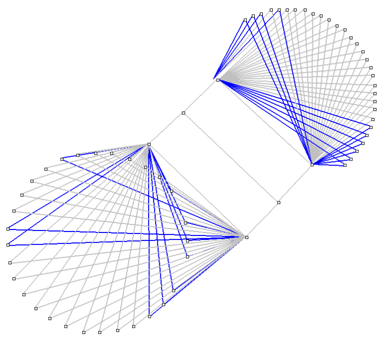


Figure 3.9: Distribution of link utilization with time for failures under MPLS: SYM.



(a) Hotspots in the network with IP and failures: MESH



(b) Hotspots in the network with IP and failures: SYM

Figure 3.10: Hotspots in the network.

the same metric for the corresponding traffic. For MESH, Figure 3.11 and 3.12 show that almost 98% of the total TE-LSPs and traffic respectively have an average path cost increase of 1.5 times the IP shortest path cost. It is observed that the TE-LSPs in SYM have a negligible average increase in path cost for TE-LSPs and traffic respectively. This behavior is attributed to the symmetric nature of the topology. While ISP1 and ISP2 have similar behavior with respect to the link utilization metric, they have a very different behavior with respect to path cost. Figure 3.11 shows that the maximum fraction of increase in path cost for a TE-LSP is 2.5 and 4 times the IGP shortest path cost in ISP1 and ISP2 respectively. Figure 3.12 shows that for ISP1, more than 96% of the traffic follow paths that have a cost similar to the IGP shortest path cost (negligible increase in fraction of IGP path cost). For ISP2, about 91% of the traffic on an average has a path cost less than 1.5 times the IGP shortest path cost. Figure 3.11 shows that more than 99% of the TE-LSPs in ISP1 have path costs similar to that of the IGP shortest path. Note that although MPLS-TE tries to ensure that traffic is rerouted along non congested paths, most of the rerouted traffic does not experience a significant path cost increase.

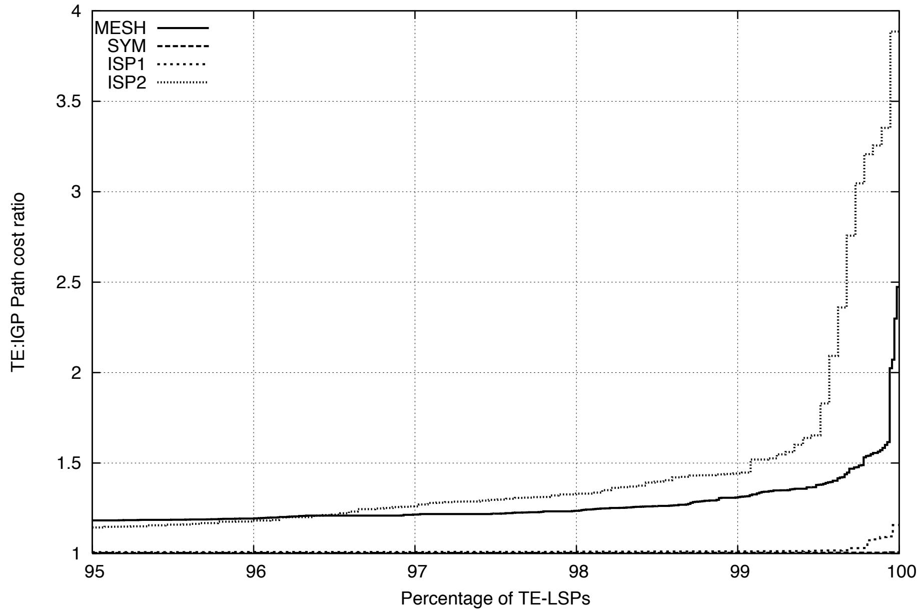


Figure 3.11: Distribution of increase in path cost for TE-LSPs.

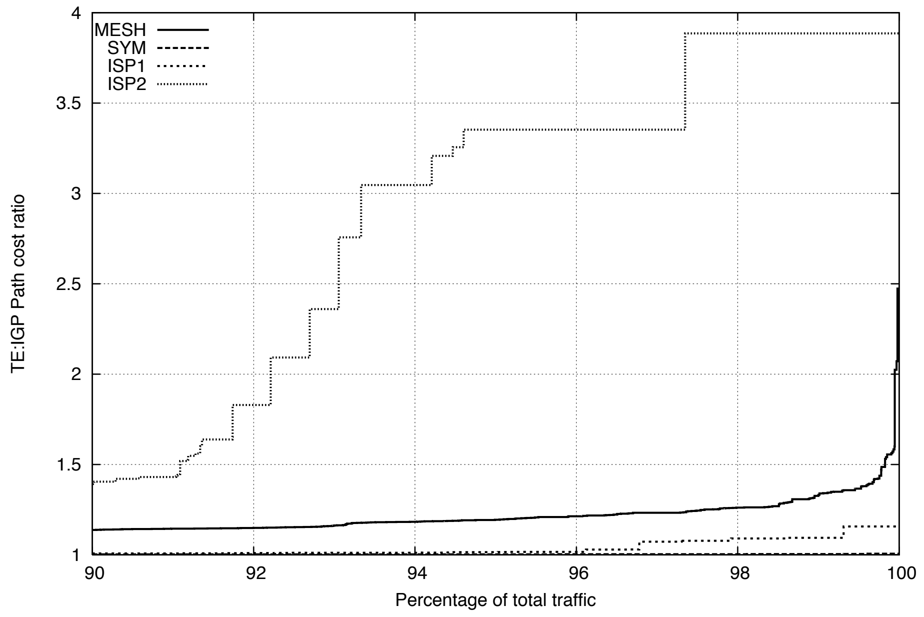


Figure 3.12: Distribution of increase in path cost for fraction of traffic.

3.4 Summary

Although Service Providers exclusively relying on a link cost optimization technique (IP TE) can effectively employ Traffic Engineering, link failures often disturb the equilibrium of the network by causing large traffic shifts and congestion. In this scenario, the common practice of changing link metrics to counter this behavior has limited efficacy and often has global undesired affects. In contrast, MPLS-TE keeps in mind the state of the topology and routes accordingly for satisfying constraints. It is demonstrated that the expense of avoiding congestion by routing on longer paths is negligible using MPLS-TE. It has been shown that MPLS-TE provides efficient techniques to avoid congestion both in steady state and under network overload conditions. Eventhough these metrics are of high interest, they had not been carefully quantified in the literature. Results from this research have been published in [19].

4. Dynamic Traffic Engineering

4.1 Introduction

Network bandwidth management is one of the prime areas of focus for service providers. Over the years, as traffic on the Internet has grown immensely, the requirement for a delicate balance between meeting QoS requirements and bandwidth provisioning has arisen. To meet perennially growing demands, dynamic bandwidth provisioning is an effective option. Though several solutions have been proposed, there are serious practical implications that are often overlooked, making a deployable solution very difficult. This chapter studies dynamic bandwidth allocation in great detail and proposes new techniques to deploy the same.

4.1.1 Motivation

The primary motivation is to design and develop and effective Dynamic Bandwidth Allocator. Before the design phase for such a mechanism begins, it is important to first understand all aspects of implementation on the underlying system so that no deployment issues arise. For this, the first goal is to understand the dynamics of the network when such a mechanism is in use. Lessons learnt from this analysis will be used in the design phase of a new mechanism. The next goal is to study the dependence of a dynamic algorithm on the traffic itself and deduce if any, dependence on the traffic profile. Ultimately, insights gained from these exercises need be kept in mind during the development of a new dynamic bandwidth allocation mechanism.

4.1.2 Related Work

Network resource allocation has been an area of study for many researchers in different fields of networking. When trying to allocate resources in a shared data center running web applications, researchers have realized the need for dynamic mechanisms, which are able to adapt to time variability of web workloads in order to provide Quality of Service (QoS) guarantees to such applications. Similarly, when dealing with multimedia streaming over the Internet, the research community has yet again converged to the need of dynamic resource allocation [21; 22; 23]. While applications and scenarios may vary, the search for QoS provisioning brings up the need for resource reservation and to take time-varying network conditions into account. Since network resources are shared, providing guarantees to applications is a complex task and are usually provided by reserving a fraction of network resources for every traffic aggregate. This fraction mostly depends on the expected traffic load and QoS requirements of the application. The workload of web applications, for instance, is known to vary dynamically over multiple time scales [24] and can be influenced by unanticipated events such as breaking news or a planned event such as the live 8 performances broadcasted around the globe, etc. While over-provisioning resources based on worst case workload estimates can result in potential underutilization of resources, under-provisioning resources can result in violation of QoS guarantees. These limitations lead to the alternate approach of allocating resources dynamically based on the variations in workload. Several research papers have investigated this proposal [21; 22; 23; 24; 25; 26; 27; 28; 29; 30; 31; 32]. While many researchers concentrate on forecasting the traffic demand, others propose a centralized controller to provide optimum allocation. In both cases, most works concentrate on the problem of optimally partitioning the traffic into classes and the link capacity into bandwidth partitions to be assigned to each class. Based on on-line measurements (or traffic forecasting), the network then combines real-time traffic and network state information in order to optimize the size of the bandwidth partitions or the service rate of the traffic classes to achieve better performance [32]. In Section 4.2 we investigate the problem of dynamic resizing after traffic classes have been already decided on; resources (reserved bandwidth) have been reserved in the network for each class and a corresponding route has been found in the network for the incoming traffic. In Section 4.3 augments the Dynamic TE mechanism with resizing and rerouting policies to customize the mechanism according to needs. Section 4.4 evaluates the performance of Dynamic TE on a mix of voice and data traffic in an international traffic scenario. The concept of overlapping and non-overlapping traffic periods arising due to timezones has been previously introduced in [33] and [34].

Dynamic bandwidth reservation has been an area of research for all. In the context of MPLS-TE, research undertaken in this area is compared to [35] and [36]. In [35], the authors present an ARCH-based traffic forecasting and dynamic bandwidth provisioning mechanism. The first phase of the mechanism involves ‘model identification,’ where ARCH model fitting for measured data is undertaken and results in the estimation of model parameters. In the forecast phase, the Maximum Mean Square Error (MMSE) of the traffic is calculated from the reserved bandwidth. This is then augmented with the forecast of a probability of increase or decrease of traffic based on the previous two sampling periods. Together, the MMSE and the probability forecast result in the control of reservation. The filter based prediction approach presented in [36] attacks the problem differently. It relies on a filtering mechanism that estimates the number of active connections based on periodic measurements. Prediction is then undertaken to gauge future connections and correspondingly change the reservation. It is also augmented with an ‘emergency procedure’ that reacts to sudden unexpected spikes only. Though interesting, these methods lack in several areas that are discussed and compared to in Section 4.5.

4.1.3 Contribution

In Section 4.2, we present Dynamic Traffic Engineering, a bandwidth reservation mechanism coupled with path computation to find the shortest path corresponding to the changing reservation. Realistic simulations are undertaken to analyze the dynamics of the network when such a mechanism is deployed in the network. Findings are published in [37].

Section 4.3 and Section 4.4 discusses the augmentation of Dynamic TE with resizing and rerouting policies and analyzes the performance of Dynamic TE on mixed traffic and international network scenarios. Findings are published in [38].

Section 4.5 presents a novel bandwidth resizing mechanism that keeps in mind the lessons learnt from Dynamic TE and policies. It analyzes signaling issues in great detail and compares performance with two similar works. Findings are published in [39].

4.2 Dynamic Traffic Engineering

To start the investigation, we propose the Dynamic Traffic Engineering Mechanism (Dyn-TE). It combines the dynamic resizing of MPLS Tunnel bandwidth with path computation to find a path the shortest path corresponding to the new reservation. Variations and modifications to this sequence is introduced as more insight is gained into network behavior and performance. The following system parameters are defined initially, tunnel reserved bandwidth, traffic sampling period, set of traffic samples and tunnel resizing period.

Tunnel reserved bandwidth. MPLS Tunnels that are setup in the network are associated with a peak reservation value denoted as λ_i^{max} for Tunnel i. When dynamic reservations are taking place, the reservation associated with TE-LSP corresponding to tunnel i at time t is denoted as λ_i^t. Also, reserved bandwidth of TE-LSP i in resizing period r is denoted as $\lambda_i(r)$.

Traffic sampling period. The head-end LSR can sample the traffic that is being forwarded into the corresponding tunnels. The sampling period is configurable and is denoted as t_s.

Tunnel resizing period. The tunnel resizing period is the period of time after which the head-end resizes the reservation associated with the tunnel based on the samples collected. It is configurable and is denoted as t_r.

Set of traffic samples. The set of samples of TE-LSP i collected during resizing period r is denoted as $\mathcal{X}_i^r = (x_i^{r,1}, x_i^{r,2} \ldots x_i^{n,r})$ where $x_i^{n,r}$ denotes n^{th} sample collected in resizing period r of TE-LSP i.

Formally, Dynamic TE is denoted as solution of two problems:

$$\begin{cases} \lambda_k(r+1) = RESERVATION(\mathcal{X}_i^r) & \text{New reservation} \\ \boldsymbol{p_k} = COMPUTEPATH(\lambda_k(r+1)) & \text{New path} \end{cases}$$

Here, an instance for $RESERVATION$ is the set of samples taken in a resizing period and the solution is a new bandwidth reservation for the tunnel. An instance for $COMPUTEPATH$ is a network topology, source, destination and a constraint. The solution is a sequence of links that form the path from the source to the destination and guarantees that the constraint is met on each link. Dynamic TE in its simplest form involves resizing the tunnel reservation to the maximum sample in a resizing period followed by finding the shortest path to fit the reservation.

Algorithm 1 $RESERVATION(\mathcal{X}_i^r)$

```
Require: $r > 0$
1: $\lambda_i(r+1) = MAX(\mathcal{X}_i^r))$
2: return $\lambda_i(r+1)$
```

Algorithm 2 $COMPUTEPATH(u, v, \mathcal{D}^r, \lambda_k(r+1))$

```
1: $w_l^{temp} = 0 \vee a_l^r \leq \lambda_k(r+1)$
2: $\mathcal{D}_r^{temp} = (\mathcal{L}_r^{temp})$
3: $\boldsymbol{p_k} = DIJKSTRA(\mathcal{D}_r^{temp})$
4: $\delta_k = w(\boldsymbol{p_k})$
5: return $\boldsymbol{p_k}$
```

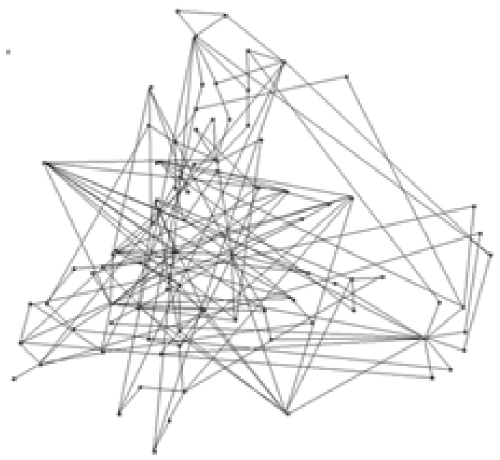
4.2.1 Simulation Results

Four cases are studied, Dynamic TE and Static TE with and without link failures (DYN, F-DYN, STAT and F-STAT respectively). Traffic is then scaled to observe the ability of Dynamic TE to fit more traffic.

Simulation Setup

The Dynamic TE mechanism simulated resizes reservation every 60 minutes ($t_r = 60$) and samples traffic every 5 minutes ($t_s = 5$). The Static TE mechanism maintains peak reservation ($\lambda_k^t = \lambda_k^{max} \forall k \in \mathcal{K}$) throughout the duration of the simulation.

Two realistic yet structurally different topologies are selected. The motivation for using two distinctly different topologies is to observe the performance dependence on topology. MESH as shown in Figure 4.1(a) represents a service provider core and is a highly connected network having numerous possible paths for a source destination pair. In networks like these, traffic setup failures seldom occur when links fail because of its high degree of connectivity. In contrast, the SYM topology as shown in Figure 4.1(b) is a very ‘constrained’ network in terms of connectivity. It represents a scenario where TE is deployed in an Edge to Edge fashion with a small core in between. Number of paths for a source destination pair are limited thereby increasing the possibility of path setup failures. SYM has been obtained from the RocketFuel trace files [40] whereas SYM is obtained from [41].


(a) Topology for MESH.

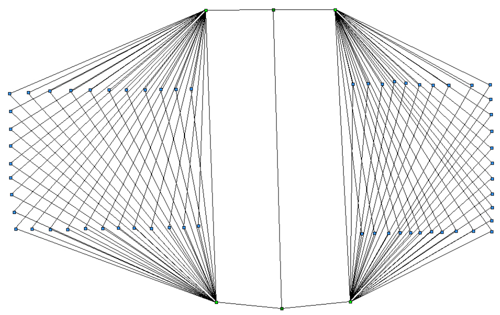

(b) Topology for SYM.

Figure 4.1: Topologies used for performance evaluation.

Details of MESH and SYM are tabulated in Table 4.1. Table 4.2 shows how the TE-LSP sizes are distributed in the network. The same traffic matrix is used for both the networks. The traffic profile is similar to that in Figure 3.3.

Table 4.1: Topology details for MESH and SYM

Type	Network Description						
	Net name	# of nodes	# of links	OC3 links	OC48 links	OC192 links	1 Gbps
Core	MESH	83	167	0	132	35	0
Edge	SYM	66	127	40	0	7	80

Table 4.2: TE-LSP size distribution

	TE-LSP Description				
Type	Net	#	0Kb–1Mb	1Mb–20Mb	20Mb–50Mb
Core	MESH	6806	70%	25%	5%
Edge	SYM	3540	70%	25%	5%

Performance Analysis

This section presents performance analysis of Dynamic TE using some of the metrics discussed in Chapter 2, Section 2.2.

1. **Number of TE-LSPs Not on Their Shortest Path**

   Figure 4.2 shows the behavior TE-LSP paths with respect to time in Network 1 with and without failures for Static and Dynamic TE methods and a scale factor of 1.0.

   Observation. Static TE remains constant in the absence of link failures and shows no particular behavior in the presence of link failures.

   Hypothesis. In the case of Static TE without failures (STAT), the TE-LSPs are setup in the network with their maximum peak and never change in path or reservation. Thus, the number of TE-LSPs not on their shortest IGP path never changes, showing up as a straight line. When link failure events take place with Static TE (F-STAT), the TE-LSPs initial setup size is the same as before. As the network undergoes link failures, TE-LSPs are rerouted. The failed TE-LSPs are now rerouted on routes that can be longer or shorter than its current route, depending on the network state. This behavior is illustrated in Figure 4.2.

   Observation. This metric shows a periodic trend for Dynamic TE in the ab-

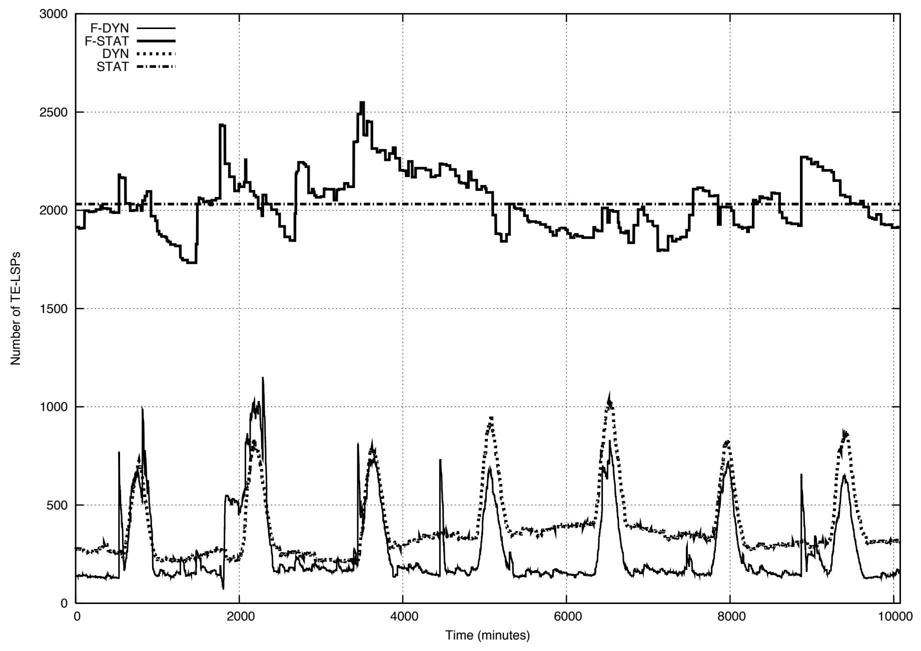


Figure 4.2: TE-LSPs not on the shortest path. MESH, Scale Factor=1.0.

sence and presence of link failures (DYN and F-DYN)

Hypothesis. When Dynamic TE is used and there are no link failures (DYN), resizing changes the reserved bandwidth of the TE-LSP according to the traffic profile. This causes more TE-LSPs to be routed through shorter paths. When the amount of traffic is low, reservation is reduced accordingly freeing up bandwidth on links. This freed bandwidth is used by other TE-LSPs allowing more to be on their shortest path. The opposite happens when the amount of traffic is large. Reservations increase, causing TE-LSP routes to be longer and more spread over the network. Thus, the number of TE-LSPs not on their shortest paths increases daily during the peak period of traffic and reduces during the off-peak periods. In the presence of link failures, there is always more room available on other paths to reroute the TE-LSPs that were traversing the failed link. Although in this case a reroute might cause the new path of a failed TE-

LSP to be longer than its IGP path, much fewer actually are as compared to the static peak based scenario with failures. The affect of link failures can be mitigated by the deployment of Dynamic TE. This fact should certainly be stressed since the use of Dynamic TE removes one of the challenges of constrained-based routing, which relates to the potential inability to find a path especially after failures and/or lead to having more TE-LSPs following longer routes. Constraint relaxation approaches are usually required which increases the network management complexity. It can be seen on Figure 4.2 that the use of Dynamic TE greatly helps in that respect since the number of TE LSPs that do not follow the shortest paths after failure is greatly reduced as compared to the static approach. Its worth mentioning that in this case, the number of TE-LSPs not on the shortest path seem to be smaller than DYN and is a result of several TE-LSPs failing to find a route on failure, freeing up additional bandwidth that allows TE-LSPs that are still up to route through the less loaded links.

Observation. When scale factor is increased, the trend for this metric remains similar but the number of TE-LSPs not on their shortest path increases.

Hypothesis. Figure 4.3 shows a similar behavior as Figure 4.2 (same metric), however, here we see that STAT always has a higher number of TE-LSPs not on their shortest path, even when compared to F-STAT. Since the scaling factor is now k = 3.0, several TE-LSPs (570 from Table 4.3) were not setup (no path was found that could fit them) in the first place, and several more were down after failure (1210 from Table 4.3). The network is therefore “emptier after failure (but each TE-LSP has double the volume of traffic as before) thus explaining the higher number of TE-LSPs on their shortest path. DYN and F-DYN behave similar to Figure 4.2 (0 LSPs fail to be setup in DYNs case, since they are initially setup with 0 bandwidth reservation). A smaller amount

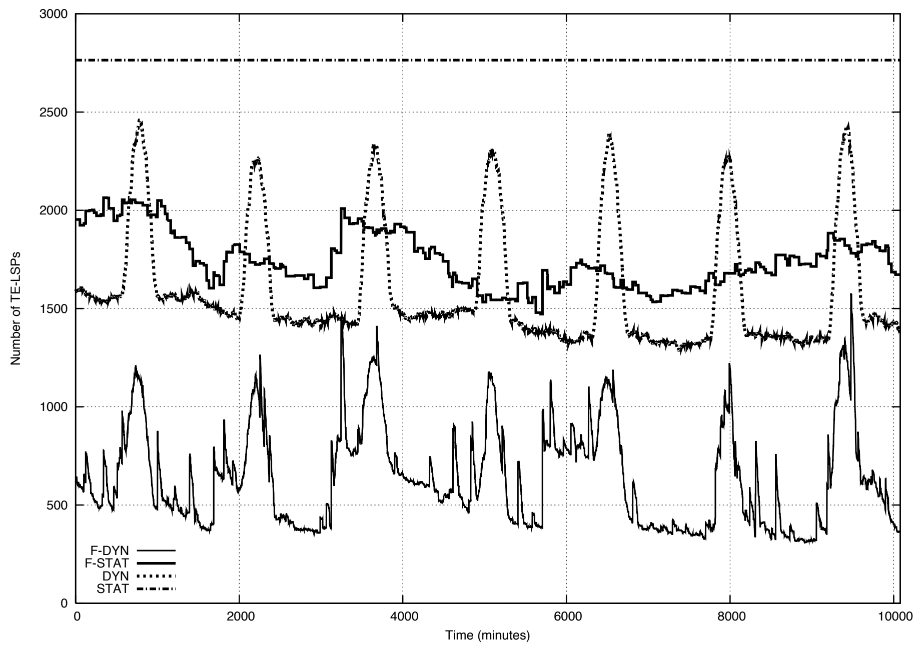


Figure 4.3: TE-LSPs not on the shortest path. MESH, Scale Factor =3.0.

was down after failure (441 from Table 4.3), since no path was found in which those TE-LSPs could be setup. This aspect is a fundamental observation when comparing the two approaches where it can be seen that Dynamic TE clearly allow a service provider to carry more traffic while performing constrained based routing. This is even truer in the case of link failure.

2. **Maximum Reserved Bandwidth on Links**

Observation. The STAT case always results in every link having a non-varying maximum utilization, since there are no failures or rerouting. Figure 4.4 shows the distribution of maximum reserved bandwidth experienced by the links. About 10% of the links are at 100% utilization and stay so for the duration of the simulation for DYN and STAT.

Table 4.3: TE-LSP setup failure for MESH and SYM

MESH				
Traffic Scale Factor	Static Peak Based TE		Dynamic TE	
	Failed to setup	Down on link failure	Failed to setup	Down on link failure
$k = 1.0$	0	0	0	0
$k = 2.0$	570	1210	0	441
$k = 3.0$	1645	2510	0	1515
SYM				
Traffic Scale Factor	Static Peak Based TE		Dynamic TE	
	Failed to setup	Down on link failure	Failed to setup	Down on link failure
$k = 1.0$	0	655	0	560
$k = 2.0$	380	1055	0	1125
$k = 3.0$	660	1341	0	1391

Hypothesis. With Dynamic TE (DYN), the maximum link utilization is similar to STAT since the maximum size of the TE-LSPs can be at most their maximum peak, which is always the reservation regime for Static TE (STAT).

Observation. Links experience more reserved bandwidth with F-STAT and F-DYN.

Hypothesis. For F-STAT, links that were already utilized to a certain degree now accommodate the newly rerouted TE-LSPs too. This increases the maximum reservation as seen in Figure 4.4 (every link is more utilized than in STAT case, and almost 20% are at 100% utilization). For F-DYN, when TE-LSPs are rerouted due to a link failure, the amount of reservation required is not always equal to the maximum peak. It will depend on the current traffic volume on that TE-LSP. Since there is an increase in utilization of the links onto which the failed TE-LSPs are rerouted, the maximum utilization a link experiences increases, as shown in Figure 4.4. This increase is not, however, as much as in the case for Static TE undergoing failure (F-STAT).

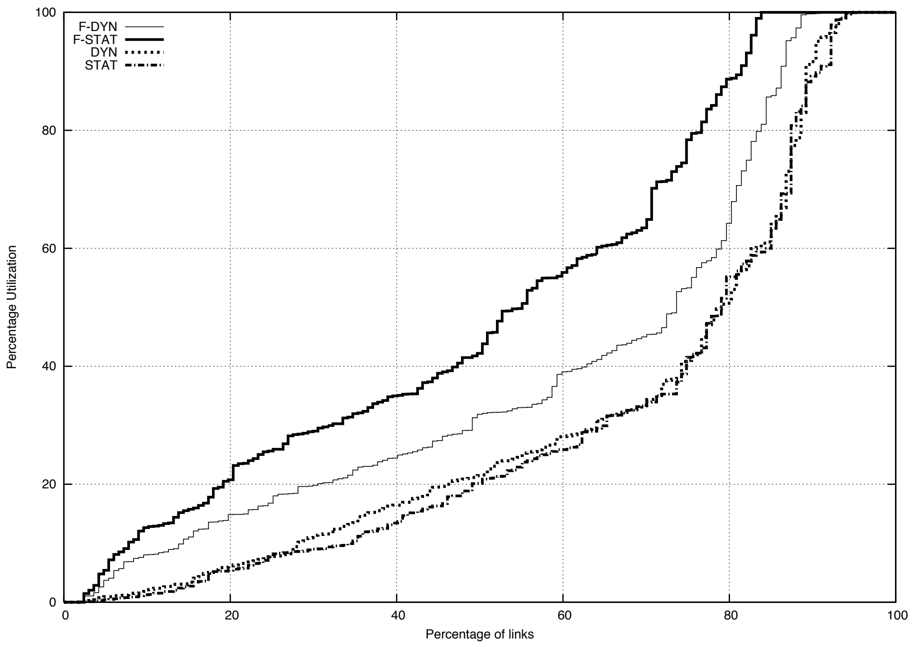


Figure 4.4: Distribution of maximum reserved bandwidth. MESH, Scale Factor=1.0.

Observation. For Scale Factor=3.0, maximum link reservation is more for DYN than STAT.

Hypothesis. Figure 4.5 shows this behavior. On MESH, with STAT, Table 4.3 shows the setup failure of 4155 TE-LSPs as compared to the setup failure of 1515 TE-LSPs for DYN. The extra TE-LSPs with DYN are routed over many more links thus increasing the maximum reserved bandwidth experienced by the links.

Observation. Figure 4.6 shows the distribution of maximum reserved bandwidth for SYM. The point to note is the difference in trend with MESH. Since the network is constrained structurally, the network core always experiences more reserved bandwidth than the rest of the network.

3. **Maximum TE-LSP Setup Requests at a Router**

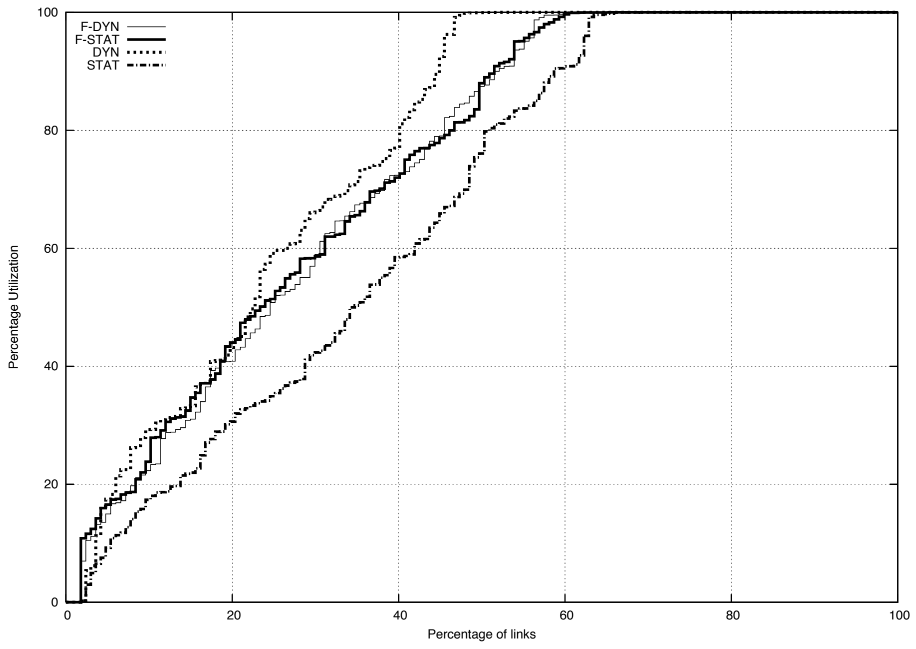


Figure 4.5: Distribution of maximum reserved bandwidth. MESH, Scale Factor=3.0.

Observation. The maximum number of TE-LSP setup requests received at a router is more for Dynamic TE than for Static TE in both the presence and absence of link failures.

Hypothesis. Figure 4.7 shows the number of TE-LSP setup request received by a node after the initial setup of all TE-LSPs at the start of the simulation. In the STAT case, no TE-LSP setup request occurs (TE-LSPs are never rerouted since failures do not occur). With DYN, nodes receive setup requests only when a better route has been found for a TE-LSP for its new size and also when they are resized to a higher value. With F-STAT, nodes receive setup requests only when TE-LSPs need to be rerouted on the event of a failure. Finally, for F-DYN, nodes receive setup requests on optimization of routes and on the rerouting of

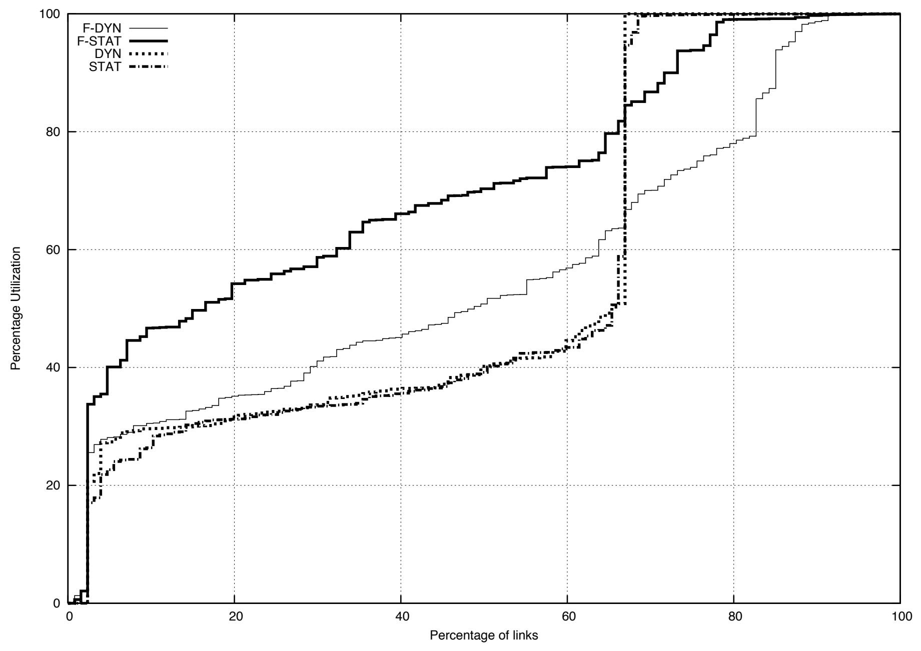


Figure 4.6: Distribution of maximum reserved bandwidth. SYM, Scale Factor=3.0.

disrupted TE-LSPs on a link failure resulting in the higher value shown in the figure.

4. **Maximum TE-LSPs Traversing a Router**

   This metric shows the behavior of the network in terms of the maximum number of TE-LSPs that are passing through a node. The motivation as mentioned previously was to study the worst case occurring.

   Observation. With Dynamic TE, routers have more TE-LSPs traversing through them effectively increasing router load when compared with Static TE. This is true in the absence and presence of link failures. For an increase traffic scale factor, this behavior is more pronounced.

   Hypothesis. Figure 4.8 shows that for Static TE (STAT), the available bandwidth on the links is consumed rapidly (since every TE-LSP is setup with its

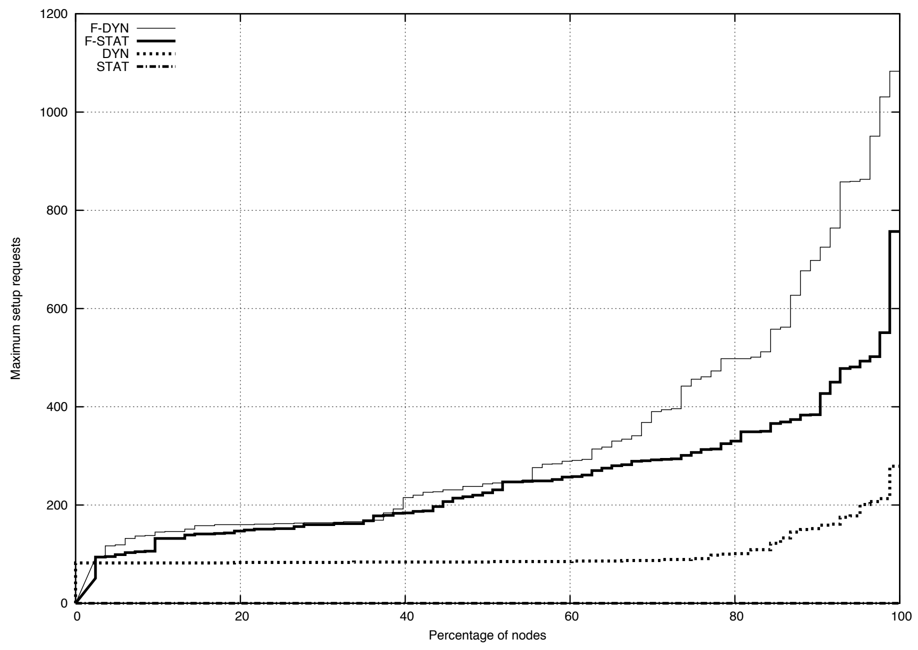


Figure 4.7: Maximum setup requests received distribution. MESH, Scale Factor=1.0.

peak rate). The TE-LSP routes thus start to lengthen and spread over the network. This results in a reduction of the number of TE-LSPs through a node. Since no failures occur in this case, TE-LSPs stay on their assigned routes keeping the maximum number of TE-LSPs passing through a node in the network constant.

When failure is added to Static TE (F-STAT), TE-LSPs that need to be routed either spread more over the network or converge over a certain common set of links that have available bandwidth due to previous failures and restorations. If the TE-LSPs spread, this further reduces the maximum number of TE-LSPs the network has passing through one of its nodes. If however the TE-LSPs converge to a common section of the network, the maximum number of TE-LSPs on a node in the network increases. It is important to make note of the fact that the information conveyed by Figure 4.2 is inversely related to that conveyed by

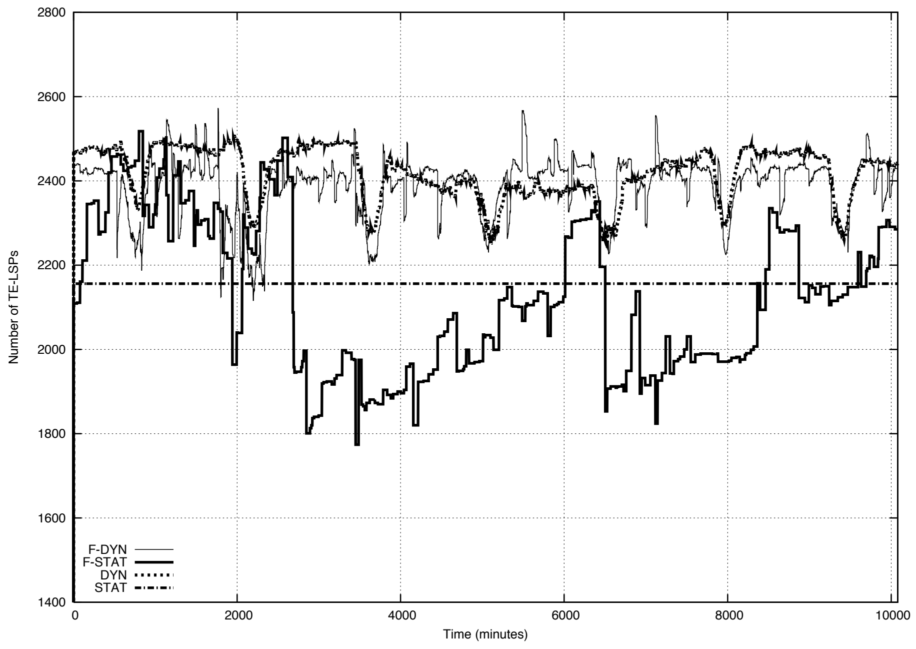


Figure 4.8: Maximum TE-LSPs traversing a router. MESH, Scale Factor=1.0.

Figure 4.8. As more and more TE-LSPs start to follow their shortest path, the maximum number of TE-LSPs passing through a node increases.

For Dynamic TE without failures (DYN), we also observe a periodic behavior of the maximum number of TE-LSPs that pass through a node. The number decreases periodically when traffic load is high and then increases to its original value. The reason for that is that during the peak period of traffic, more TE-LSPs are routed along longer routes having a less loaded set of links due to the increase in their reservation. This spreads the TE-LSPs over the network reducing the maximum number of TE-LSPs passing through a node.

Finally, for the F-DYN case, when TE-LSPs are rerouted due to a failure, the relative spread over the network is much less compared to the Static TE case, as seen in Figure 4.8. This does not result in a significant reduction of the maximum number of TE-LSPs passing through a node. This is a result of the inherent characteristic of Dynamic TE, which tries to keep the set of TE-LSPs spread only over a relatively smaller part of the network by periodically resizing and also permitting the traffic to follow a better (shorter) path.

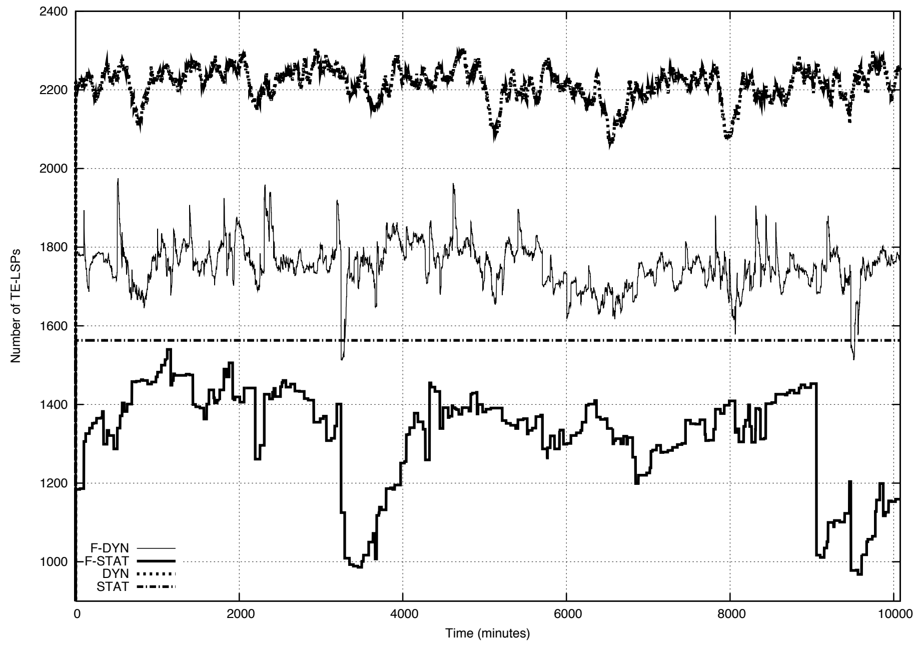


Figure 4.9: Maximum TE-LSPs traversing a router. MESH, Scale Factor=3.0.

Figure 4.9 shows that the maximum number of TE-LSPs passing a node is lower for F-STAT due to the high number of TE-LSPs not setup or down after failure. Also, since the size of the TE-LSPs is now much larger, they spread around the network leading to lesser of them traversing a router.

4.2.2 Summary

In this section we studied the benefits of a decentralized dynamic resource reservation mechanisms: Dynamic TE. Several metrics of interest were described and compared through simulation results. With Dynamic TE, most TE-LSPs are routed through their shortest path leading to a small total path cost. The traffic matrix need not be known a priori (as in IP TE or Static TE cases), since Dynamic TE takes measurements of the current traffic (a great benefit, since it is difficult for service providers to have this data.). Moreover, a service provider using Dynamic TE is able to fit more traffic through the network, since the reservations are a reflection of the actual traffic rate at the moment and not its peak. Finally, Dynamic TE also helps mitigate failures, since more TE-LSPs are able to find a new route after failure, due to the fact that the TE-LSPs are setup with a reservation that reflects their actual traffic volume rather than the traffic peak.

4.3 Provisioning and Rerouting Policies

Based on the insights obtained from the initial performance analysis of Dynamic TE, policies to control resizing and rerouting of tunnels are introduced. From the perspective of service providers, the motivation of using policies is to provide the ability to modify and customize Dynamic Traffic Engineering based on resource and topological constraints. Numerous variations can be implemented to achieve better utilization in terms of reserved bandwidth and actual traffic associated with a TE-LSP. These methods could utilize characteristics of the TE-LSP such as route costs and traffic profiles in addition to various threshold values to base their decision on for resizing. This section discusses various policies that can be coupled with the Dynamic TE mechanism to address provider specific requirements. One of the long term goals is to also incorporate lessons learnt from Dynamic TE policies during the development of a new bandwidth resizing mechanism.

4.3.1 Peak, Average and Weighted Sample Based Mechanisms

Dynamic resizing involves periodic sampling and resizing. The overall change in utilization varies on how the samples collected are used. Various possibilities could be taken into consideration dependent on computation and sampling complexity. In this paper, weighed samples, average of all samples and the maximum sample over a sampling period are studied as options for the next resizing.

The set of samples of TE-LSP i collected during resizing period r is denoted as $\mathcal{X}_i^r = (x_i^{r,1}, x_i^{r,2} \ldots x_i^{n,r})$ where $x_i^{n,r}$ denotes n^{th} sample collected in resizing period r of TE-LSP i.

As defined in Section 4.2, $\mathcal{X}_k^r = (x_k^{r,1}, x_k^{r,2} \ldots x_k^{n,r})$ where $x_k^{n,r}$ denotes n^{th} sample collected in resizing period r of TE-LSP k.
For a Peak Sample based mechanism, TE-LSP k will be resized to

$$\lambda_k(r+1) = \max(\mathcal{X}_k^r) \tag{4.1}$$

For an Average Sample based mechanism, the TE-LSP k will be resized to a value defined as

$$\lambda_k(r+1) = \text{avg}(\mathcal{X}_k^r) \tag{4.2}$$

For a Weighted Sample based mechanism, the TE-LSP k will be resized to

$$\lambda_k(r+1) = \frac{(c_1.x_k^{r,1} + c_2.x_k^{r,2} + \cdots + c_n.x_k^{n,r})}{(c_1 + c_2 + \cdots + c_3)} \tag{4.3}$$

where $c_1, c_2, \ldots, c_n$ are the set of weights chosen.

4.3.2 Conditional Resizing

Figure 4.10 shows the sequence of steps involved in Dynamic TE. After a new size has been determined, changing the reservation involves three stages, firstly, computing a new shortest route based on the new size, secondly, reserving bandwidth on the new route and, finally, tearing down the old TE-LSP. The first two stages incur significant processing and signaling overhead if undertaken frequently. A conditional resizing policy can be used to prevent TE-LSPs from re-routing on every resize by bounding the total path cost of the new path. A path cost higher than this bound will not allow a TE-LSP to resize or reroute. This will also ensure the TE-LSPs are on shorter paths. The condition can be expressed as

$$\text{Cost(New Path)} \leq \beta \cdot \text{Cost(Current Path)}, \tag{4.4}$$

where the cost functions Cost($\boldsymbol{p_k}$) is the total path cost of path $\boldsymbol{p_k}$ of TE-LSP k and β is the the factor which bounds the path cost. Similarly, frequent resizing can be prevented by conditionally resizing when there is a increase or decrease in traffic by a certain fraction of the maximum or current demand. This can be expressed as

$$\lambda_k(r+1) - \lambda_k(r) \leq \gamma \cdot \lambda_k^{max}, \tag{4.5}$$

$$\lambda_k(r+1) - \lambda_k(r) \leq \gamma \cdot \lambda_k(r), \tag{4.6}$$

where γ is the factor that governs the fraction of difference that controls the resizing.

It should be noted that all the above policies can be used in combination and also selectively across tunnels based on their characteristics. The performance of each policy and their use will be analyzed in detail in the next section.

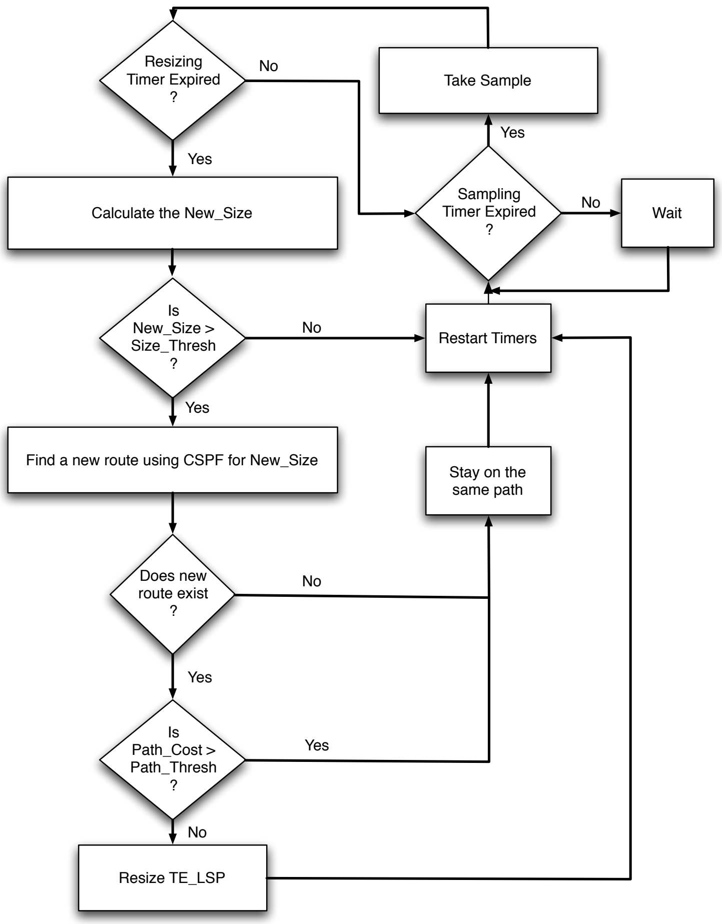


Figure 4.10: Resizing policies with dynamic TE.

4.3.3 Simulation Results

Simulations are undertaken on SYM. The sampling period is $t_s = 5$ minutes and the resizing period is $t_r = 60$ minutes.

Performance Analysis

1. **Sample usage**

   Three ways of how the sample set could be used have been studied, using only the maximum of samples, weighted average of the samples and simple average of samples. Sample information can be used in a plethora of ways each having its own motivation, this discussion presents the influence of basic sample usage on QoS. An aim is also to reduce the processing on a router since potentially, this mechanism could be running for tens of thousands tunnels. For weighted average, constants $c_1, c_2 \ldots, c_n$ from Equation 4.3 are assigned the index of the sample in the resizing period.

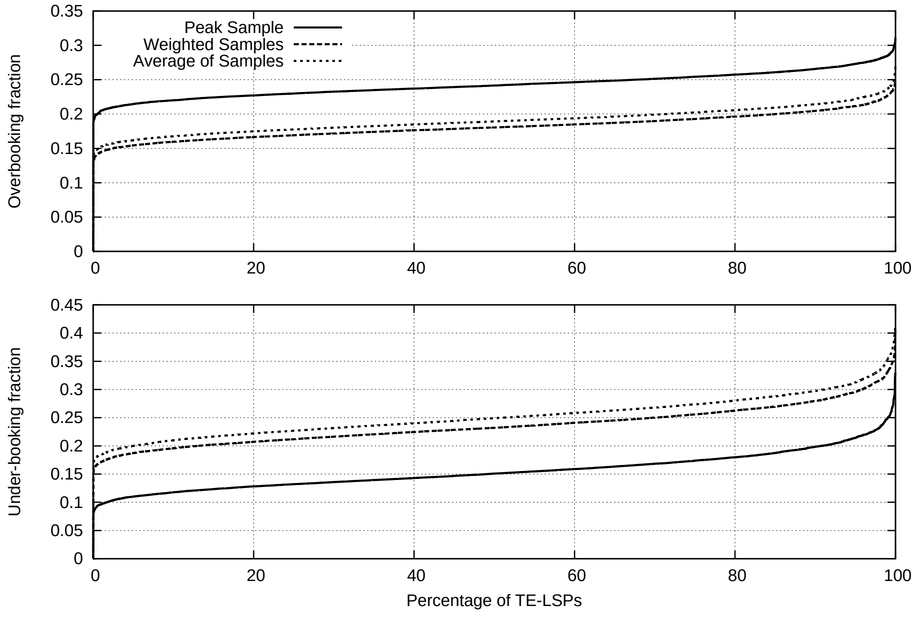


Figure 4.11: Booking distribution based on sample usage.

Observation. Except overbooking and under-booking fractions, all metrics have similar performance. Using the maximum of samples in a resizing period causes more overbooking and much less under-booking on average as compared to the weighted-average and simple average methods.

Hypothesis. Figure 4.11 shows this behavior.

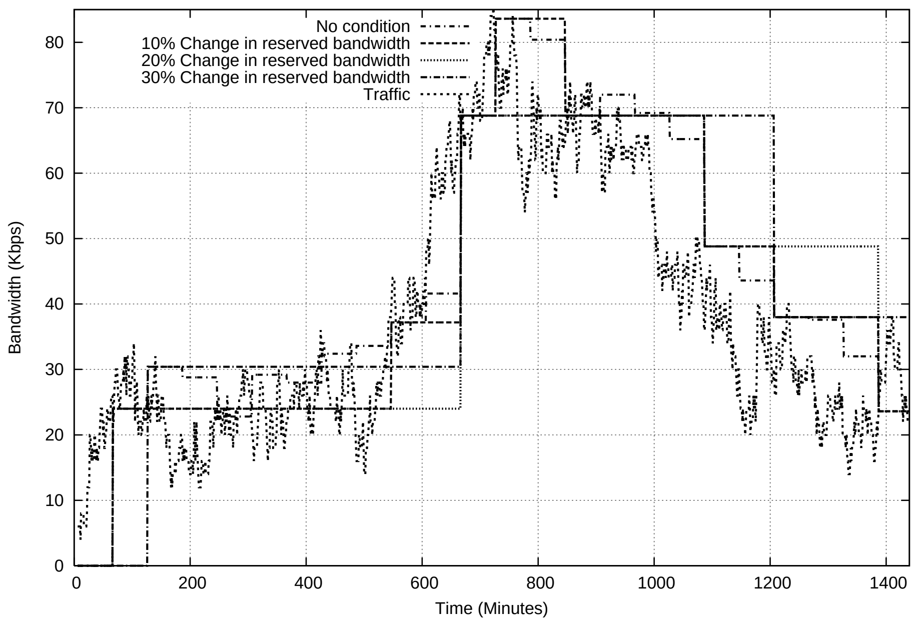


Figure 4.12: Variation in resizing based on bandwidth change conditions.

2. **Conditional Resizing Based on Bandwidth Change Bounds**

   Resizing takes place only when the projected bandwidth requirement for the tunnel differs from the current size by a certain threshold. For initial analysis purposes this threshold is taken to be a fraction of the maximum tunnel size ($\lambda_k^{max}, k \in \mathcal{K}$). The motivation is to reduce resizing and associated signaling. The tradeoff is the amount of over and under-booking that takes place.

   Observation. Increasing the threshold for bandwidth change bound reduces the the amount of resizing for a tunnel and keeps the corresponding TE-LSP on the same path more often.

   Hypothesis. Figure 4.12 shows the behavior of reserved bandwidth for a tunnel with conditional resizing using bandwidth change bounds. Table 4.4 supports the motivation for employing a bandwidth change bound. It is seen that a small bound of $\beta = 0.1$ drastically reduces the number of resizes and still keeps the

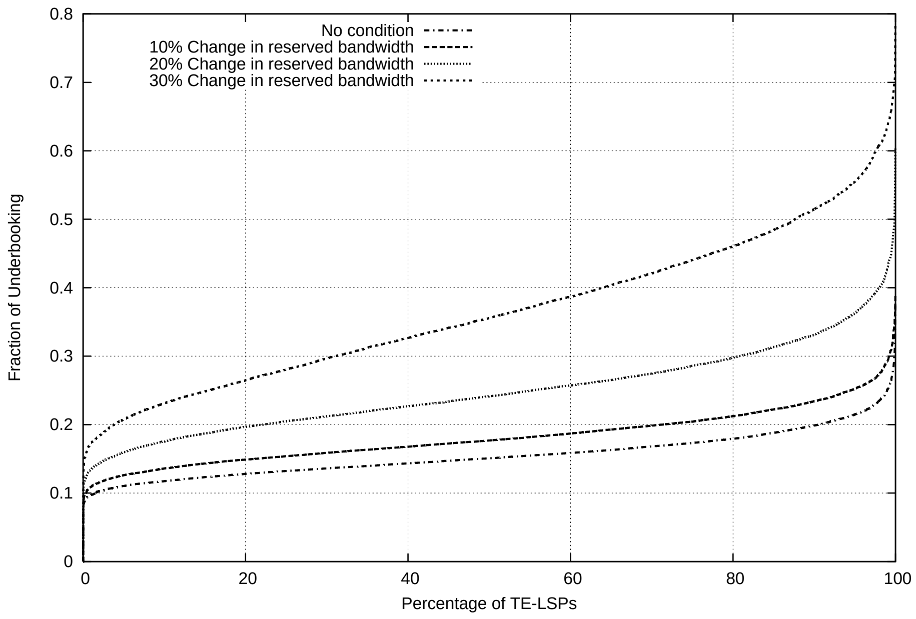


Figure 4.13: Underbooking distribution based on bandwidth change conditions.

fraction of under-booking low as shown in Figure 4.13. It can be seen that higher bandwidth change bounds can cause more of undesirable under-booking, potentially leading to congestion. Also as the bound is increased, there are more chances that the TE-LSP is resized on the same path thereby reducing jitter. This happens especially during downsizing when the same path has the capacity to fit the TE-LSP with lower size.

Table 4.4: Results for reserved bandwidth bounds

Threshold Fraction (y)	Number of Resizes	Fraction of times on the same path
No bound	23	0.82
0.1	7.5	0.84
0.2	3.6	0.95
0.3	2.8	0.99

3. **Conditional Resizing Based on Path Cost Bounds**

   In this policy, resizing is controlled by bounds on path costs. The motivation is to prevent traffic from following long paths.

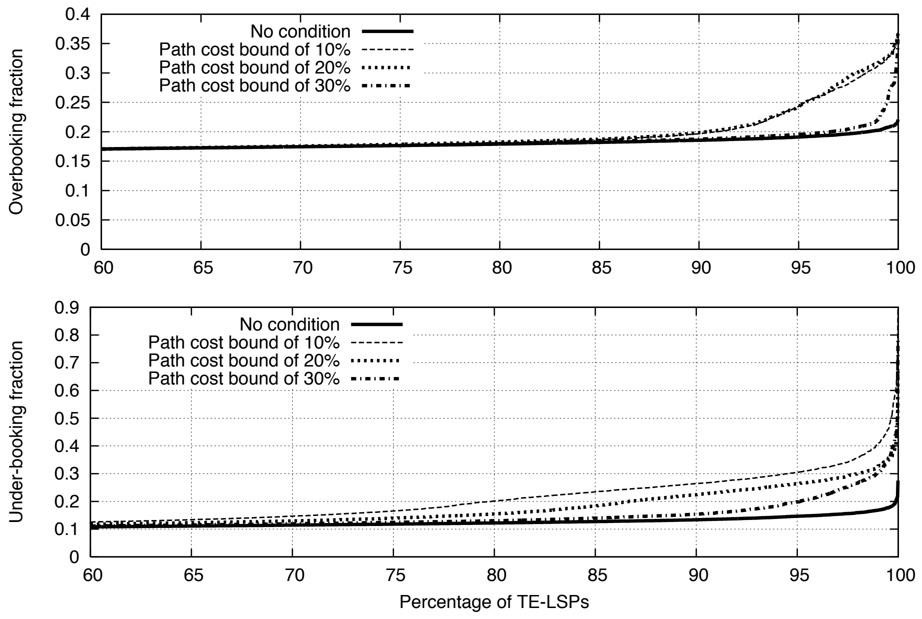


Figure 4.14: Booking distribution based on path cost bounds.

Observation. Reducing the path cost bound leads to more under-booking but keeps the increase in path cost low.

Hypothesis. Figure 4.14 shows that reducing path cost bounds leads to under-booking. This is because TE-LSPs are not allowed to resize to a higher reservation value when their new path cost is greater than the bound. Similarly, overbooking is caused when a TE-LSP cannot downsize because of the cost bound. It should be noted that the performance of this mechanism is topology dependent as a network with high degree of connectivity will lead to more routes meeting the path cost bound. It should also be noted that preventing resizing does not prevent traffic flow into the tunnel and might lead to congestion.

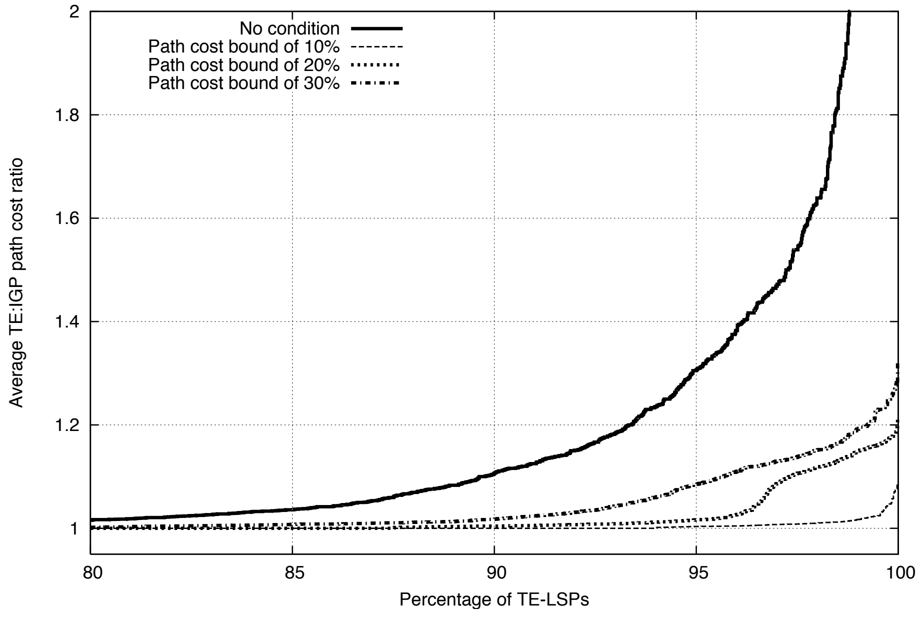


Figure 4.15: Path cost increase distribution.

Figure 4.15 shows the distribution of the fraction of increase in cost. With low bounds, it can be seen that on an average, more TE-LSPs are on shorter paths, thus meeting the motivation behind such a policy. Table 4.5 also shows that as expected, more resizing takes place when the bounds are higher. Also, for lower value of bounds, resizing mostly takes place on the current path, thus leading to a higher fraction of the time a TE-LSP follows the same path.

4.3.4 Summary

The introduction of policies into Dynamic TE brought forward several interesting results. We observed that by using simple threshold based tuning parameters, Dy-

Table 4.5: Results for path cost bounds

Threshold Fraction (x)	Number of Resizes	Fraction of times on the same path
No bound	23	0.82
1.1	16.1	0.92
1.2	17.8	0.83
1.3	20.8	0.81

namic TE can be customized to perform according to specific requirements. These variations in the working of Dynamic TE has also given insight into how various overheads such as signaling and processing can be reduced. Several important lessons are learnt and can be used in the design phase of new bandwidth reservation mechanisms.

4.4 Effect of Traffic Profile Characteristics

Networks have evolved to carry worldwide traffic for diverse kinds of applications such as voice, data, video gaming etc. Traffic loads for international carriers have many interesting characteristics arising out of inherent properties of these applications. Medium to large enterprises with sites in multiple countries prefer to rely on a single global provider for their telecommunication needs rather than dealing with multiple national and international carriers. Here, voice and data traffic are carried across several timezones and have varied characteristics for number of peak periods and average-to-peak traffic ratio. Discussions on and sample traffic profiles can be found in [33], [34], [41], and [42].

When such profiles co-exist in the network, Dynamic TE can be used to free unused reserved bandwidth in the network. Figure 4.16 illustrates a situation where the peak periods for data and voice traffic are completely non-overlapping (due to different timezones). In this situation, unused reserved bandwidth from one kind traffic can be freed during off-peak periods to accommodate peak period traffic of

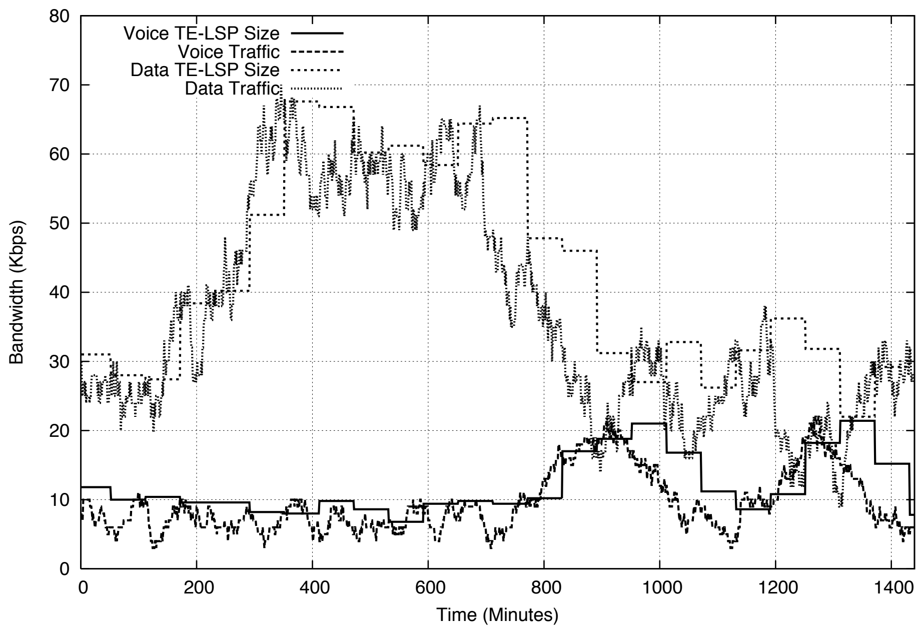


Figure 4.16: Non overlapping data and voice traffic peak periods.

another type. From Figure 4.17 it can be seen that overlapping peak periods (due to same timezone) for both voice and data traffic can cause more links to have higher reserved bandwidth. This also reduces the chances of finding available bandwidth in the network during link failures or sudden traffic spikes.

The motivation is to analyze the performance of Dynamic TE when it is deployed on different traffic types, each with its own characteristic. Lessons learnt from this analysis will contribute to the design and development of a novel bandwidth reservation mechanism that will be allow it to be equally effective with different kinds of traffic.

4.4.1 Performance Analysis

In this section, it is shown that dynamically resized TE-LSPs lead to several benefits without requiring a priori knowledge of the traffic matrix or the type of

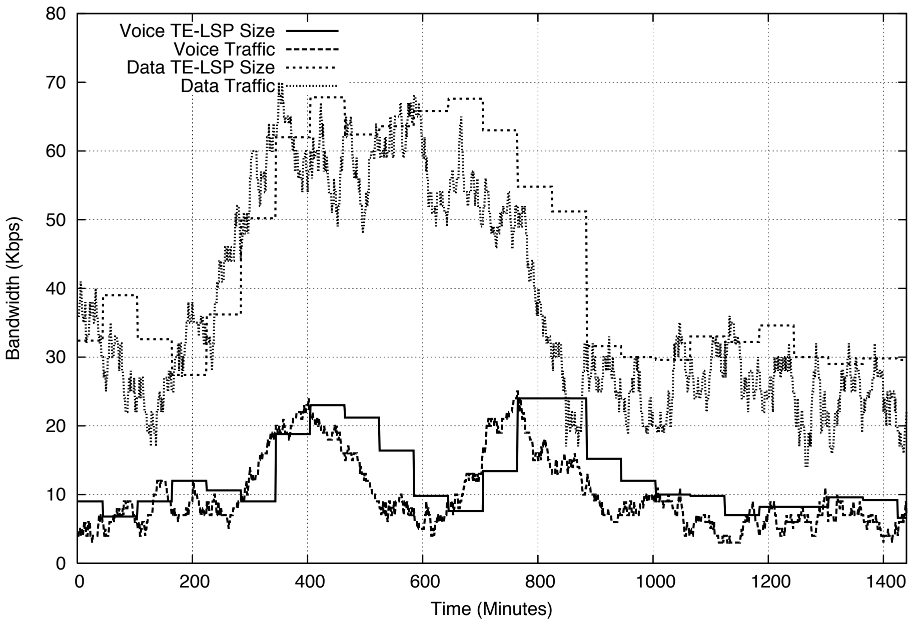


Figure 4.17: Overlapping data and voice traffic peak periods.

traffic. For our experiments, we select the peak based periodic resizing mechanism. It is not claimed as the best option, but instead allows for a worst case analysis from the point of view of over-reservation of bandwidth. Over reservation also causes TE-LSPs to follow undesirable longer paths. A sampling period of 5 minutes and resizing period of 60 minutes has been determined as they are the default setting for routers currently. The goal is to compare the performance of Dynamic TE on mixed traffic from the same timezone with mixed traffic from different timezones only.

Every plot shows the performance of the network for seven days (10080 minutes) of simulated traffic. All results correspond to a mix of 25% voice and 75% data traffic (as obtained from [41]).

Scale Factor=1.0

1. **Fraction of TE-LSPs Not on Their Shortest Path**

Figures 4.18 and 4.19 show behavior of TE-LSP routes in the network with and without failures for Static and Dynamic TE. Both figures capture the fraction of TE-LSPs that are not on their IGP shortest path. Figure 4.18 corresponds to the scenario where all traffic sources (voice and data) are in the same timezone whereas Figure 4.19 corresponds to the scenario where voice and data sources are in different timezones respectively.

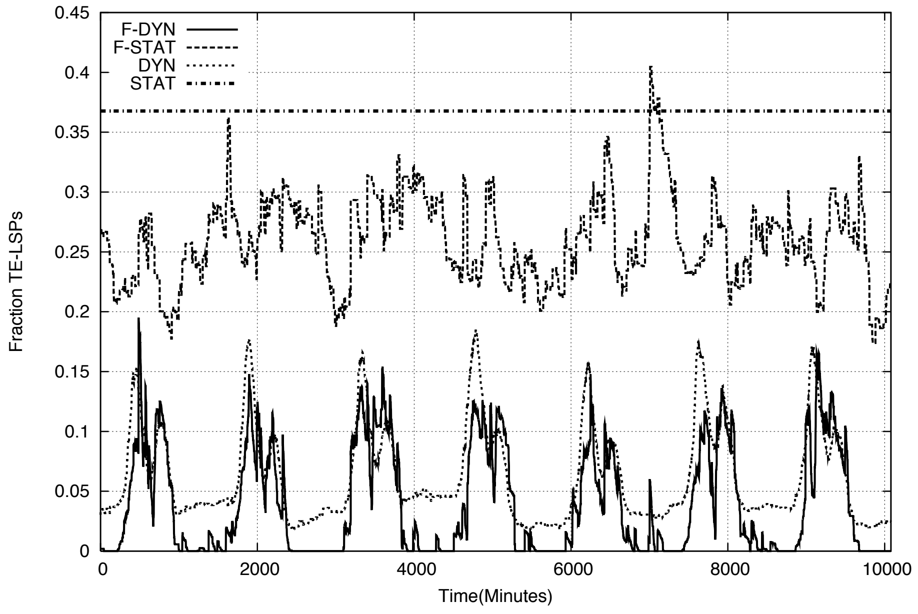


Figure 4.18: Fraction of TE-LSPs not on their shortest path. Overlapping peak periods.

Observation. With Dynamic TE, the fraction of TE-LSPs that are on their shortest paths with traffic having non-overlapping peak periods differ significantly from those that carry traffic having overlapping peak periods.

Hypothesis. In Figure 4.18, since the peak periods overlap, the fraction of TE-LSPs not on the shortest path with Dynamic TE is much higher than the

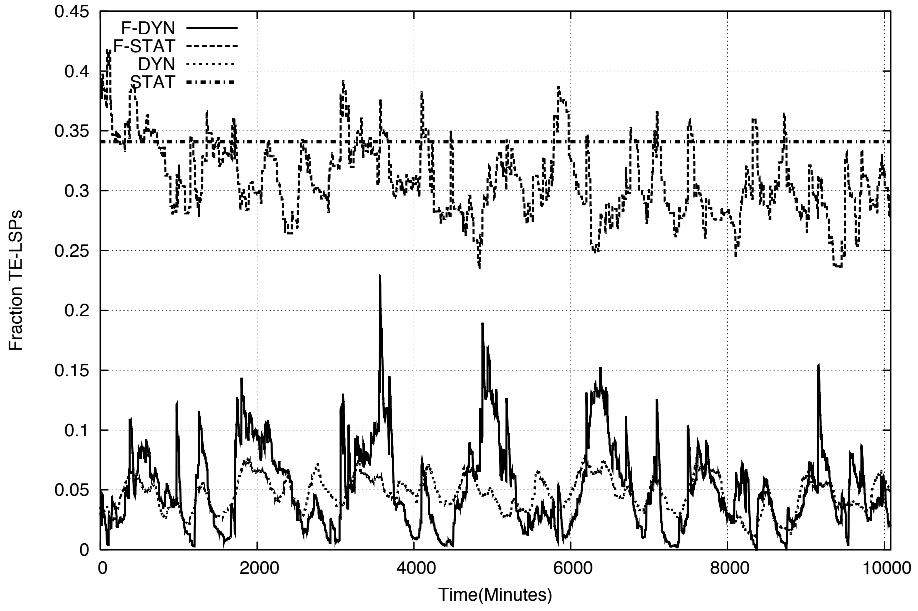


Figure 4.19: Fraction of TE-LSPs not on their shortest path. Non-overlapping peak periods.

scenario where peak periods are non-overlapping, as shown in Figure 4.19. In the non-overlapping case, during any peak period, only one type of traffic (either voice or data) is peaking allowing Dynamic TE to put more TE-LSPs (and corresponding traffic) onto shorter paths. This is different for TE-LSPs carrying traffic with overlapping peak periods. In this case, at any peak time, both traffic types are peaking together, thus requiring their maximum reservation simultaneously. This causes TE-LSPs to be on longer paths in order to fit their reservation since less bandwidth is available.

2. **Maximum TE-LSPs Traversing a Router** Figures 4.20 and 4.21 show the maximum number of TE-LSPs that are passing through a node in the network. It depicts the behavior of only the maximum number and does not convey any information about the associated node. It should be noted that this metric could

be different at every instant, and also does not give any information about the distribution of the loads on nodes. The motivation, as mentioned previously, was to study the worst case load (in terms of maintaining state) occurring in the network. This metric is of significant importance from a provider's perspective since it gives an estimate of increase in state maintenance cost due to the mechanism.

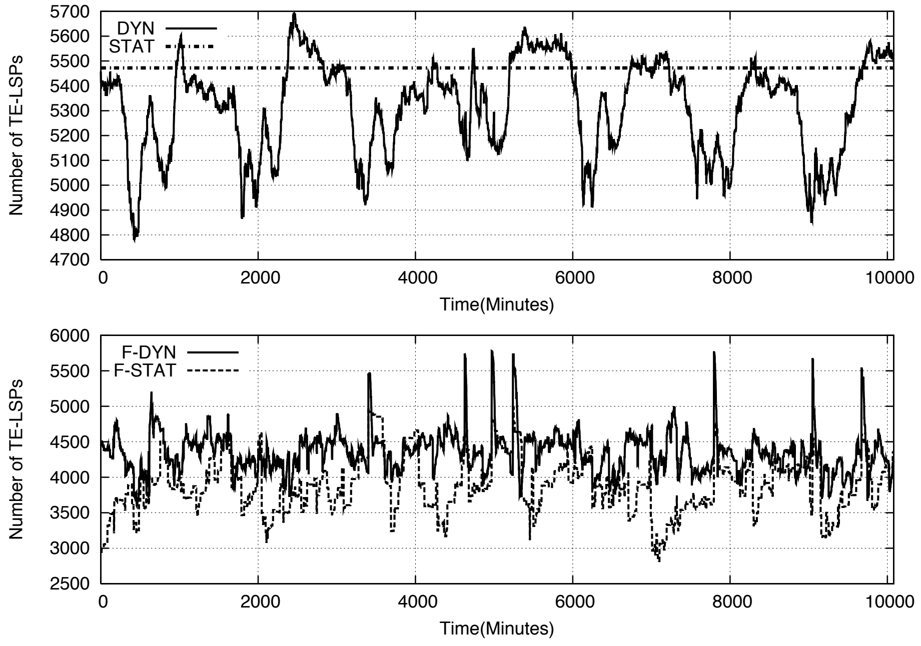


Figure 4.20: Maximum number of TE-LSPs passing a node. Overlapping peak periods.

Observation. Traffic with overlapping peak periods lead to routers having fewer TE-LSPs passing through them as compared to traffic with non-overlapping peak periods.

Hypothesis. As explained previously in Section 4.2, higher bandwidth reservation causes the TE-LSPs to follow longer paths causing them to spread in

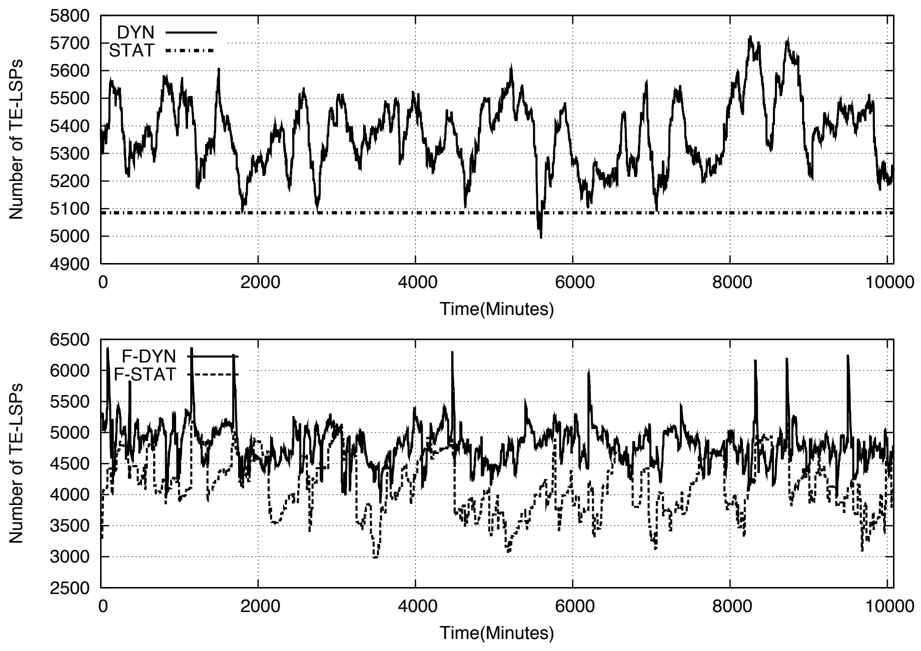


Figure 4.21: Maximum number of TE-LSPs passing a node. Non-overlapping peak periods.

the network. This leads to the routers having fewer TE-LSPs passing through them as in the overlapping peak period case. As only one type of traffic peaks at one time in the non-overlapping case, more TE-LSPs can be routed on shorter paths, leading to heavier router loads.

Scale Factor=2.0

It is now interesting to see the change in behavior when the traffic matrix is doubled. The motivation behind doing so is to see how much extra room created by Dynamic TE can be used to fit more traffic. Most of the metrics behave in similar manner as presented previously and only two metrics are of real interest, underbooking and overbooking fractions. Underbooking is of significant importance as it occurs when the traffic flowing through a TE-LSP is more than the bandwidth reserved for it and is undesirable. It is more favorable to keep this condition down to a minimum.

1. **Unberbooking and Overbooking Fractions**

   Observation. Dynamic TE allows more traffic to be setup in the network. when traffic has non-overlapping peak periods.

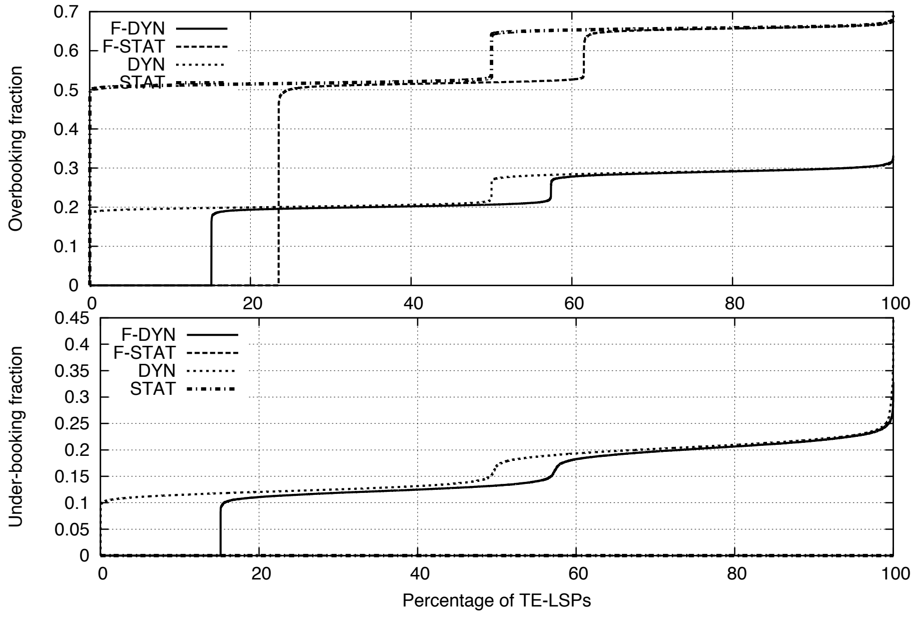


Figure 4.22: Distribution of underbooking and overbooking fractions. Overlapping peak periods. Scale Factor=2.0.

Hypothesis. Figure 4.22 shows the distribution of overbooking and underbooking (defined in Section 2.2) across the TE-LSPs for non-overlapping peak period traffic and a traffic scaling of 2.0. Results for overlapping traffic are similar and have been omitted for the sake of brevity. The STAT and F-STAT cases show a high amount of overbooking taking place since the reservation always stays at its maximum. Also, because of this, there is no underbooking taking place. The TE-LSPs that have 0.0 overbooking are the ones that fail

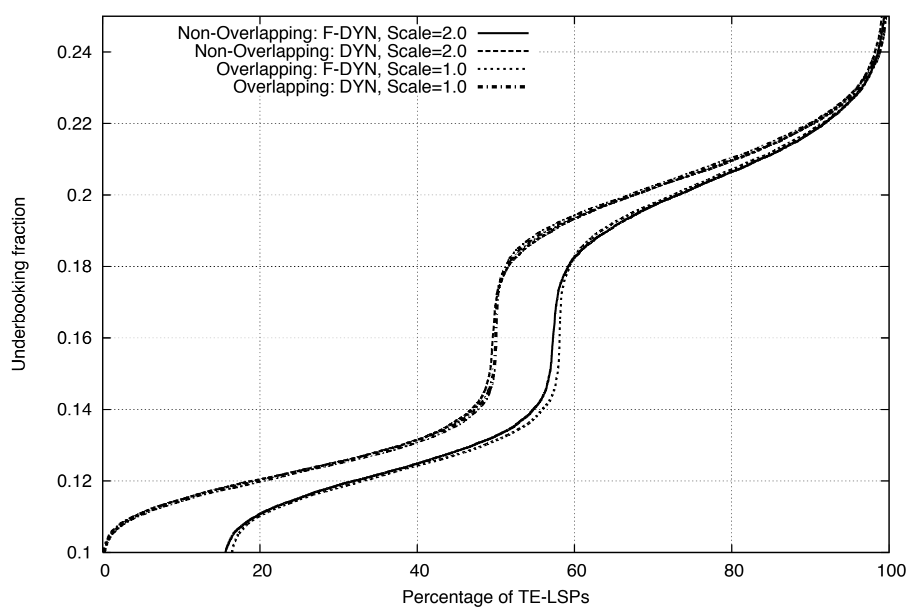


Figure 4.23: Distribution of underbooking fractions. Overlapping and non-overlapping peak periods. Scale Factor=1.0 and 2.0.

during the course of the simulation. The underbooking distribution in Figure 4.22 shows that less underbooking takes place with failures (F-DYN). This is because the extra 'room' created in the network from failed TE-LSPs allow more TE-LSPs to resize.

Underbooking fraction distribution of Scale Factor 1.0 of overlapping peaks with Scale Factor 2.0 of non-overlapping peaks is compared next. This allows the estimate of how much more traffic can fit with Dynamic TE when peak periods do not overlap. Figure 4.23 shows that when peak periods are non-overlapping, the traffic scaled by 2.0 causes similar average underbooking as overlapping peak periods with a scale factor of 1.0. This is because on an average, more bandwidth is available in the network during the respective peak periods for voice and data traffic when the peak periods are *not* overlapping, allowing more traffic to fit in this scenario (Scale Factor=2.0). This shows that a service provider can fit more traffic into its network when they originate from different timezones by placing them intelligently in their network and using Dynamic TE.

2. **TE-LSP Setup Failure**

   Table 4.6 summarizes the results for the number of setup failures and TE-LSPs that are down after failure for the static and dynamic cases. TE-LSP initial setup failures and after link failures are similar for the static peak based TE. This is because reserving at peak values leaves little room to route TE-LSPs. Also, when traffic has overlapping peak periods, less room is available with Dynamic TE during the peak periods as compared to non-overlapping peak periods. This causes more TE-LSPs to fail during link failures with overlapping peak periods than non-overlapping peak periods.

Table 4.6: TE-LSP failure status

Traffic Scale Factor	**Static Peak Based TE**		**Dynamic TE**	
	Failed to setup	Down on link failure	Failed to setup	Down on link failure
Overlapping Traffic				
$k = 1.0$	0	2367	0	2334
$k = 2.0$	1323	4563	0	4154
Non-Overlapping Traffic				
$k = 1.0$	0	2315	0	2212
$k = 2.0$	1314	4587	0	3841

4.4.2 Summary

This section showed that the performance of a dynamic resource allocation mechanism depends on how it is deployed. The fact that different traffic profiles can result in different performance gains was brought to light. An important requirement to be kept in mind when designing a new bandwidth allocation mechanism is that it should be easily deployable on any kind of traffic and should have no dependence on prior profile knowledge.

4.5 Trend Based Bandwidth Provisioning

Performance analysis of Dynamic TE, resizing/rerouting policies and effect of traffic profiles provided insight into the dynamics of the network when a resource reservation mechanism is deployed. This section introduces a novel dynamic bandwidth provisioning scheme for Traffic Engineered tunnels based on the lessons learnt from the previous sections. The mechanism uses information from the traffic trend and traffic trend history to take resizing decisions.

4.5.1 Design Motivation

The motivation for pursuing the development of the Trend-based bandwidth provisioning mechanism is based on the following factors:

1. **Signaling Overhead.** Resizing to follow traffic closely incurs signaling overhead arising from setting up a TE-LSP with the new reserved bandwidth and tearing down the old one to free up resources. Signaling associated with resizing at the protocol level have seldom been studied. To be a competitive deployable alternative, a primary concern is the reduction of associated signaling.

2. **Selective Resizing.** Traffic arises from several different kinds of applications, each of which have respective QoS constraints. It is thus not required for all TE-LSPs to have the same resizing regime.

3. **Distributed in Nature.** MPLS-TE networks are distributed systems where no entity possesses global knowledge. Information is updated using underlying protocol signaling only. Resizing mechanisms should be no different and only rely on local knowledge to provide effective results.

4. **Computational Overhead.** Primary concern for an online mechanism is the computational overhead incurred, as source routers need to run the resizing mechanism simultaneously for several thousand originating TE-LSPs [43]. A resizing mechanism should execute periodically rather than being a continuous background process.

5. **Memory Based.** It is widely observed ([44], [45]) that barring sudden anomalies, traffic is periodic with profiles generally following a daily pattern of peak and off-peak periods. Information from this systematic variation should be used by the resizing mechanism.

6. **Traffic Profiles.** Traffic characteristics differ based on the corresponding application. They include different peak to average ratios, number of peak periods in a day and peak period durations. The mechanism should have consistent performance on every kind of profile.

4.5.2 Capturing the Trend

The proposed mechanism introduces no new functional entities and relies on the existing architecture of present-day routers. There are three aspects to capturing traffic trend.

1. Samples and Their Usage. Sampling is a basic functionality used for traffic measurement and is available on every router. To influence resizing decisions based on past and present, two Circular Queues are used. The first queue, called the *Sample*

Queue (SQ) holds four of the latest samples. The oldest sample gets updated when a new sample is taken as in Figure 4.24. Element i of this queue is represented by Q_i^s. The second queue, called the Weighted Average Queue (WAQ), holds the weighted average of the SQ elements. The weight of an element is inversely proportional to the distance from the queue head. Thus, the latest sample has weight 4, the next one a weight 3 and so on. Similar to the SQ the WAQ is updated every time a new sample is taken. The behavior of the WAQ is similar to that of the temporal average taken over a sliding window in [46] that is fast and easy to control [47]. Element i of the WAQ is represented as Q_i^w. Elements of SQ and WAQ provide an estimate of the general trend of traffic. The weighted average smoothens the samples and prevents fluctuations in traffic from estimating an inconsistent traffic trend. Storage of prior samples also allows traffic history to be kept in account while provisioning for the future. This mechanism uses two sampling periods namely t_{low} and t_{high} where $t_{low} < t_{high}$ and the default sampling period is t_{high}.

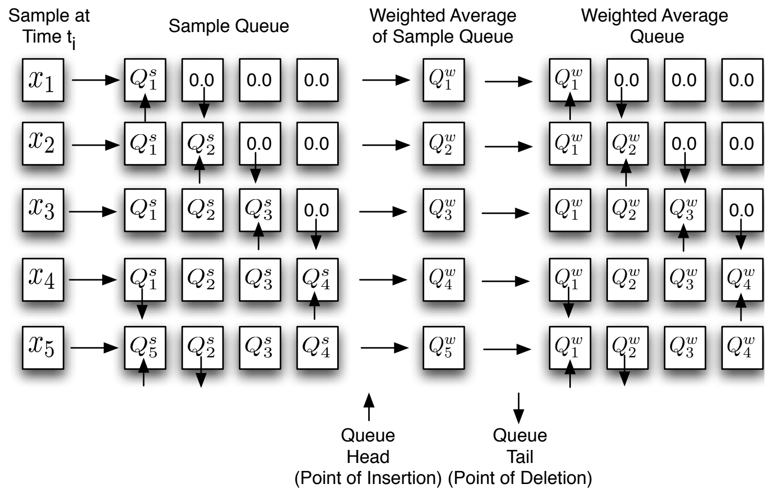

Sample at Time t_i
Sample Queue
Weighted Average of Sample Queue
Weighted Average Queue
x_1
x_2
x_3
x_4
x_5
0.0
Queue Head (Point of Insertion)
Queue Tail (Point of Deletion)

Figure 4.24: Storage of samples and their usage.

2. Slope Based Estimator. The *Slope based Estimator* (SE) calculates the traffic gradient from the WAQ. It is used only when conditions (detailed in the next section) that require resizing are met. The four elements in the WAQ are used to compute three 'slope' values, that provide a *direction* for future reservation values. This is also the reason why only four elements are required. Figure 4.24 shows that when sample x_5 is taken, the three slope values using the WAQ are calculated as.

$$s_1 = \frac{Q_5^w - Q_4^w}{t_5 - t_4}, s_2 = \frac{Q_4^w - Q_3^w}{t_4 - t_3}, s_3 = \frac{Q_3^w - Q_2^w}{t_3 - t_2} \tag{4.7}$$

The value returned by the SE is

$$R_{new}^{slope} = \frac{3 * s_1 + 2 * s_2 + 1 * s_3}{3 + 2 + 1} * (t_5 - t_2) \tag{4.8}$$

3. Memory Based Moderator. Traffic has systematic variations in intensity over the period of a day. Peak and non-peak periods in a day reflect the periodic usage patterns. It is useful as well as important to keep this periodicity in mind and apply its knowledge while resizing. To achieve this, *'Seed Points'* are equally distributed in *time* over the period of one day. At these times, the current weighted average value of the SQ is stored. This provides the traffic trend around this point of time. These seed points help to guide the SE. The *Memory based Moderator* (MM) takes as input a time instance and based on the values stored at the nearest neighboring *Seed Points*, returns an estimated value. For time t_r its location will always occur as $t_i^{seed} < t_r < t_{i+1}^{seed}$ where t_i^{seed} and t_{i+1}^{seed} are neighboring seed points. Similarly, t_{i+2}^{seed} is the seed point following t_{i+1}^{seed}. It is important to consider the values stored at all three seed points since the past, the present, and the future need to be taken into account. A simple weighing factor achieves this. The weighing factor z is deduced from the time values $t_i^{seed}, t_r, t_{i+1}^{seed}$. It is defined as $z = \frac{t_{i+1}^{seed} - t_r}{t_{i+1}^{seed} - t_i^{seed}}$ Let the values stored at the seed points be $Q_i^{seed}, Q_{i+1}^{seed}$ and Q_{i+2}^{seed} respectively. The memory based moderator returns an estimated value for the new reserved bandwidth as

$$R_{new}^{mem} = \frac{z * Q_i^{seed} + Q_{i+1}^{seed} + (1-z) * Q_{i+2}^{seed}}{1 + z + (1-z)} \tag{4.9}$$

It should be noted that *Seed Points* are *cyclic* in nature and the last seed point in a day is followed by the first seed point of the next day. Seed points are updated at the end of a day where corresponding seed points hold the average values from previous days and the current concluding day.

4.5.3 Trend Based Bandwidth Provisioning Algorithm

This section explains the Trend based Bandwidth Provisioning Algorithm in detail.

1. **Control Parameters.** An objective is to keep the bandwidth provisioning algorithm tunable using as few parameters as possible to ease deployment and usability. The algorithm introduced in this paper uses two parameters for control: under-booking and over-booking fractions, denoted x_{ub} and x_{ob} respectively. Both parameters can be expressed in absolute kilobytes or as a fraction of the maximum reservable size of the traffic tunnel (denoted as R_{max}). In this paper, these parameters are expressed as the latter. An instance of under-booking arises when the traffic flowing through the tunnel is less than the bandwidth reserved for it and vice-versa for over-booking. The parameters x_{ub} and x_{ob} denote the maximum tolerable/allowable under-booking or over-booking for the corresponding traffic (before resizing is required).

2. **Trigger Conditions and Sampling Change.** Based on the samples, *it is more important to react faster and more aggressively to increasing traffic than*

decreasing traffic. This idea is one of the main design criteria for the algorithm.

(a) Increasing Reservation. When the latest sample crosses the current reserved bandwidth by *at least* $x_{ub} * R_{max}$ for the first time, more samples are taken with period t_{low}. This leads to a low reaction time and more sensitivity to under-booking. The algorithm then waits for all SQ elements to be values greater than the current reservation by at least $x_{ub} * R_{max}$. It then changes the reservation based on the SE or MM. The sampling period remains t_{low} as long as consecutive samples are above the current reserved bandwidth by at least $x_{ub} * R_{max}$. If a sample fails to cross the current bandwidth by at least $x_{ub} * R_{max}$ before the SQ is full of such samples, the sampling period changes to t_{high}. Waiting for the SQ to get filled before deciding to resize prevents sudden spikes from causing resizes. These steps are illustrated in Figure 4.25.

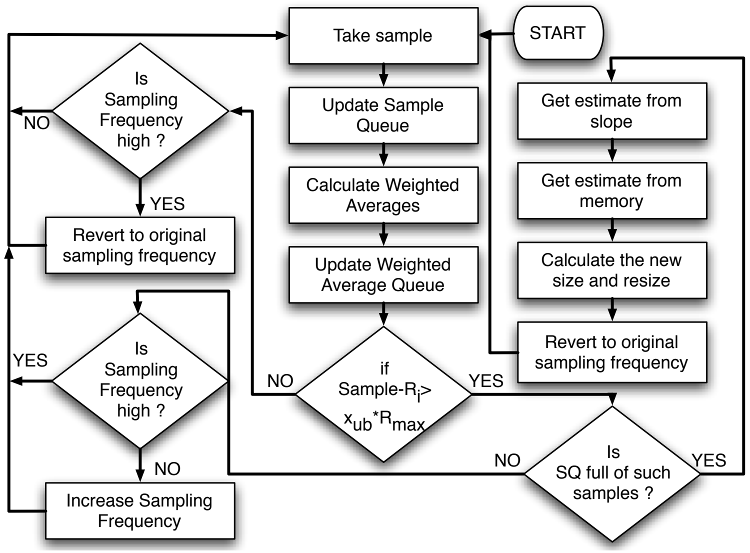

Take sample
START
Is Sampling Frequency high ?
NO
YES
Update Sample Queue
Get estimate from slope
Calculate Weighted Averages
Get estimate from memory
Revert to original sampling frequency
Update Weighted Average Queue
Calculate the new size and resize
Is Sampling Frequency high ?
YES
Revert to original sampling frequency
NO
if Sample-R$_i$> x$_{ub}$*R$_{max}$
YES
NO
Is SQ full of such samples ?
YES
NO
Increase Sampling Frequency

Figure 4.25: Algorithm flowchart for increasing reservation.

(b) Decreasing Reservation. The trigger condition to reduce reservation is less reactive and takes a more *cautious* approach. Once the *weighted average* value of the SQ falls below the reserved bandwidth by a least $x_{ob} * R_{max}$ for the first time, a counter is started. The sampling period however, is not reduced. When the WAQ contains consecutive values that are less than the current reserved bandwidth by at least $x_{ob} * R_{max}$ for *one hour*, the *current* weighted average value of the SQ and MM are then used to change the reservation to a lower value. A larger or smaller duration/wait period can be chosen for decreasing reservation to tune the conservative nature of the algorithm.

3. **Final Provisioned Bandwidth.** The final reservation depends on the values returned by the SE and the MM.

(a) Increasing Reservation. Increasing reservation is based on the following:

$$R_{new} = max(R_{new}^{slope}, R_{new}^{mem} + (x_{ob} - x_{ub}) * R_{max}) \tag{4.10}$$

This decision selects a value based on the current trend and the past variation of traffic.

(b) Decreasing Reservation. When, reservation has to be lowered, the algorithm is less reactive and uses either the MM or the weighted average value of the SQ to change reservation. Thus,

$$R_{new} = min \begin{cases} Q_t^w + (x_{ob} - x_{ub}) * R_{max} & \text{if } Q_t > R_{new}^{mem} \\ R_{new}^{mem} + (x_{ob} - x_{ub}) * R_{max} & \text{otherwise} \end{cases} \tag{4.11}$$

An over-provisioning factor of $(x_{ob} - x_{ub})$ is used to over-provision enough bandwidth to prevent resizing again if a steady decreasing traffic is followed by a sudden *spike* and a *dip* in traffic where both changes cross the thresholds x_{ob} and x_{ub} respectively. Having the difference $(x_{ob} - x_{ub})$ as a factor prevents a resize if such a condition arises since none of the thresholds will be crossed. Over-provisioning during downsizing also leads to lesser resizing when traffic is growing as reserved bandwidth reaches the peak sooner.

4.5.4 Simulation Results

In line with the motivation, simulations have been undertaken on realistic traffic profiles spanning large durations of time. Topologies and traffic matrices that reflect present-day service provider deployment have been used.

Simulation Setup

In line with the motivation, simulations have been undertaken on realistic traffic profiles spanning large durations of time. Topologies and traffic matrices that reflect present-day service provider deployment have been used. Details of the traffic profiles and topologies are given in Tables 4.8. Figure 4.26 also shows the changes in reservation brought about by the algorithm. Seven days of traffic (each day having the peak period characteristics) for each type of profile is used to study the performance of the algorithm. For sake of brevity and lack of space, only the details for MESH will be presented. The sampling period $t_{high} = 5$ minutes, $t_{low} = 2$ minutes. The set UP of thresholds for LSA flooding when reserved bandwidths increases on a link is: UP $= [15, 30, 45, 60, 75, 80, 85, 90, 95, 97, 98, 99, 100]$ with the set DOWN being the same in reverse order. These thresholds have been obtained from [13].

Performance Metrics

This section explains the metrics that will be used to analyze the performance of the proposed mechanism. These metrics pertain only to this mechanism and hence are defined here separately.

1. **Average Underbooking and Overbooking Durations.** If c_t^i is the current traffic flowing through TE-LSP i, underbooking occurs when $c_t^i > x_{ub} * R_{max} + R_t^i$ and overbooking occurs when $c_t^i + x_{ob} * R_{max} < R_t^i$. These metrics capture the fraction of the total time underbooking and un-required overbooking take place. If t_{ub}^k and t_{ob}^k are variables that capture an instance of underbooking and overbooking respectively at time t for TE-LSP k, then the values assigned are

Table 4.7: Details of the topologies used in simulations

Network Description							TE-LSP Distribution		
Name	# of nodes	# of links	OC3 links	OC48 links	OC96 links	OC192 links	[0,20) Mbps	[20,50) Mbps	(50,1500] Mbps
MESH	83	167	0	132	0	35	5105	1701	0
ISP1	158	280	65	92	0	123	1916	280	346
ISP2	88	168	0	49	9	110	1510	149	188

Table 4.8: Traffic profiles used in the simulations

Type of Traffic	Profile Description		
	# of Peak Periods	Peak Duration	Peak:Average Ratio
P_1: Data	1	6 hours	2:1
P_2: Data	1	14 hours	4:1
P_3: Voice	2	4 hours	4:1

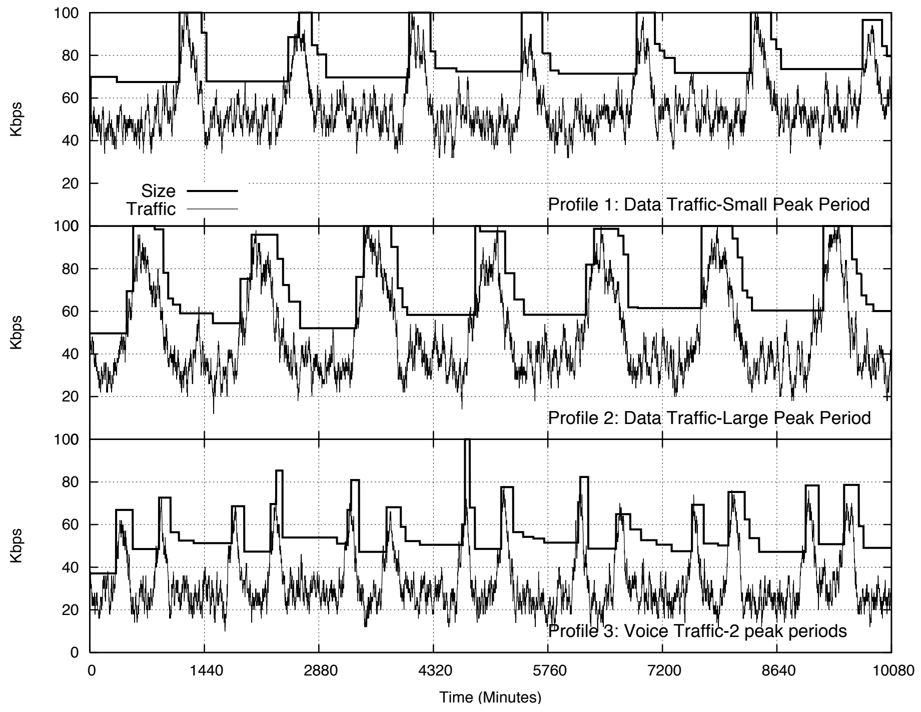


Figure 4.26: Traffic profiles and reservation resizing. $x_{ub} = 0.005, x_{ob} = 0.25$.

$$t^i_{ub} = min \begin{cases} 1 & \text{for } c^i_t > x_{ub} * R_{max} + R^i_t \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

$$t^i_{ob} = min \begin{cases} 1 & \text{for } c^i_t + x_{ob} * R_{max} < R^i_t \\ 0 & \text{otherwise} \end{cases} \tag{4.13}$$

Average values for these metrics as a fraction of the total time are defined as $T^i_{ub} = \frac{1}{T}\sum_{t=1}^{T} t^i_{ub}$ and $T^i_{ob} = \frac{1}{T}\sum_{t=1}^{T} t^i_{ob}$

2. **Average Underbooking.** If the instantaneous traffic and reservation at time t for TE-LSP i is c^i_t and R^i_t respectively, then the average underbooking for a TE-LSP i is defined as $B^i_{ub} = \frac{1}{\sum_{t=1}^{T} t^i_{ub}} \sum_{t=1}^{T} \max(\frac{c^i_t - (R^i_t + x_{ub} * R^i_{max})}{R^i_{max}}, 0.0)$, where T is the duration of simulation.

3. **Average Overbooking.** The average unwanted over-booking is defined as $B_{ob}^{i} = \frac{1}{\sum_{t=1}^{T} t_{ob}^{i}} \sum_{t=1}^{T} \max(\frac{R_t^i - (x_{ob} * R_{max}^i + c_t^i)}{R_{max}^i}, 0.0)$, where T is the duration of simulation. The algorithm should ideally free up unused bandwidth whenever possible, and lower the average overbooking.

4. **Average Number of Resizes.** Excessive resizing leads to signaling overhead and traffic disruptions. This metric captures the average number of times a TE-LSP is resized. If the number of resizes of TE-LSP i is denoted by n_i, for a total of K TE-LSPs in the network, the average number of resizes is $N_{avg} = \frac{1}{K} \sum_{i=1}^{K} n_i$.

5. **Average Inter-Resize Time.** If resizing occurs very frequently it will cause frequent traffic shifts leading to instability and degrading QoS. If resizes to R_i^k and R_{i+1}^k for TE-LSP k occur at time t_i and t_{i+1}, the average inter-resize time is $T_R^k = \frac{1}{n_k} \sum_{i=1}^{n_k} (t_{i+1} - t_i)$, where n_k is the number of resizings for TE-LSP k.

6. **Number CAC Rejections per Unit Time.** Less resizing reduces the possibility of CAC rejections. If L_t^{cac} denote the number of LSAs originating at time t due to a CAC rejection, the number of CAC rejections per unit time is $L_{avg}^{cac} = \frac{1}{T} \sum_{i=1}^{T} L_t^{cac}$, where T is the duration of simulation.

7. **Average Number of Flooded LSAs due to Threshold Crossing.** Less resizing may result in less thresholds being crossed on links resulting in reduced flooded LSAs. Similarly to the above metric, it is denoted as $L_{avg}^{th} = \frac{1}{T} \sum_{i=1}^{T} L_t^{th}$, where T is the duration of simulation.

8. **SE, MM and Queue Usage.** During resizing, one of the entities amongst the SE, MM or the WAQ is used. Keeping track of their usage shows their contribution in the working of the algorithm. M_i^{up}, S_i^{up} denote the number of times the MM and SE were used for upsizing and M_i^{down}, Q_i^{down} denote the number of

times the MM and WAQ were used for downsizing TE-LSP i. Expressed as a fraction of total resizes,

$$M_{avg}^{up} = \frac{1}{K}\sum_{i=1}^{i=K}\frac{n_i}{M_i^{up}} \text{ and } S_{avg}^{up} = \frac{1}{K}\sum_{i=1}^{i=K}\frac{n_i}{S_i^{up}} \tag{4.14}$$

$$M_{avg}^{down} = \frac{1}{K}\sum_{i=1}^{i=K}\frac{n_i}{M_i^{down}} \text{ and } Q_{avg}^{down} = \frac{1}{K}\sum_{i=1}^{i=K}\frac{n_i}{Q_i^{down}} \tag{4.15}$$

Performance Analysis

1. **Varying x_{ub}.**

   This section shows the performance of the algorithm with varying x_{ub} and a constant x_{ob}. Table 4.9 tabulates the results.

   (a) **Average Number of Resizes.**

   The amount of resizing depends on three aspects. The value of x_{ub}, number of peak periods in a day, and ratio of peak to average traffic. Sensitive traffic requiring low x_{ub} increases algorithm reaction to increasing traffic causing more upsizing on small traffic increases.

   Observation. The N_{avg} column from Table 4.9 shows that increasing tolerance by increasing x_{ub} reduces resizing and increases T_R.

   Hypothesis. Reductions arise from lesser downsizing ($M_{avg}^{down} + Q_{avg}^{down}$). Low x_{ub} results in high ($x_{ob}-x_{ub}$) leading to more over-provisioning (Eq. 4.11) to prevent under-booking in the future. However, when traffic decreases, it crosses the x_{ob} threshold often, resulting in more downsizes. Also, since lower x_{ub} causes less but aggressive upsizing and over-provisioning, it takes several downsizings to reduce the reservation. Though lowering x_{ub} causes a small increase in upsizing required to provision for the peak, downsizing is less since over-provisioning is less. Amongst the different profiles,

P_1 with the smallest peak to average ratio of traffic has the least average resizing since it takes lesser upsizing to reach the peak and then lesser downsizing to reach the average. Profile P_2 has a higher peak to average ratio of traffic than P_1 and thus leads to more resizing. Profile P_3 leads to the most resizing since it has two peaks as well as similar peak to average traffic ratio as P_2.

Table 4.9: Performance for varying x_{ub}

x_{ub}	N_{avg}	T_R hour	M^{up}_{avg} %	S^{up}_{avg} %	M^{down}_{avg} %	Q^{down}_{avg} %	L^{cac}_{avg} min^{-1}	L^{th}_{avg} min^{-1}
	Performance Metrics for Profile P_1							
0.005	29.53	5.49	25.06	3.05	38.28	33.59	.18	1.42
0.01	28.80	5.60	25.34	3.18	38.37	33.08	.17	1.41
0.02	27.49	5.82	25.76	3.64	38.12	32.45	.18	1.42
0.03	26.31	6.05	26.08	4.21	37.63	32.05	.17	1.38
0.04	25.21	6.29	26.40	4.76	36.84	31.97	.17	1.35
	Performance Metrics for Profile P_2							
0.005	45.24	3.52	19.43	3.24	40.21	37.10	.22	1.98
0.01	44.14	3.61	19.46	3.50	40.46	36.55	.22	1.97
0.02	41.99	3.78	19.49	4.12	40.72	35.65	.21	1.84
0.03	39.87	3.97	19.53	4.72	40.78	34.94	.21	1.79
0.04	37.98	4.16	19.55	5.30	40.74	34.39	.20	1.78
	Performance Metrics for Profile P_3							
0.005	56.22	2.85	11.45	19.58	43.91	25.04	.22	1.69
0.01	54.72	2.93	11.00	20.06	44.16	24.76	.21	1.66
0.02	51.98	3.08	9.90	21.35	44.19	24.53	.21	1.60
0.03	49.59	3.22	8.67	22.77	43.84	24.69	.21	1.58
0.04	47.61	3.35	7.45	24.24	43.15	25.13	.20	1.53

(b) **Upsizing and Downsizing.**

Observation. Eq. 4.11 shows that decreasing reservation is always over-provisioned by a factor of $(x_{ob} - x_{ub})$ leading to higher reservation for lower values of x_{ub}.

Hypothesis. Since reservation is at a high value, it takes fewer upsizings

$(M^{up}_{avg} + S^{up}_{avg})$ to reach the reservation for peak traffic when traffic starts to grow. This behavior occurs consistently within each profile. For the same value of x_{ub}, Profile P_1 requires less upsizing than P_2 since it has a lesser peak to average ratio and the peak reservation takes place sooner. Profile P_3 has more upsizing than P_2 due to the presence of two peak periods where reservation decreases before increasing again. Amongst the profiles, P_3 uses the SE more than the MM on upsizing due to high slopes that results as a combination of the high peak to average ratio of traffic and the low duration of the peak period. Profiles P_1 and P_2 on the other hand use the MM more than the SE for upsizing as the peak period is spread over a longer duration of time resulting in lower slope values. Higher peak to average traffic ratio causes relatively more downsizing. Table 4.9 shows that for the same value of x_{ub} and x_{ob}, more downsizing takes place for Profile P_2 than for P_1 even though they have the same number of peak periods. This is due to the peak to average ratio being larger for P_2. Similarly, more downsizing took place for P_2 compared to P_3 due to the former having a longer period over which traffic reduces eventhough both have similar peak to average traffic ratio. Profile P_3 having two close peak periods does not provide much room for downsizing between them. In all the downsizing cases, the MM is used more than the value of the weighted average from the sample queue. The MM has more accurate estimates as it continually updates the Seed Points by weighing the previously held value and the currently observed values.

(c) **LSA Flooding.**

Observation. Lesser resizing of all TE-LSPs lead to lesser LSA flooding.

Hypothesis. This is because less thresholds are crossed as the total reservation on links change less. Less resizing also reduces CAC rejections as TE-LSPs are rerouted less. Higher x_{ub} values cause less resizing and rerouting and hence less LSA flooding. The number of LSAs flooded in the network with time are illustrated in Figure 4.27 and 4.28 for $x_{ub} = 0.005$, $x_{ob} = 0.10$ and $x_{ob} = 0.25$. Correlating Figures 4.27 and 4.28 with Profile P_1 in Figure 4.26, it is seen that when traffic is peaking resulting in upsizing, more LSAs are flooded due to an increase in CAC rejection and threshold crossing on the links. Similarly, when downsizing starts after a peak period is over, more LSAs are flooded for the same reasons.

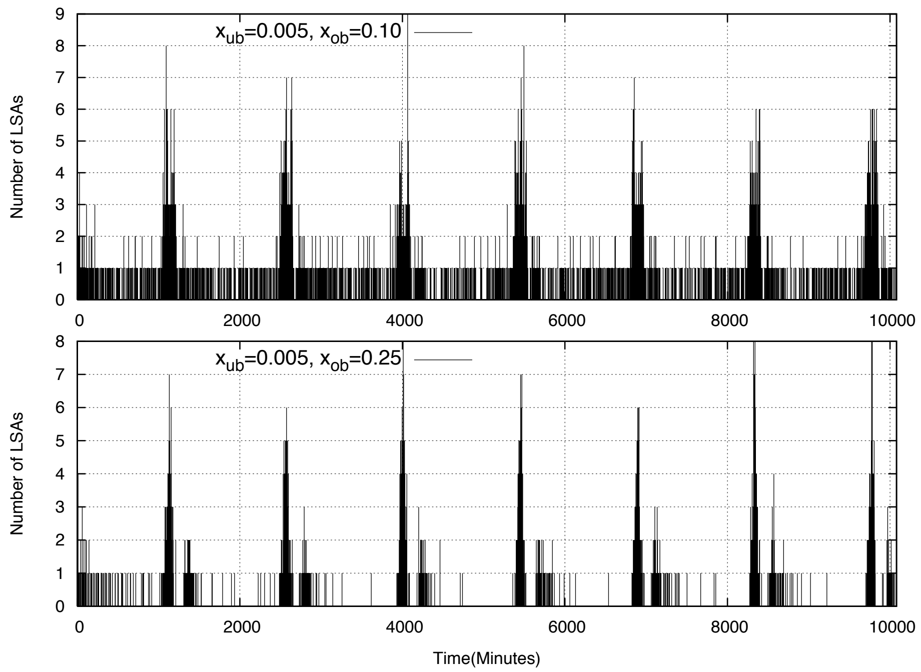


Figure 4.27: LSA flooding due to CAC Rejections.

(d) **Booking.**

Observation. Figure 4.29 and 4.30 show the performance of the algorithm

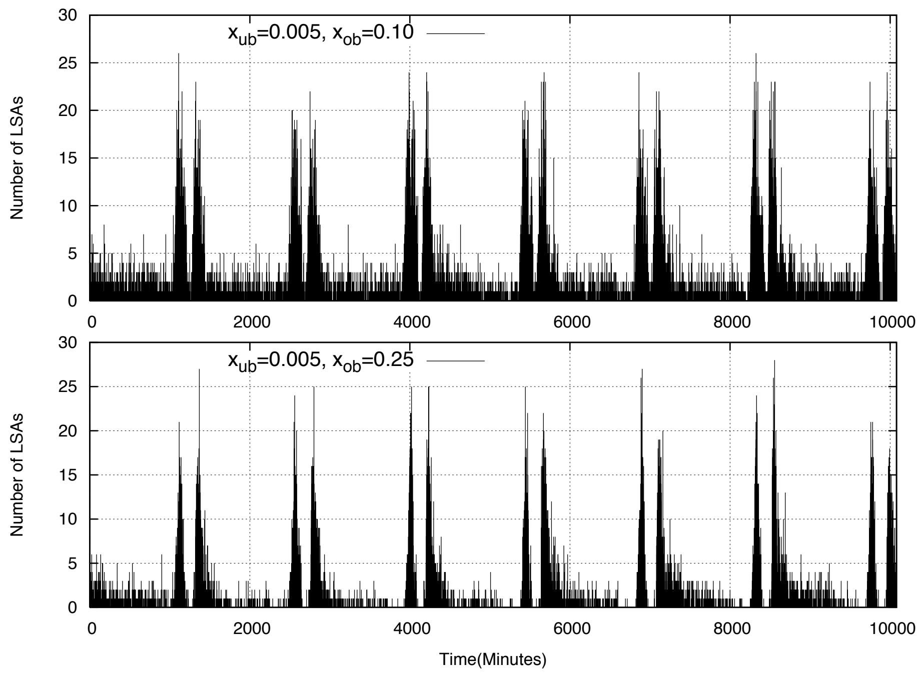


Figure 4.28: LSA flooding due to threshold crossing on links.

with respect to booking. Figure 4.29 shows the distribution of average underbooking for all TE-LSPs for Profile P_1 on MESH. For all values of x_{ub}, the average underbooking (B^i_{ub}) lies between 2% and 6% for MESH. The high degree of overlapping in the underbooking distribution shows that the algorithm has similar reaction times to traffic growth irrespective of x_{ub} values. The algorithm performs best with $x_{ub} = 0.005$ in terms of *total* underbooking which ranges from 2.5% to 6.5% ($[0.02+x_{ub}, 0.06+x_{ub}]$) as opposed to the values for $x_{ub} = 0.04$ which range from 6% to 10%. Also, $x_{ub} = 0.005$ also causes the largest *total* overbooking ranging from 30% to 33% ($[0.05 + x_{ob}, 0.08 + x_{ob}]$).

Hypothesis. This happens due to the relatively high over-provisioning taking place for sensitive traffic. Figure 4.30 shows the distribution of the fraction of total time when underbooking and overbooking instances took place. Again, for all values of x_{ub}, the behavior of the distribution is similar and lies between 0.6% and 2.2% of the total time. This behavior further corroborates the insensitivity of the reaction time of this algorithm to this parameter. From Figures 4.29 and 4.30, it is seen that average *total* underbooking ($x_{ub} + 0.02$) takes place for the shortest durations when $x_{ub} = 0.005$. Figure 4.31 shows two important observations. Firstly, the average underbooking is not dependent on TE-LSP sizes. Secondly, large average underbooking takes place for very short durations. Figure 4.31 brings to light the information hidden from Figures 4.29 and 4.30.

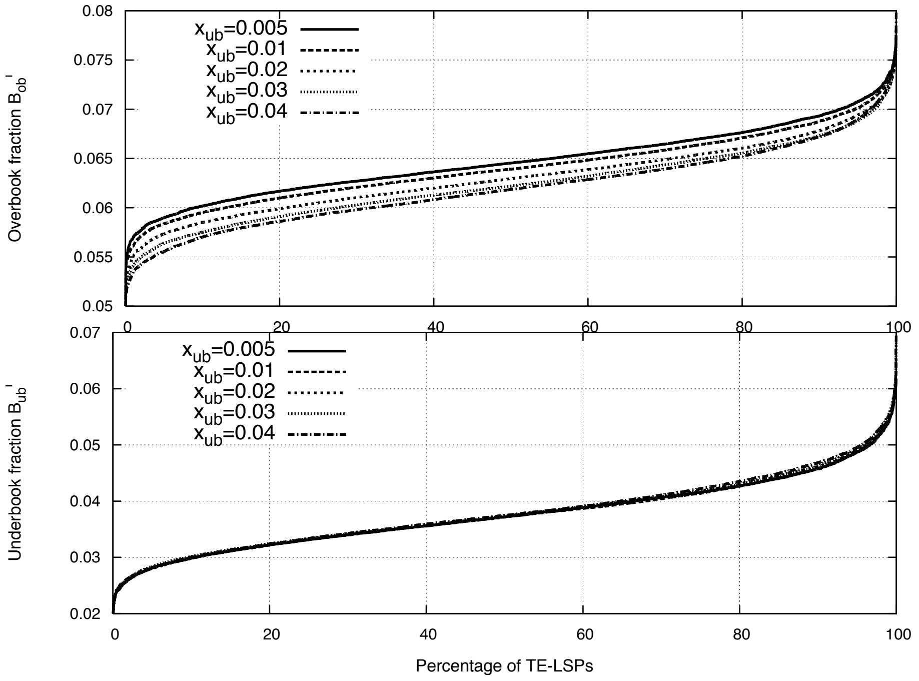


Figure 4.29: Distribution of B^i_{ub} and B^i_{ob}. $x_{ob} = 0.25$, varying x_{ub}.

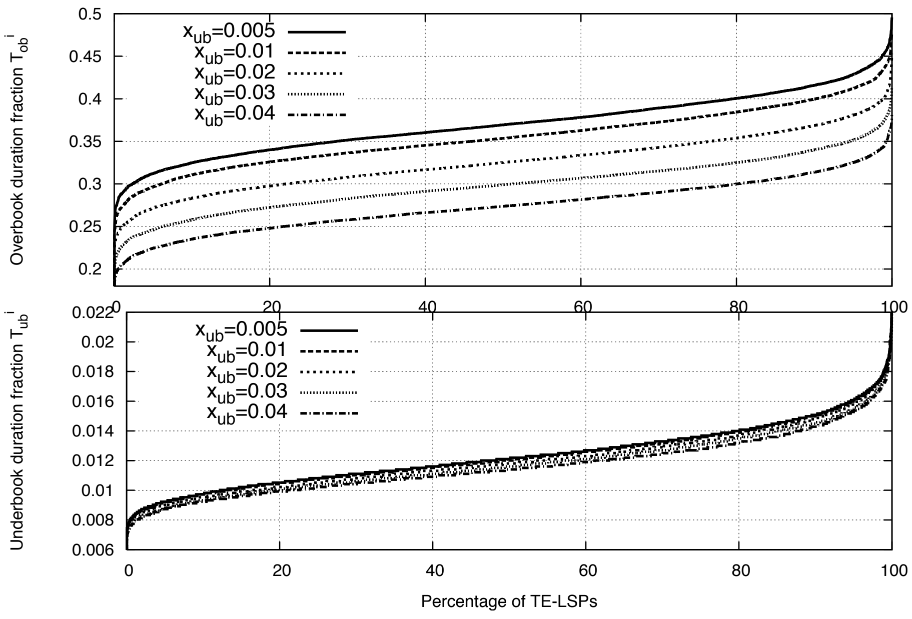


Figure 4.30: Distribution of T_{ub}^i and T_{ob}^i. $x_{ob} = 0.25$, varying x_{ub}.

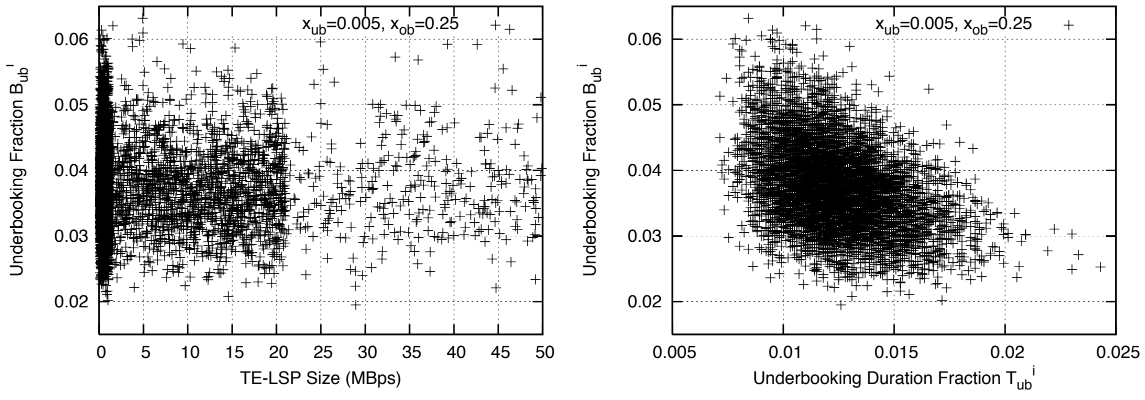


Figure 4.31: Scatter Plot of B_{ub} with TE-LSP sizes and T_{ub} for $x_{ub} = 0.005$.

2. **Varying x_{ob}.**

This section discusses the performance of the algorithm with varying x_{ob} and $x_{ub} = 0.005$. This value of x_{ub} provides the best performance in terms of reducing underbooking. Results are tabulated in Table 4.10. Some metrics have similar behavior as in Table 4.9 and will not be discussed for brevity. Also due to space constraints, analysis for profile P_1 is presented only.

(a) **Upsizing and Downsizing.**

Observation. For a fixed x_{ub}, lower values of x_{ob} lead to more upsizing $(M^{up}_{avg} + S^{up}_{avg})$.

Hypothesis. When traffic decreases, reservation is over-provisioned by a factor $(x_{ob} - x_{ub})$. Since the degree of over-provisioning is *less* for lower values of x_{ob}, more upsizing needs to take place on increasing traffic in order to reserve for the peak. Lower values of x_{ob} lead to higher usage of the SE (S^{up}_{avg}) since the difference in the relatively lower current reserved bandwidth and increasing traffic leads to high slopes for the same value of x_{ub}. The trend of SE and MM usage amongst the different profiles remains the same and differ only in actual numbers due to differences in duration, number of peak periods and peak to average traffic ratio. High values of x_{ob} can cause more downsizing $(M^{down}_{avg} + S^{down}_{avg})$. When traffic is decreasing, and is below the reservation by the threshold set by x_{ob}, small decreases can cause more downsizing. The factor $(x_{ob} - x_{ub})$ creates a large difference between the traffic and the reserved bandwidth making it easier to meet downsizing conditions. Thus, having a very high is as undesirable as a very low x_{ob}.

(b) **Booking.**

Observation. Increasing x_{ob} causes more over-booking as expected, but also affects underbooking.

Hypothesis. Figure 4.32 shows that the lowest value of $x_{ob} = 0.10$ causes the least underbooking. However, Figure 4.33 shows that this corresponds to the largest fraction of time compared to the higher x_{ob} values. This

Table 4.10: Performance on MESH for varying x_{ob}

x_{ub}	N_{avg}	T_R hour	M_{avg}^{up} %	S_{avg}^{up} %	M_{avg}^{down} %	Q_{avg}^{down} %	L_{avg}^{cac} min^{-1}	L_{avg}^{th} min^{-1}
	Performance Metrics for Profile P_1							
0.10	77.58	2.15	8.39	27.52	34.76	29.31	.37	2.96
0.15	50.43	3.28	13.84	17.54	38.56	30.04	.26	2.09
0.20	35.37	4.59	20.23	7.61	39.70	32.44	.20	1.63
0.25	29.53	5.49	25.05	3.05	38.26	33.61	.18	1.46
0.30	27.31	6.03	27.61	1.78	40.40	30.18	.18	1.39
	Performance Metrics for Profile P_2							
0.10	87.01	1.89	11.99	21.35	30.92	35.71	.34	3.11
0.15	63.19	2.56	16.04	13.87	33.78	36.27	.29	2.33
0.20	51.66	3.10	18.75	7.84	36.94	36.45	.25	2.10
0.25	45.21	3.53	19.41	3.25	40.21	37.10	.22	1.96
0.30	39.66	4.00	19.84	.92	42.71	36.49	.20	1.81
	Performance Metrics for Profile P_3							
0.10	90.67	1.81	6.31	30.72	32.90	30.05	.29	2.46
0.15	72.53	2.23	8.17	26.84	36.12	28.84	.26	2.10
0.20	62.72	2.57	9.99	23.12	39.87	26.98	.24	1.75
0.25	56.20	2.86	11.46	19.58	43.89	25.04	.21	1.64
0.30	51.88	3.10	12.91	16.27	47.23	23.56	.21	1.68

happens due to the very frequent up and downsizings taking place because of low x_{ub} and x_{ob} values. Also, Figure 4.32 shows that lower x_{ob} unintuitively causes more overbooking for longer durations. This is because aggressive upsizings are caused more on small traffic increases using the slope detector (S_{avg}^{up} from Table 4.10).

3. **Varying Seed Points and Sampling Frequency.**

This section presents analysis for varying Seed Points and change in sampling periods. Simulations were run for three different seed points shown in Table 4.11. The values of $x_{ub} = 0.005$ and $x_{ob} = 0.20$ were fixed.

Observation. Varying Seed Points results in almost no change in resizing behavior and the underbooking fraction and corresponding durations show similar trend as in Figure 4.29 and 4.30.

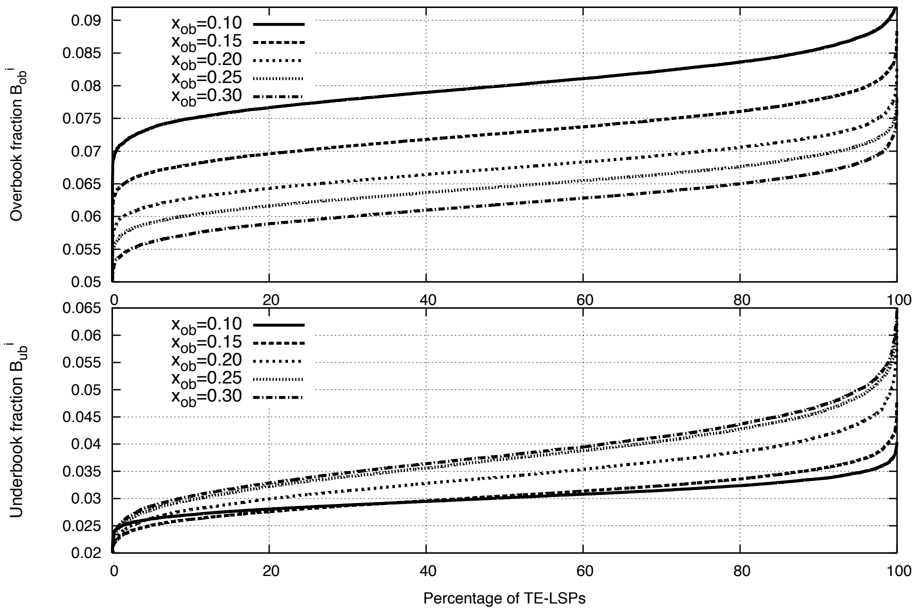


Figure 4.32: Distribution of B^i_{ub} and B^i_{ob}. $x_{ub} = 0.005$, varying x_{ob}.

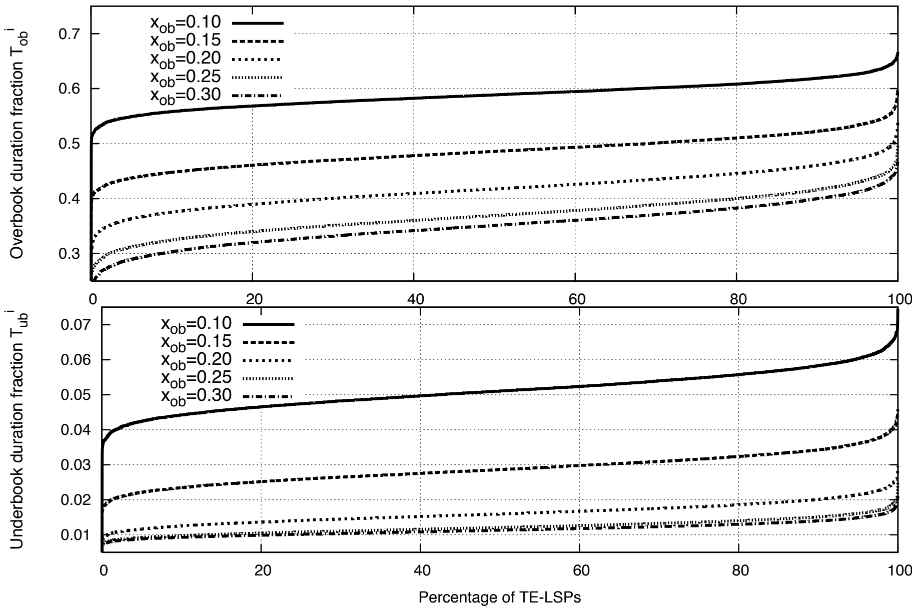


Figure 4.33: Distribution of T^i_{ub} and T^i_{ob}. $x_{ub} = 0.005$, varying x_{ob}.

Hypothesis. These results show that as long as the number of Seed Points are sufficient enough to capture the systematic variation in traffic, performance is not affected. For example, if the number of seed points is 4, they might fail to capture a peak period lasting for 3 hours depending on how they are placed in time. Figure 4.34 and 4.33 show that better QoS is achieved by reducing the sampling period t_{low}. Lesser underbooking takes place for much lesser fractions of time when $t_{low} = 1$ minute. This happens due to a decrease in reaction time when sampling frequency is increased (inter-sample time is decreased) making the algorithm detect under-booking sooner and react quicker.

Table 4.11: Performance on MESH for varying seed points

	Performance Metrics for Profile P_1							
Seed Points	N_{avg}	T_R $hour$	M_{avg}^{up} %	S_{avg}^{up} %	M_{avg}^{down} %	Q_{avg}^{down} %	L_{avg}^{cac} min^{-1}	L_{avg}^{th} min^{-1}
6	30.05	5.41	24.96	3.42	36.77	34.82	.18	1.48
12	29.52	5.49	25.06	3.04	38.25	33.62	.18	1.47
24	29.52	5.49	25.06	3.04	38.25	33.62	.18	1.47
t_{low} min	Performance Metrics for Profile P_1							
3	29.38	5.52	24.77	3.03	38.30	33.87	.18	1.46
2	29.52	5.49	25.06	3.03	38.25	33.63	.18	1.42
1	29.82	5.44	25.27	3.16	38.19	33.34	.18	1.43

The Trend Based mechanism has been compared to similar mechanisms in [35] and [36]. Comparison results are published in [39].

4.5.5 Summary

In this section a novel bandwidth provisioning technique based on the traffic trend was proposed. The prime motivations included reduction of computational and signaling overhead while adhering to the QoS requirements. New entities to evaluate the

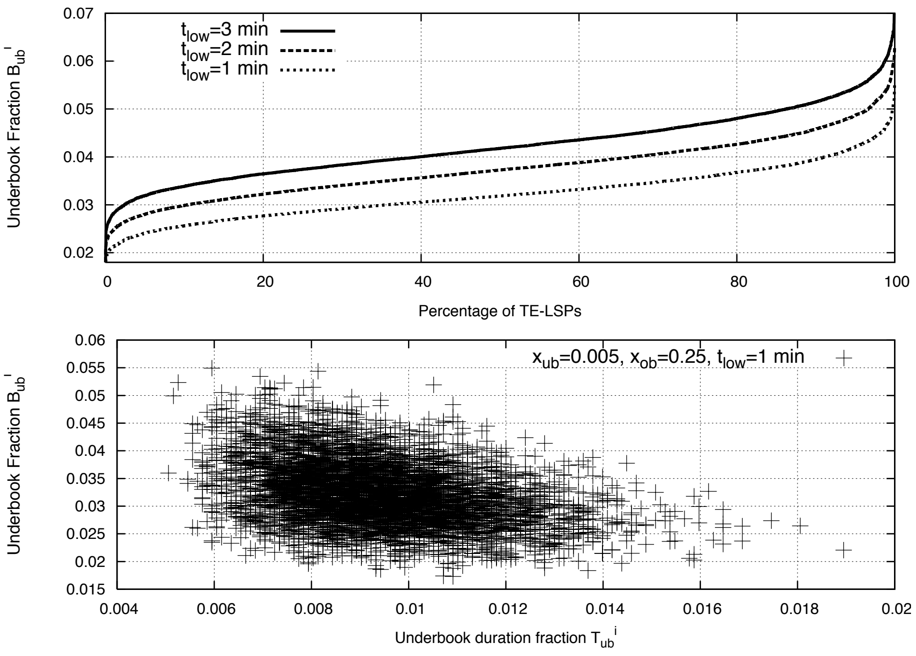


Figure 4.34: Distribution of B^i_{ub} for varying t_{low} and T^i_{ub} behavior with B^i_{ob} for t_{low}=1 min. $x_{ub} = 0.005, x_{ob} = 0.25$.

slope of traffic, maintain samples and *memorize* previous traffic trends are introduced and equally contribute to effective provisioning, by providing information for the algorithm to resize the reservation. It is shown that decreasing and increasing traffic require separate approaches to resizing that are inter-related and together affect the amount of resizing that takes place. Having a highly reactive approach to increasing traffic adheres to QoS requirements closely by reducing underbooking. A *cautious* approach to downsizing provisions for sudden spikes and requires fewer resizings when traffic grows again in the next peak period. Realistic traffic profiles are used in the simulations, bringing out the working dependence amongst the different entities of the algorithm. The algorithm effectively provisions bandwidth while meeting QoS constraints and, at the same time, reduces associated signaling and computational overhead.

5. Inter-Domain Path Computation

With ever increasing requirements posed by the significant advances in networking, service providers may use Traffic Engineering (TE) techniques to efficiently manage resources and provide consistent Quality of Service (QoS). Traffic Engineering in Multiprotocol Label Switching (MPLS) and Generalized Multiprotocol Label Switching (GMPLS) networks are fundamentally based on constraint-based path computation. In gist, constraint based path computation involves the pruning of links that do not satisfy constraints and subsequently using the shortest path algorithm on the resulting subgraph [13]. This process is simple and efficient when the path involves only one domain but can potentially become severely resource heavy, complex and inefficient when multiple domains are involved. To address this problem, the Path Computation Element (PCE) Working Group (WG) at the Internet Engineering Task Force (IETF) has been developing an architecture that will allow multi-domain path computation to be simple and efficient. The PCE architecture introduces a special computational entity that will cooperate with similar entities to compute the best possible path through multiple domains.

5.1 Introduction

A PCE is a node that has special path computation ability and receives path computation requests from entities known as the Path Computation Client (PCC). The PCE holds limited routing information from other domains allowing it to possibly compute better and shorter inter-domain paths than those obtained using the traditional per-domain approach. Amidst other purposes, PCEs are also being advocated for CPU intensive computations; minimal cost based TE-LSP placement; backup path computations and bandwidth protection. Alongwith the process of identifying the requirements and development of the architecture accordingly, a plethora of work is underway at the PCE WG on the new communication protocols that will make this architecture work.

5.1.1 Motivation

As is evident, a thorough performance study to justify and quantify the motivation for this new architecture is required. As the architecture is in its infancy, a detailed analysis comparing existing approaches to a PCE based approach has never been undertaken.

5.1.2 Related Work

Detailed performance analysis of the PCE Architecture has not been undertaken yet. Being a nascent architecture, several technological elements and features of the architecture are still in the process of development. RFC 4655 [48] specifies a PCE based architecture. RFC 4657 [49] covers PCE communication protocol generic requirements, and RFC 4674 [50] discusses the requirements for PCE discovery. Several working group IETF drafts are currently underway defining PCE communication protocol in different scenarios, PCE based interlayer TE, protocol extensions for OSPF and IS-IS for PCE discovery [51] and [52] were finalized in January 2008 whereas PCC-PCE communication [53], and objective function based path computation [54] are being worked upon by the community.

5.1.3 Contributions

Our research identifies the most significant performance metrics of this architecture and then presents detailed simulation results and accompanying analysis to contrast the performance of the existing approach with that of the PCE. Detailed results are published in [55] and [56].

5.2 Inter-Domain Path Computation

This section gives a brief overview of the two path computation methodologies, the per-domain and the PCE-based approach respectively. Related issues that arise during path setup are also highlighted. Both approaches use the Constrained Shortest Path First Algorithm (CSPF) [13]. CSPF first prunes all the links in the topology that do not satisfy the constraint (such as available bandwidth, delay, etc.) and then runs the Dijkstra's Shortest Path Algorithm on the resulting subgraph. The metric used for this computation is the weight assigned to the links. In the analysis undertaken for this paper, size of the TE-LSP (bandwidth to be reserved on every link of the path) is the only constraint considered. It should be noted that the path computed by CSPF always depends on the current load conditions of the network. Rerouting of TE-LSPs due to link failures, setup of other TE-LSPs prior to the current setup request are some factors that can result in the computation of different paths for the same request. Under any circumstance though, CSPF will always return the shortest path satisfying the constraint corresponding to the current loading in the network. Working of the two path computation techniques is explained in Figure 5.1(b) and Figure 5.1(c) respectively using the simple network in Figure 5.1(a).

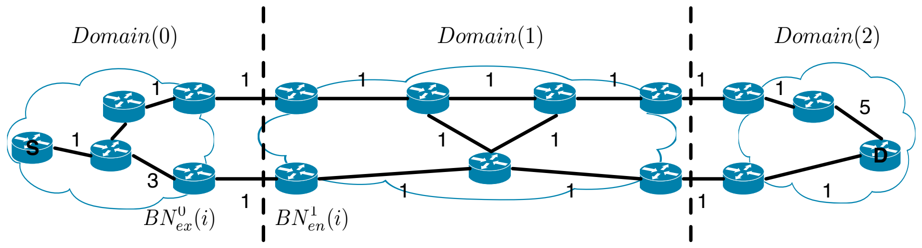
5.2.1 Backward Recursive PCE-based Computation

Path computation across domains is particularly difficult as the visibility of a head-end router is restricted only to the domain it lies in. The Backward Recursive PCE-based Computation (BRPC) utilizes multiple PCEs to compute the shortest inter-domain constraint path along a determined sequence of domains. Inherent in the PCE architecture, this technique also preserves confidentiality across domains, an important requirement when domains are managed by different Service Providers [57]. The BRPC procedure is *backward*, as the path is computed in a reverse fashion from the destination area to the source area. It is *recursive* since the same basic sequence of steps is repeated for every intermediate domain lying between the source and the destination. Every domain has certain "entry" boundary nodes (BN) into the domain and "exit" BNs out of the domain (into other domains). The following notation is defined for a clear explanation of the BRPC procedure.

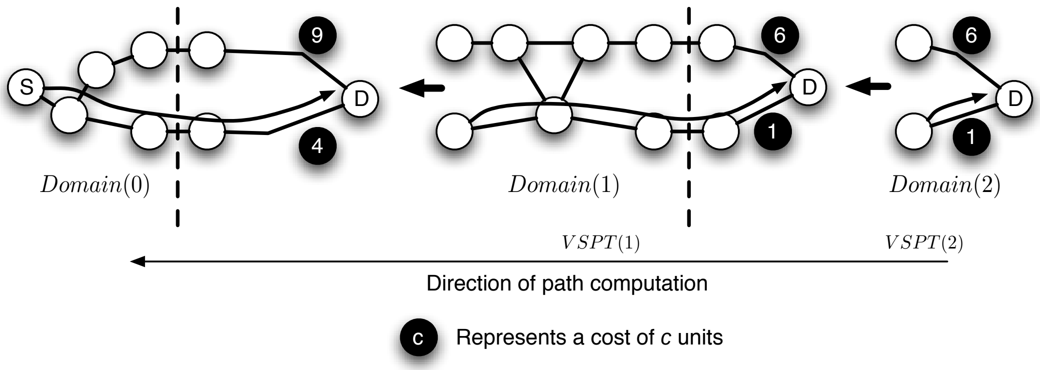
The entry BN of domain i joins domains $i-1$ and i. Let the source and destination domains be denoted by 0 and n respectively and there be $n-1$ intermediate domains $1, \cdots, n-1$. For a domain i, the k^{th} entry BN is denoted by $BN^k_{en}(i)$ and the k^{th} exit BN is denoted by $BN^k_{ex}(i)$. BRPC uses the concept of a Virtual Shortest Path Tree (VSPT). $VSPT(i)$ denotes the Multipoint to Point (MP2P) tree formed by the set of constrained shortest paths from the entry BNs of domain i to the destination router. Each link of tree $VSPT(i)$ represents a shortest path. The BRPC procedure is as follows. For the destination domain n, $VSPT(n)$ is computed from the list of shortest paths to all the entry BNs from the destination node. This is illustrated in Figure 5.1(b). $VSPT(n)$ is then sent to the PCEs in the previous domain (domain $n-1$ in this case) where it is concatenated to their TEDs (as links). This TED is then used by the PCEs in domain $n-1$ to compute $VSPT(n-1)$. This sequence is repeated until the source domain and subsequently, the head-end router of the TE-LSP is reached, as shown in Figure 5.1(b).

5.2.2 Per-Domain Path Computation

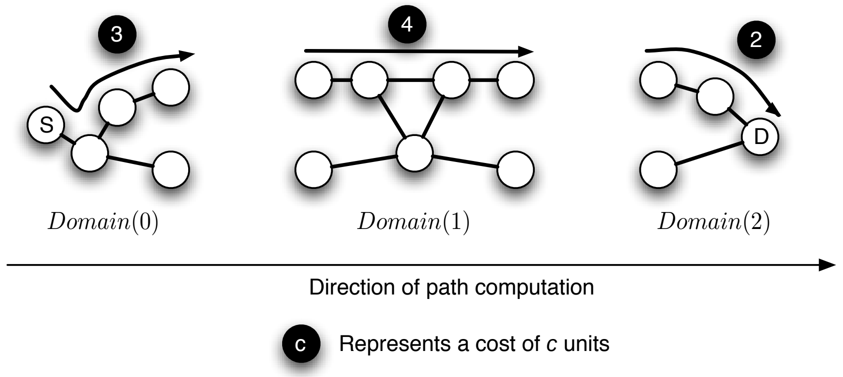
In contrast to the BRPC procedure, the per-domain path computation technique involves the computation of individual *path segments* in every intermediate domain



(a) Example Network with associated link costs.



(b) VSPTs formed by the BRPC procedure and their costs.



(c) Per-domain path computation and costs of the path segments.

Figure 5.1: Inter-domain path computation.

without the sharing of any path information from other domains. The complete path for a TE-LSP is obtained by concatenating the path segments that are computed for every domain i. Figure 5.1(c) illustrates a simple scenario of per-domain path computation. The head-end router computes the first path segment by finding the shortest path to the *nearest* exit BN. The second path segment is then computed for the second domain by this BN to the next nearest exit BN. This method does not necessarily always yield the shortest path across domains.

5.2.3 Path Setup

Several setup issues arise after a path has been computed and the headend router of the tunnel tries to setup reservation along the route using the Resource Reservation Protocol with Traffic Engineering extensions (RSVP-TE) [7]. They result from outdated TEDs on any or all of the routers involved in the path computation process. The TED is maintained on every TE enabled router and holds information about several attributes of the links in its domain of existence. Path computation based on CSPF uses the TED and results in a path that satisfies the constraints put on one or a combination of these attributes. If the TED on the router computing the path is outdated, a Call Admission Control (CAC) failure occurs at the entry point router of the link corresponding to the outdated information. In such a situation, if the underlying routing protocol is Open Shortest Path First with TE extensions (OSPF-TE), a Link State Advertisement (LSA) is flooded, or if it is Intermediate System-Intermediate System (IS-IS), a Link State Packet (LSP) is flooded. This flooding updates the TEDs of all the routers with the most updated information with respect to the concerned link and is restricted only to the domain in which the CAC failure occurred. The per-domain and the PCE approach proceed differently in this situation. Details of LSA flooding and their mechanisms are explained in great detail in [13].

PCE based approach. The BRPC method [57] is used to compute the VSPT between $BN_{en}^{k}(i)$ and the destination. When a CAC failure occurs, a new VSPT for that domain is computed and path setup is retried in the domain. The number of CAC failures that take place with the PCE approach is *at most* equal to that of the per-domain approach.

Per-Domain approach. Here, a *crankback* [58] is used to report a setup failure to the router that is trying to setup the path. For example, without any loss of generality, let the CAC failure take place in domain i when $BN_{en}^{k}(i)$ was setting up a path to $BN_{ex}^{j}(i)$. Upon receiving the failure information, $BN_{en}^{k}(i)$ tries to find a new shortest path to the next closest $BN_{ex}^{(j+1)}(i)$. This process continues till no path can be found from $BN_{en}^{k}(i)$ to any $BN_{ex}(i)$. In this situation, the setup failure information from $BN_{en}^{k}(i)$ is sent upstream to $BN_{en}^{k}(i-1)$ in domain $i-1$ using a crankback. $BN_{en}^{k}(i-1)$ then selects the next $BN_{ex}^{(j+1)}(i-1)$ to enter domain i. Information about setup failures can propagate using the crankback to the source domain containing the headend router. This cycle repeats till a path can be setup or no path is found after exhausting all possibilities. It should be noted that crankback signaling is associated only with the per-domain approach to path computation.

5.3 Simulation Results

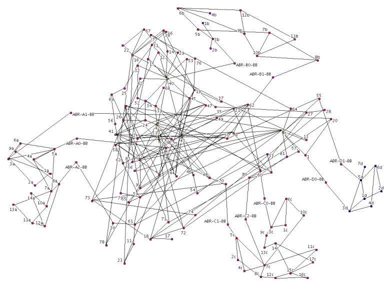
5.3.1 Simulation Setup

1. **Topology Description.** To obtain meaningful results applicable to present day provider topologies, simulations have been run on two realistic topologies representative of current service provider deployments. They consists of a large *back bone* area to which four smaller areas are connected. For the first topology named MESH-CORE, a highly connected backbone was obtained from RocketFuel [40]. The second topology has a symmetrical backbone and is called

SYM-CORE. The four connected smaller areas are obtained from [41]. Details of the topologies are shown in Table 5.1 alongwith their layout in Figure 5.2. All TE-LSPs setup on this network have their source and destinations in different areas and all of them need to traverse the backbone network. Table 5.1 also shows the number of TE-LSPs that have their sources in the corresponding areas along with their size distribution. The LSA updates are restricted only to the domain in which they originate. The nodes have a path computation engine that operates on the routing information in the TED. In the per-domain approach, routing information is limited only to the domain of existence. In the PCE approach, routing information is passed on to the domains according to the methodology outlined in the BRPC procedure [57]. Link failures are induced in all the areas. Each link in a domain can fail independently with a mean failure time of 24 hours and be restored with a mean restore time of 15 minutes. Both inter-failure and inter-restore times are uniformly distributed. When a link fails, an LSA is flooded by the adjacent routers of the link and all the TEDs of the routers lying in the same domain of existence are updated. When a link on the path of a TE-LSP fails, reservation is removed along the path and the head-end router tries to setup the TE-LSP around the failure. No attempt to re-optimize the path of a TE-LSP is made when a link is restored. The links that join two domains never fail. This step has been taken to concentrate only on how link failures within domains affect the performance.

5.3.2 Performance Analysis

In the figures, ‘PD-Setup’ and ‘PCE-Setup’ represent results corresponding to the initial setting up of TE-LSPs on an empty network using the per-domain and the PCE approach respectively. Similarly, ‘PD-Failure’ and ‘PCE-Failure’ denote the results


(a) Topology for MESH-CORE.

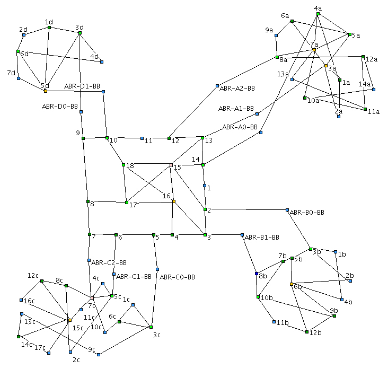

(b) Topology for SYM-CORE.

Figure 5.2: Topologies used in the simulations.

under a link failure scenario. A period of one week is simulated.

1. **Path Cost.**

   The inter-domain path cost is the sum of costs of individual path segments in each domain. We study the average and the maximum path costs of the inter-domain TE-LSPs setup.

   Observation. Figures 5.3 and 5.4 show the distribution of the average path cost of the TE-LSPs for MESH-CORE and SYM-CORE, respectively. During initial setup, roughly 40% of TE-LSPs for MESH-CORE and 70% of TE-LSPs

Table 5.1: Details of all the areas used to create the two topologies

Domain Description					TE-LSP Size	
Domain Name	# of nodes	# of links	OC48 links	OC192 links	[0,20) Mbps	[20,100] Mbps
D_1	17	24	18	6	125	368
D_2	14	17	12	5	76	186
D_3	19	26	20	6	14	20
D_4	9	12	9	3	7	18
MESH Backbone	83	167	132	35	0	0
SYM Backbone	29	37	26	11	0	0

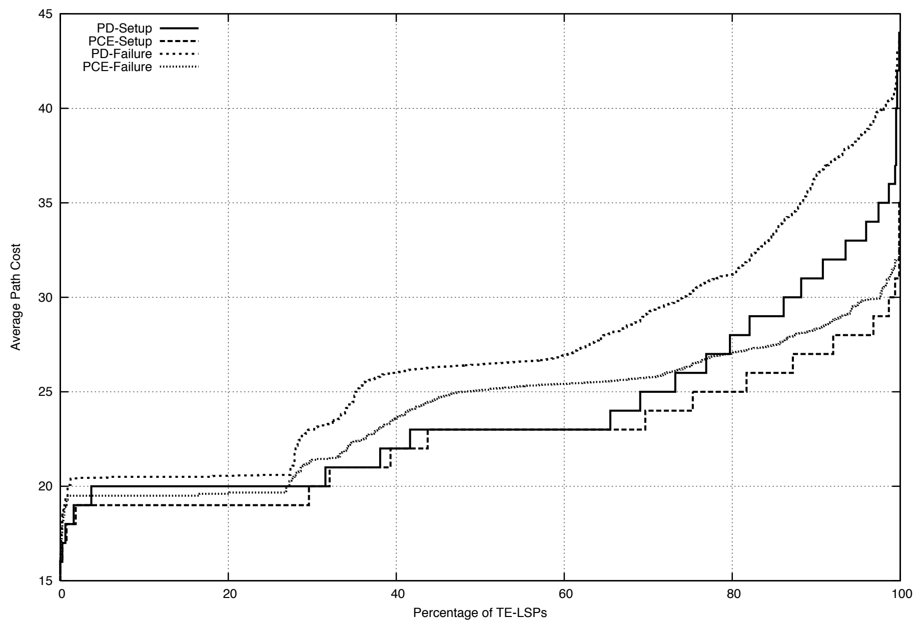


Figure 5.3: Average path cost distribution: MESHCORE.

for SYM-CORE have path costs greater with the per-domain approach (PD-Setup) than with the PCE approach (PCE-Setup).

Hypothesis. This behavior arises from the ability of the BRPC procedure to select the shortest available paths that satisfy the constraints. Since the per-domain approach to path computation is undertaken in stages where every entry BN to a domain computes the path in the corresponding domain, the most optimal route is not always found. When failures start to take place in the network, TE-LSPs are rerouted over different paths resulting in path costs that are different from the initial costs. PD-Failure and PCE-Failure in Figures 5.3 and 5.4 show the distribution of the average path costs that the TE-LSPs have over the duration of the simulation with link failures occurring. Similarly, the average path costs with the per-domain approach are much higher

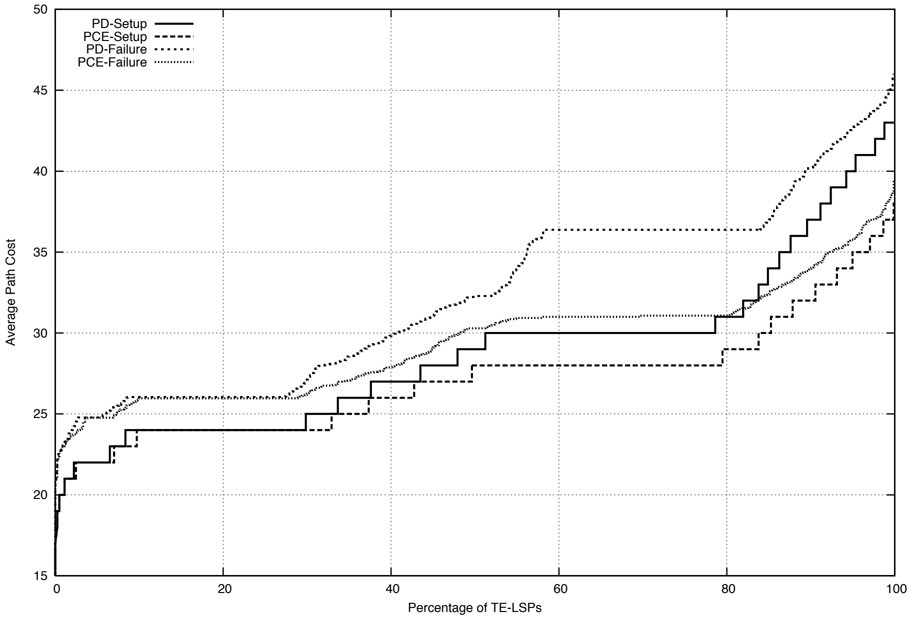


Figure 5.4: Average path cost distribution: SYMCORE.

than the PCE approach when link failures occur. Figures 5.5 and 5.6 show similar trends and present the maximum path costs for a TE-LSP for the two topologies, respectively. It can be seen that with per-domain path computation, the maximum path costs are larger for 30% and 100% of the TE-LSPs for MESH-CORE and SYM-CORE, respectively.

2. **Crankback/Setup Delay.**

   Due to crankbacks that take place in the per-domain approach of path computation, TE-LSP setup time is significantly increased. This could lead to QoS requirements not being met, especially during failures when rerouting needs to be quick in order to keep traffic disruption to a minimum. Since crankbacks do not take place during path computation with a PCE, setup delays are significantly reduced.

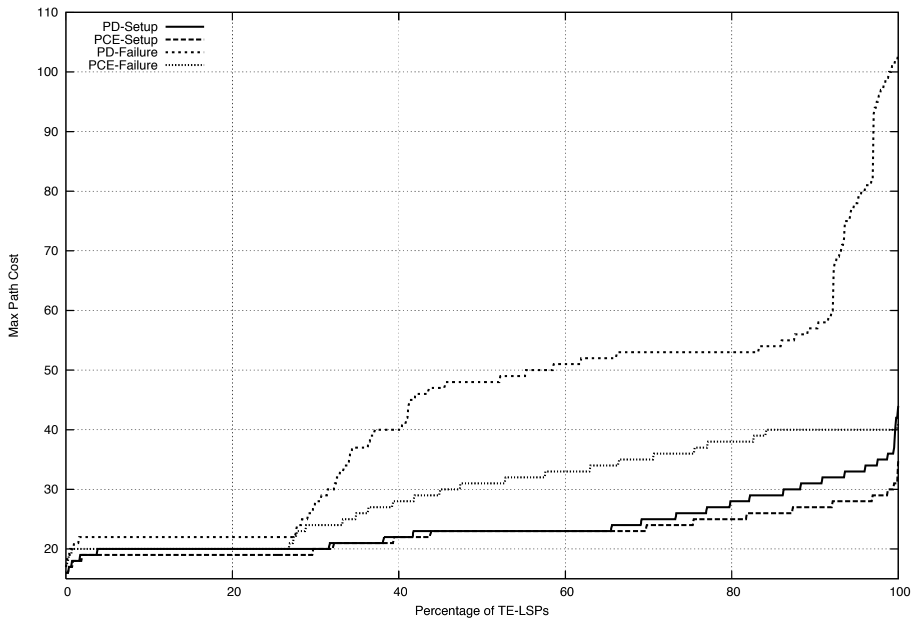


Figure 5.5: Maximum path cost distribution: MESHCORE.

Observation. Figures 5.7 and 5.8 show the distributions of the number of crankbacks that took place during the setup of the corresponding TE-LSPs for MESH-CORE and SYM-CORE, respectively. It can be seen that all crankbacks occurred when failures were taking place in the networks.

Hypothesis. Information regarding failures never propagate across domains since the corresponding LSAs are restricted only to the domain in which they originate. As a result, BNs of one domain do not have information of a failure in the next domain leading to a crankback when a route cannot be found in the next domain. Figures 5.9 and 5.10 illustrate the *proportional* setup delays experienced by the TE-LSPs due to crankbacks for the two topologies. It can be observed that for a large proportion of the TE-LSPs, the setup delays arising out of crankbacks is very large possibly proving to be very detrimental to QoS requirements. The large delays arise out of the crankback signaling that needs

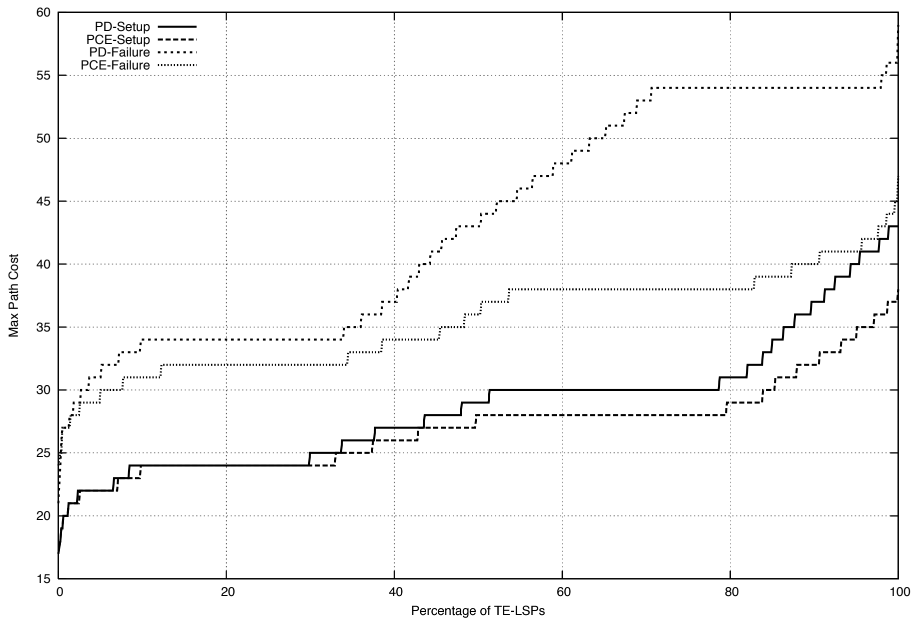


Figure 5.6: Maximum path cost distribution: SYMCORE.

to propagate back and forth from the exit BN of a domain to its entry BN. More crankbacks occur for SYM-CORE as compared to MESH-CORE as it is a very ‘restricted’ and ‘constrained’ network in terms of connectivity. This causes a lack of routes and often several cycles of crankback signaling are required to find a route.

3. **CAC Failures.**

   As discussed in the previous sections, CAC failures occur either due to an outdated TED or when a route cannot be found from the selected entry BN.

   Observation. Figures 5.11 and 5.12 shows the distribution of the total number of CAC failures experienced by the TE-LSPs during setup. About 38% and 55% of TE-LSPs for MESH-CORE and SYM-CORE, respectively, experience a CAC failures with per-domain path computation when link failures take place

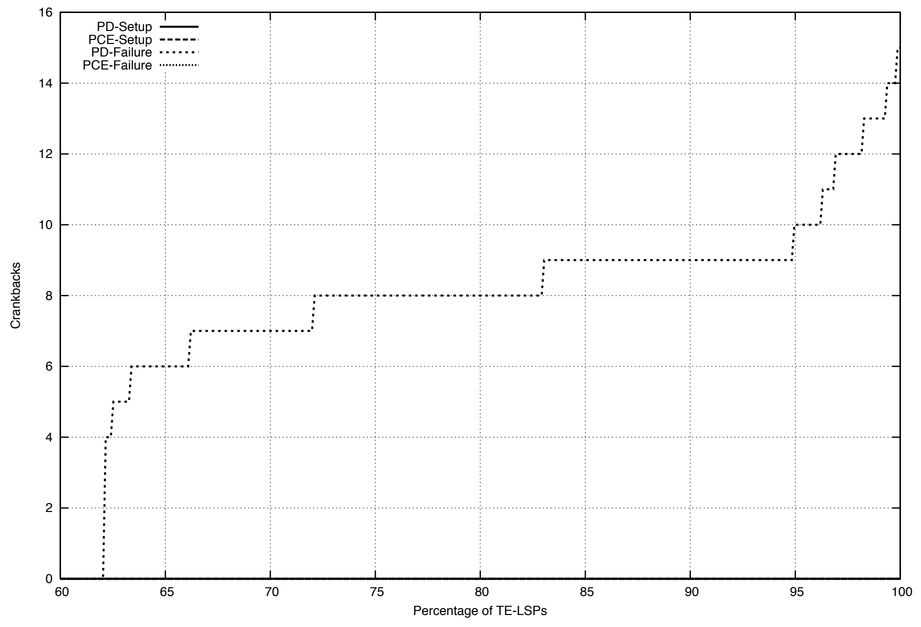


Figure 5.7: Distribution of number of crankback: MESHCORE.

in the network. In contrast, only about 3% of the TE-LSPs experience CAC failures with the PCE method.

Hypothesis. CAC failures experienced with the PCE correspond only to the TEDs being out of date. This is because with a PCE deployment, a BN that does not have a route to the destination is never selected by the BRPC procedure. This is not the case in the per-domain approach and hence there are more CAC failures associated with it.

4. **Failed TE-LSPs/Bandwidth on Link Failures.**

   Figures 5.14 and 5.13 show the number of TE-LSPs and the associated required bandwidth that fail to find a route when link failures are taking place in the topologies.

   Observation. For MESH-CORE, with the per-domain approach, 395 TE-LSPs failed to find a path corresponding to 1612 Mbps of bandwidth. For PCE,

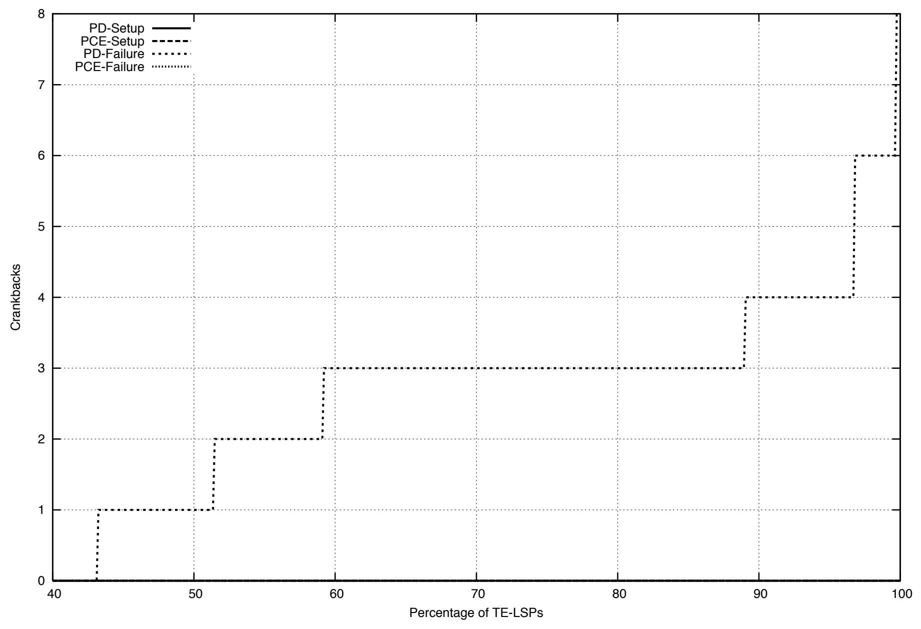


Figure 5.8: Distribution of number of crankback: SYMCORE.

this number is lesser at 374 corresponding to 1546 Mbps of bandwidth. For SYM-CORE, with the per-domain approach, 434 TE-LSPs fail to find a route corresponding to 1893 Mbps of bandwidth. With the PCE approach, only 192 TE-LSPs fail to find a route, corresponding to 895 Mbps of bandwidth.

Hypothesis. It is clearly visible that the PCE allows more TE-LSPs to find a route thus leading to better performance during link failures. This improve in performance arises due to the difference in path computation procedure too. BRPC always includes all links that satisfy the constraint in the VSPT and builds the shortest path from the destination. Since the per-domain approach does not have all information about links in other domains, after several TE-LSPs are setup, available bandwidth is left over in fragments and not sufficient to route all requests, thereby resulting in several setup failures.

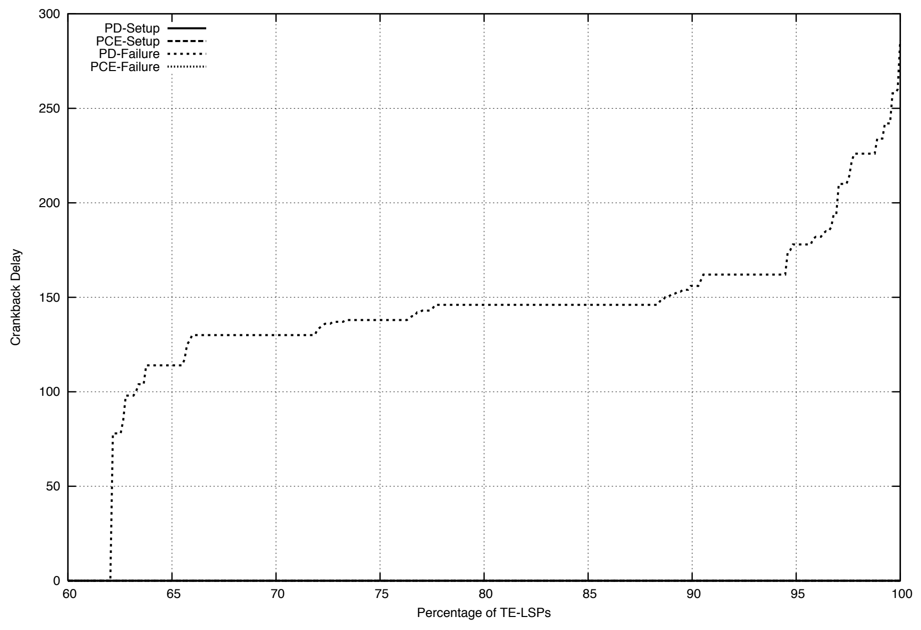


Figure 5.9: Distribution of proportional setup delay: MESHCORE.

5.4 Summary and Future Directions

In this article, a performance study with respect to several critical metrics between a PCE based and a per domain path computation deployment was presented. The PCE based approach showed better results when compared to the per domain path computation approach with the selected performance metrics. Keeping in mind that all evaluation metrics are topology and traffic dependent, this study used two significantly varied but representative topologies where the difference in performance improvement was also illustrated. Although many more nuances are left to be uncovered and analyzed in future work, novel preliminary analysis and insight into factors governing inter-domain path computation is brought to light. An overview of the current status of the IETF PCE working group was also given. The working group is currently refining drafts, as well as awaiting on deployment experiences. This should bring even further insights on the benefits of the PCE architecture for current network

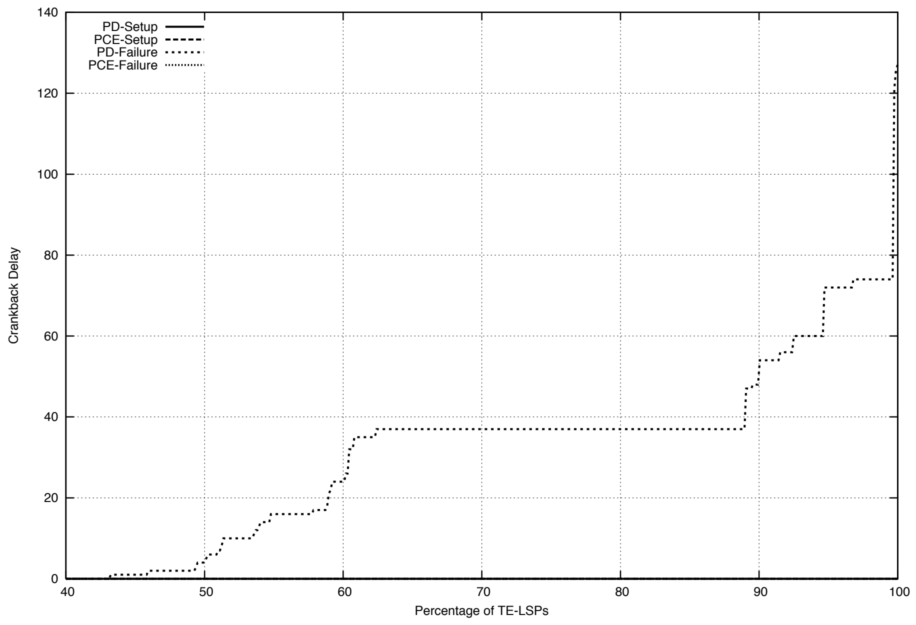


Figure 5.10: Distribution of proportional setup delay: SYMCORE.

systems.

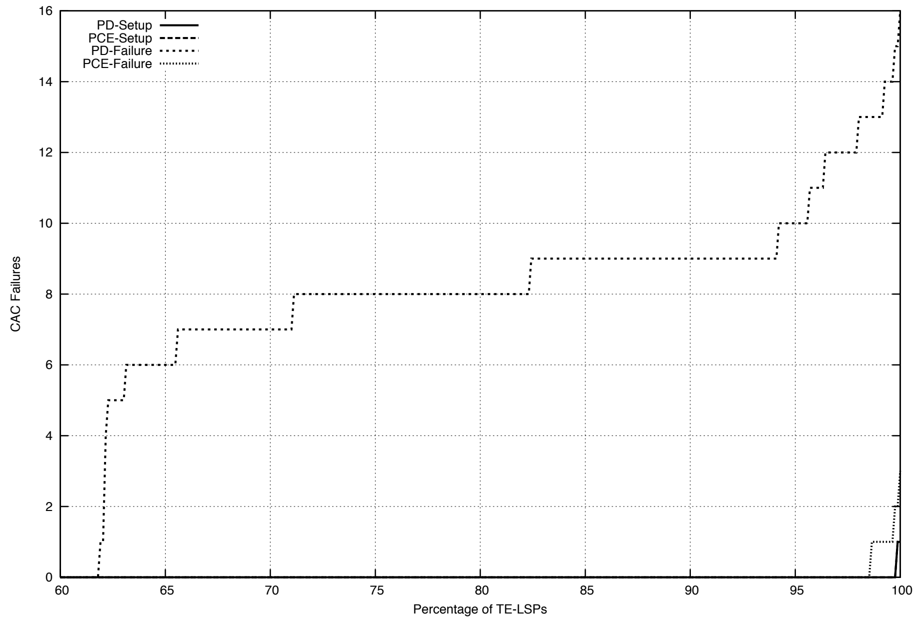


Figure 5.11: Distribution of number of CAC rejections: MESHCORE.

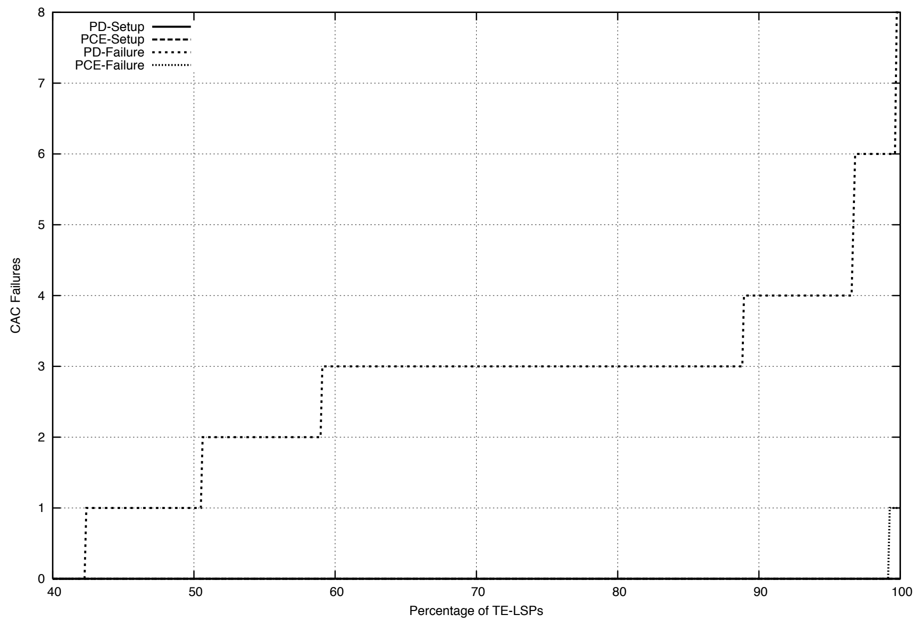


Figure 5.12: Distribution of number of CAC rejections: SYMCORE.

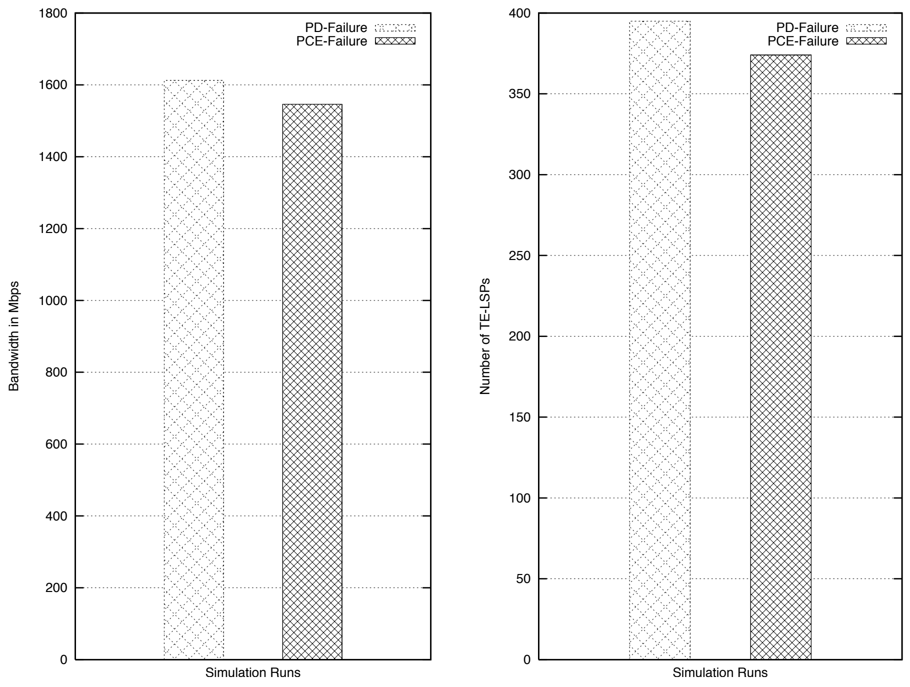


Figure 5.13: Number of setup failures and corresponding bandwidth: MESHCORE.

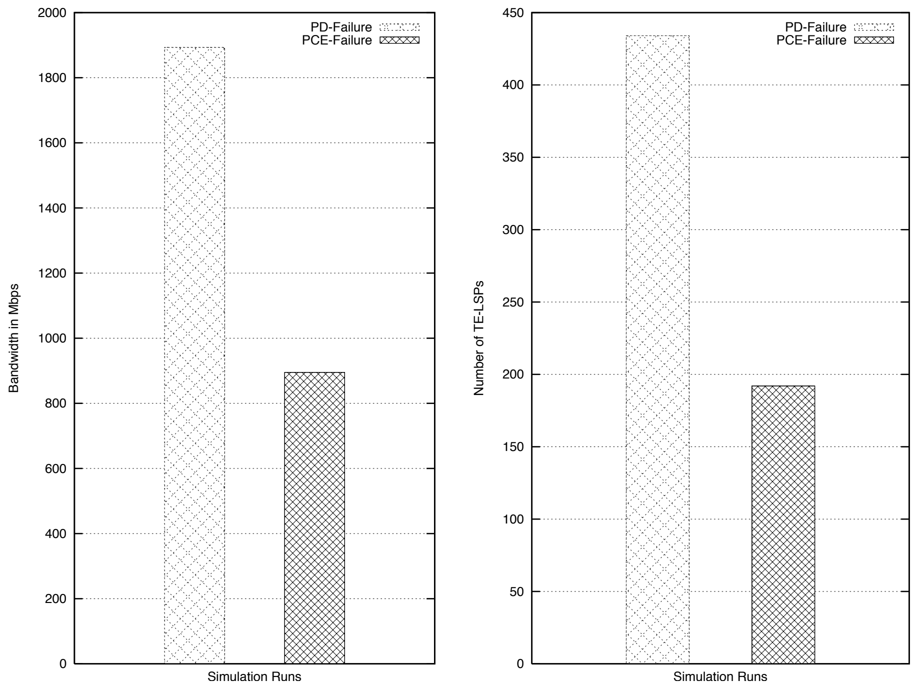


Figure 5.14: Number of setup failures and corresponding bandwidth: SYMCORE.

6. Programmable Traffic Engineering and Analysis Testbed

6.1 Introduction

In this section, we discuss the architecture of MuTANT: A Multi-protocol label switched Traffic engineering and ANalysis Testbed.

The hardware setup for a testbed is not sufficient alone to be a platform for effective testing and analysis. A software for configuring and observing the performance of the network when a new mechanism is being deployed is equally important. For long, Application Programming Interfaces (APIs) have been the source code interface to operating systems and libraries through which softwares and services have been developed. APIs have either been licensed by their owners or made completely open to the community. Either way, API availability has always resulted in the advancement of technology and introduction of new services that in turn have lead to the improvement and development of the API that initiated the trend. Development in the field of computer networks has followed a different path. The majority of services and mechanisms that exist in the link, network and transport layer are proprietary, making their progress owner dependent. Though key protocol standards are always open and converged upon through a detailed IETF process, ownership of existing services and the option to introduce new services based on these protocols are almost always exclusive to vendors. With the advent of network virtualization coupled with the flexibility gained by programmable network layers, significant headway has been made into experimental validation of newly proposed ideas. Implementation and testing of these ideas on vendor provided hardware without hindrance is still far from reality. With very little or no access to the code that runs on routers, it is almost impossible to obtain any performance measure on vendor hardware. Evaluation thus suffers from the intrinsic drawback of not being tested on real equipment and underlying protocols, often resulting in side effects that are overlooked and prevent deployment. Proposed new mechanisms almost always rely on information gathered from the state of the network, traversing traffic and knowledge of other network dynamics. The best method to *completely* test the effectiveness of new mechanisms is to implement it in the operating system of the routers. Due to proprietary content, non-disclosure issues limit such a practice thereby limiting the introduction of new functionality to the responsibility of the vendor.

This paper presents a model API that allows router configurations and provides the platform to emulate the deployment behavior of new network mechanisms. The motivation is open the tried and tested vendor platforms to the research community for experimentation. It is hoped that this will lead to the introduction and proposition of novel research solutions that will be more robust and, in general, speed up and encourage development in the networking field.

6.1.1 Motivation

1. **Testbed Setup** Research in Multi-Protocol Label Switching (MPLS) based Traffic Engineering (TE) has been prolific over the past few years yielding new techniques and insight into traffic management. The proposed new mechanisms almost always rely on information gathered from the state of the network, traversing traffic and other dynamics. Their performance is usually evaluated using detailed simulations. This evaluation method suffers from the intrinsic drawback of not being tested on real equipment and underlying protocols, often resulting in side effects that are overlooked preventing deployment. The best method to *completely* test the effectiveness of new TE mechanisms is to

implement it in the operating system of the routers. Such a process is limited by non-diclosure issues and even if possible, is tedious, involving several cycles of testing and validation before deployment. An alternative is thus required by the research community.

2. **Software API** Several programmable platforms are under development currently. Efforts such as XORP, Quagga and Zebra, provide the community with opportunities to test the coexistence of custom built protocols with standard deployed protocols. This methodology is limited by the subset of protocols and features that are currently available in the corresponding platform. Small feature sets or services with a light footprint would require the complete implementation of underlying protocols if absent. This slows down development and incurs a large overhead in terms of effort. Open platforms will also need a lot of time to mature for them to gain the confidence of service providers and enterprises to be considered a deplpyment option. As faults and bugs from protocol implementation will take place over large periods of time, providers will not be willing to use them as a platform for introduction of new services. It is thus a motivation to try and introduce a functionality mechanism that will be able to interface with existing vendor hardware with proven deployment track records. From the perspective of community developers, there are several advantages of the availability of APIs for vendor libraries. Amongst others, they include:

    - Selective Usage. Only parts of the API can be used to develop mechanisms that require them. Selective API usage will prevent code bloat and thus keep implementations simple and lightweight.
    - Layer Independence. Using selective API features will prevent a developer from worrying about correct working of underlying layers. For example, if Traffic Engineering API is used only for traffic management, the developer

can safely assume reliable Network Layer functioning of the vendor operating system. Not only does this encourage modular code, it also saves time by allowing one to concentrate only on specific layers.

- Code Sharing. Modules developed based on open APIs will allow sharing of code. This will speed development and result in improved, tested and thus more robust mechanisms.

From the perspective of vendors, the primary goal is to keep proprietary technology insulated from the community and yet provide enough means and flexibility to implement new design ideas. Figure 6.1 illustrates the idea of three planes that could exist. The Hardware Plane consists of the network equipment marketed by vendors. The API Plane consists of in-house APIs used by the vendors to introduce new features and updates. The Community Plane consists of feature implementations developed by the community using the vendor controlled API. Currently, the Hardware and the API planes are not accessible by the community.

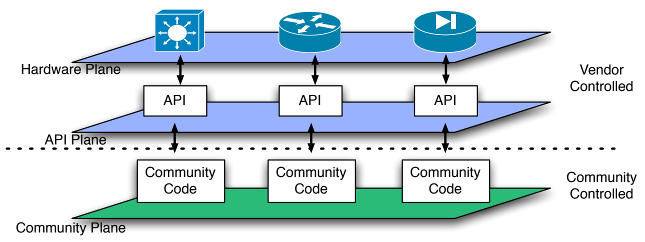


Figure 6.1: Planes in the programmability idea.

6.1.2 Related Work

Being a relatively nascent concept, opinions, debates and discussions on the exact significance and relevance of network programmability are many. In [59] the authors present an overview of what “Network Programmability” signifies, discuss the scenarios where they could be relevant and highlight new services that it can enable. The authors also discuss key issues that limit the easy introduction of new features into vendor based hardware and point out that open platforms supporting programmability will change this trend. In prior work [60] and [61], the authors also highlight the fact that interoperation of new programmable elements and existing services and hardware is a very important requirement. This requirement also proves to be a one of the motivating factors for the development of the proposed API.

In [62], the authors note that absence of support for custom implementation of new services and configurations in vendor provided hardware and present PRESTO, a configuration management platform. It also allows the deployment of new services based on a hybrid configuration language that is introduced. This language is used to define configuration rules that wrap around standard device configuration statements. Its architecture contains a data repository that contains router state/configuration information that is used in standard templates for context and functional substitution. This work brought to light two key issues. Firstly, it is very difficult for vendor provided hardware to be customizable beyond features that the underlying operating system provides. Secondly, special languages, APIs can be developed to provide functionality beyond what is provided by the router and need to be user friendly and easily extendable. Such an architecture would make it easier in the future for the community to make use of different hardware platforms, features and new protocols.

6.1.3 Contributions

We build a testbed comprising of 10 Cisco routers and 20 nodes comprising of a combination of Sun Workstations and PCs. A Java based API is also developed and used to emulate the Trend Based Bandwidth Reservation mechanism in Chapter 4, Section 4.5.

6.2 MuTANT: A MUlti-protocol label switched Traffic engineering and ANalysis Testbed

In this section, we discuss the design aspects and implementation of the testbed.

6.2.1 Design Aspects

Numerous aspects of a network need to be kept in mind while designing the testbed. Logically, they can be categorized as:

1. **Network Entities.** This consists of all the hardware and software aspects that form the physical and logical network. Hardware network entities consist of the routers and the physical links that link them. Software entities consists of the protocol level implementations in the hardware that perform the routing, resource reservations and numerous such software level tasks. In the testbed, the network entities will be configured just as standard equipment on networks are. Routing and resource reservation protocols will be deployed and will be oblivious of the background measurement or control being undertaken. This will allow the study and analysis of all aspects of the network once the TE mechanism being tested reconfigures elements.

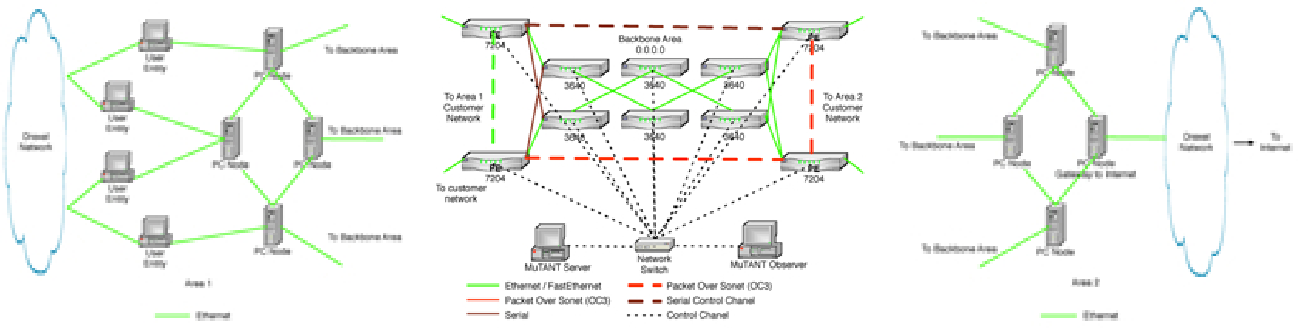
2. **Control Entities.** This consists of hardware and software that is used to configure the network entities corresponding to the TE mechanisms being tested.

Hardware can consist of commodity level PCs and physical links connecting them to special points in the network. Software on the control entities consists of widely available software and custom built software that in tandem undertake the task of measurement and reconfiguration.

3. **User Entities.** This is a logical aspect and represents usage patterns and utilization of the network.

6.2.2 Architecture and Implementation

This section describes the architecture of the testbed and how the design aspects have been implemented.



(a) Area 1 (b) Backbone (c) Area 2

Figure 6.2: Testbed setup.

1. **Network Entities.** The network entities consist of Cisco routers and 'PC Nodes'. Together they have been connected to form a representation of how a large corporate customer connects physically separated areas using VPN over a provider-based backbone network.

(a) **Cisco Routers.** The testbed consists of 10 Cisco Routers. They include 4 Cisco 7204VXR routers and 6 3640 routers. They are connected so as to provide multiple paths between any two routers in the network. The 7204VXR routers support MPLS-TE and form the "provider-edge" or 'PE' routers of the backbone while the 3640s form the 'provider' or 'P' routers of the backbone. OSPFv2 is the underlying IGP.

(b) **PC Nodes.** The PC Nodes form two separate areas of a customer and are connected to the backbone. One of the two areas acts as the source where all traffic originates and the other area acts as a sink for which all traffic is destined. Area 1 acts as the source and consists of 4 PCs. They are equipped with 450Mhz Pentium II processors and 256MB of RAM running Ubuntu 7.04 (Linux Kernel 2.6.20) [63]. Area 2 consists of 4 Sun Microsystems workstations equipped with 650Mhz UltraSparc processors and 512MB of RAM also running the same version of Ubuntu Linux. All PC Nodes are configured as OSPF routers using the eXtensible Open Router Platform (XORP) [64], [65]. XORP is an open source router platform that implements routing protocols for IPv4 and IPv6. It allows a PC to be setup and configured as a router. Since all protocols are implemented according to RFC standards, there is no issue when connecting the two areas with the backbone areas. Schematics of how XORP is setup on the PC Nodes is shown in Figure 6.4(b)

2. **Control Entities.** The control logic for all new TE mechanisms that are developed and need to be tested lies on the Control Entities. The primary control entity is called the MuTANT server. It consists of three modules, the *State Sampler*, the *Configuration Modifier*, the *Traffic Engineering Module* and the *MuTANT Observer*. The setup of Network Entities and Control Entities

are shown in Figure 6.2 and Figure 6.3.

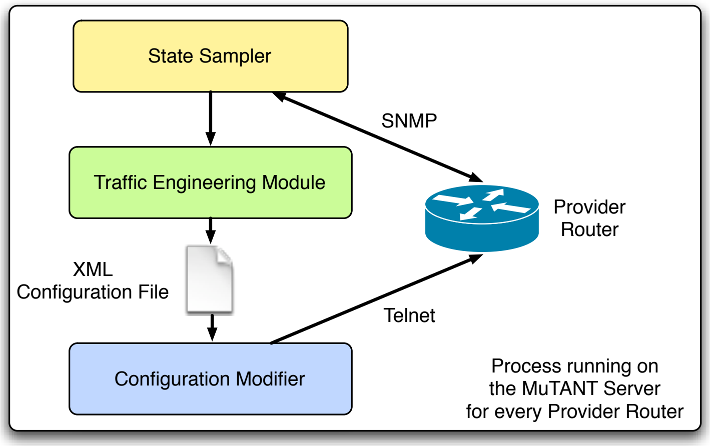

State Sampler
SNMP
Traffic Engineering Module
Provider Router
XML Configuration File
Telnet
Configuration Modifier
Process running on the MuTANT Server for every Provider Router

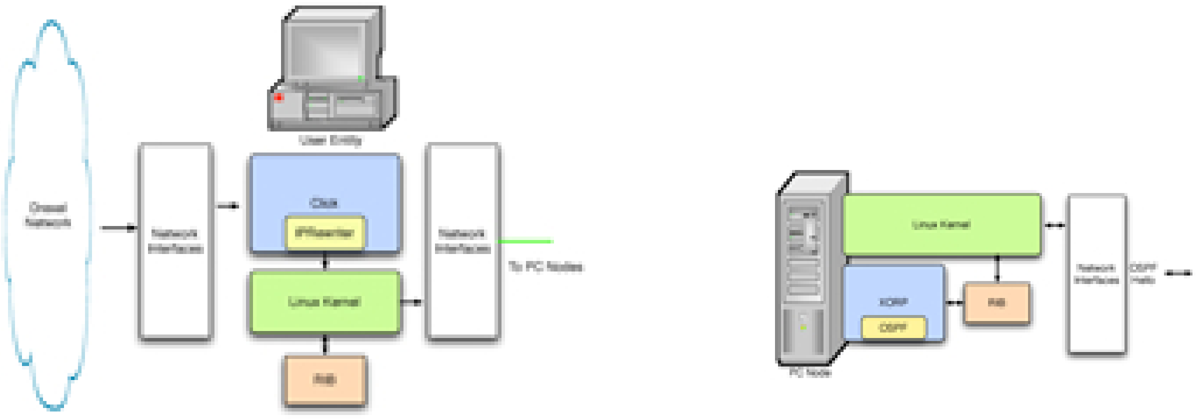
Figure 6.3: MuTANT server architecture.

(a) **State Sampler.** The State Sampler periodically obtains state information from the routers required by the new TE mechanism to act upon. This information can range from MPLS specific information, such as traffic flowing through LSPs, LSP routes, to complete TE/IGP topology that is visible to the router. A Java based SNMP application has been developed to achieve this. The Net-SNMP [66] applications are used from within the Java application. Several policies for sampling are used depending on what information is gathered. Topology information is gathered using SNMP Traps [67] only when there is a new LSA flooded whereas traffic flowing through a tunnel is sampled every 5 minutes. All information is gathered over a separate channel. Router states are gathered using the Multi Router Traffic Grapher (MRTG) [68] and stored using the Round Robin Database Tool (RRDtool) [69].

(b) **Traffic Engineering Module.** This is the module that implements the new Traffic Engineering mechanism. There is no limitation of how it is implemented. Common programming languages such as C, Java or PERL can be used. It gathers the information it needs from the State Sampler and acts upon it according to its functionality. It gives as output, the tasks needed to be carried out as a configuration file. The configuration file needs to be in XML and follows a standard format that is easy to understand and parse.

(c) **Configuration Modifier.** The *Configuration Modifier* parses the configuration file generated by the TE module and then contacts the router to change its configuration. Changing configurations may involve setting up new LSPs, changing the route or the reservation value for LSPs or several other aspects. The configuration files are written in XML. This module is written completely in Java and contains two functionalities. First, it parses the XML file using the Java Xerces API [70], then it uses Telnet to contact the corresponding router to implement the configuration change. The Apache Commons Net API [71] is used for the Telnet functionality.

(d) **MuTANT Observer.** The task of the MuTANT observer is to study and capture the metrics that are measuring the performance of the TE mechanism. Every network element can be sampled to get the desired information. The particular property to be measured on a certain device is specified in an XML file which is read using a Java application. It then gets the desired information using either SNMP or by using Telnet. The MuTANT observer can be collocated with the MuTANT Server. In our setup, the MuTANT observer is a separate PC running Linux 2.6.



(a) Forwarding using the Click Modular Router (b) Routing on a PC Node

Figure 6.4: User entity setup and PC node.

3. **User Entities.** User entities is a logical abstraction of the utilization of the network. In order to make the testbed flexible such that different TE mechanisms can be deployed for various kinds of user traffic, the fundamental requirement is having access to such traffic. This can be achieved in two ways using the Click Modular Router [72] [73]. Click is a new software architecture for building flexible and configurable routers. To generate a high volume of traffic having realistic characteristics, the Click modular router is used to forward traffic from the Drexel network onto the testbed. Using the *IPRewriter* element [74], the source and the destination IP addresses of all packets visible on the particular Drexel LAN segment is changed to that of a source and destination in Area 1 and Area 2 of the testbed respectively. Schematically, this is shown in Figure 6.4(a). This causes all the packets to traverse the backbone area and be subject to the control entities. A sample profile is shown in Figure 6.5.

   Though realistic and in high volume, this traffic does not allow QoS studies to take place. The approach to using real and relevant user traffic on the testbed

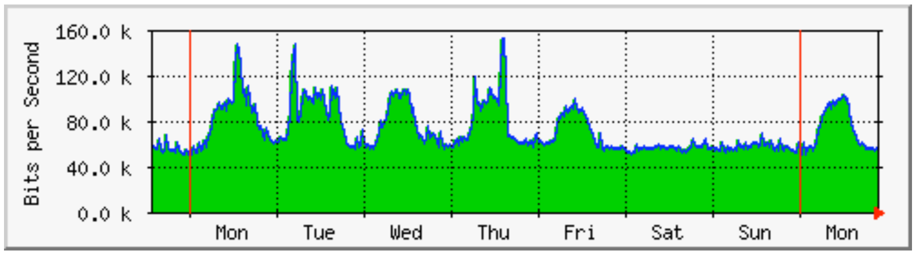


Figure 6.5: Weekly profile of traffic being forwarded through the backbone area of the testbed.

is also implemented using the Click Modular router. In this setup, one of the PCs in Area 2 acts as a gateway to the Internet. All users using the Internet lie in Area 1 and know of this gateway. The gateway in Area 2 uses the Click Modular router to implement a Network Address Port Translator (NAPT). The source IP address of any incoming packet from Area 1 are rewritten to that of the gateway, assigned a new port and sent out into the Internet. Reverse translation then takes place once packets from the Internet come back towards the clients in Area 1. This idea is similar to [75] and allows realistic user traffic to traverse the backbone area and also gives greater configurability of the PC Nodes according to different kinds of user traffic. Figure 6.6 shows a picture of the testbed in its current state.

6.3 API Architecture

This API is developed with the motivation of illustrating that new features can be tested on vendor equipment. Translating API based code to running configurations on the routers require two steps shown in Figure 6.7. After a programmer develops a program containing a sequence of methods and operations that will configure the routers and underlying protocols to for a network of his choice, the following takes place:

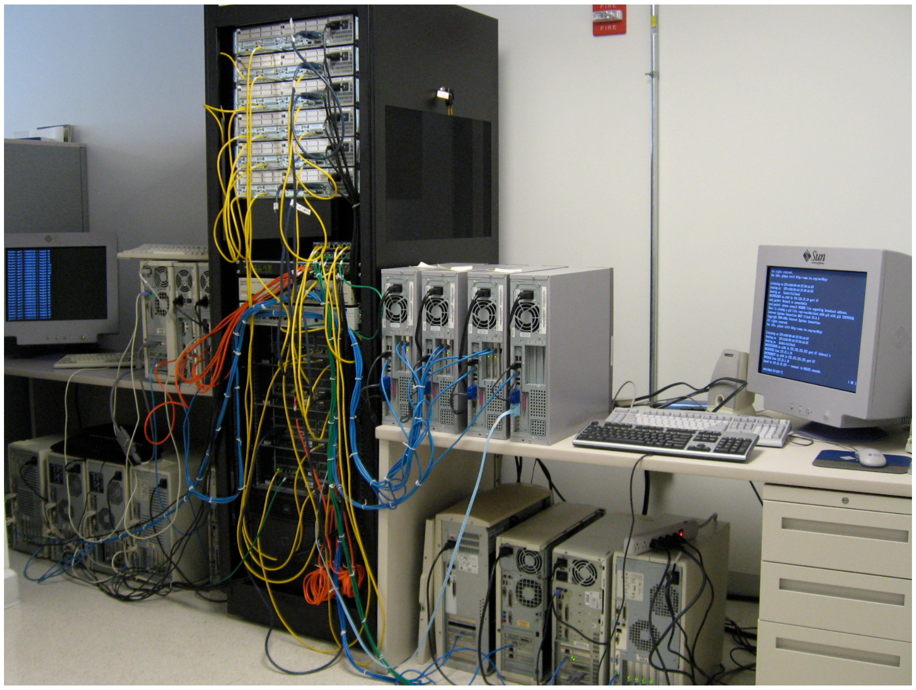


Figure 6.6: Testbed setup.

1. **Step 1.** The API internally then translates this sequence to "micro-configurations" corresponding to each network element (such as an interface, a routing protocol, link weight, etc.) that is configured by the programmer's code.

2. **Step 2.** The micro-configurations for the same router are collected to form a single configuration file which is then uploaded to the corresponding router through TFTP for update.

Figure 6.8 shows the expansion of the API plane. Since the API plane will always be controlled by the vendor, implementation details will seldom be known. The model API is developed in Java and is based on wrappers around the Net-SNMP library (developed in C). The Net-SNMP libraries provide all the SNMP [67] communication. Figure 6.8 shows the internals of the API. The accessible libraries are a collection of methods that allow access to the various entities on a router (interfaces, protocols,

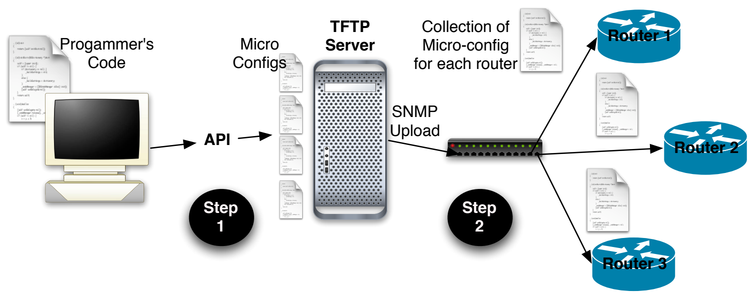


Figure 6.7: Translation of code to running configuration on routers.

etc.). API Access to equipment configurations are provided through corresponding entity MIBs.

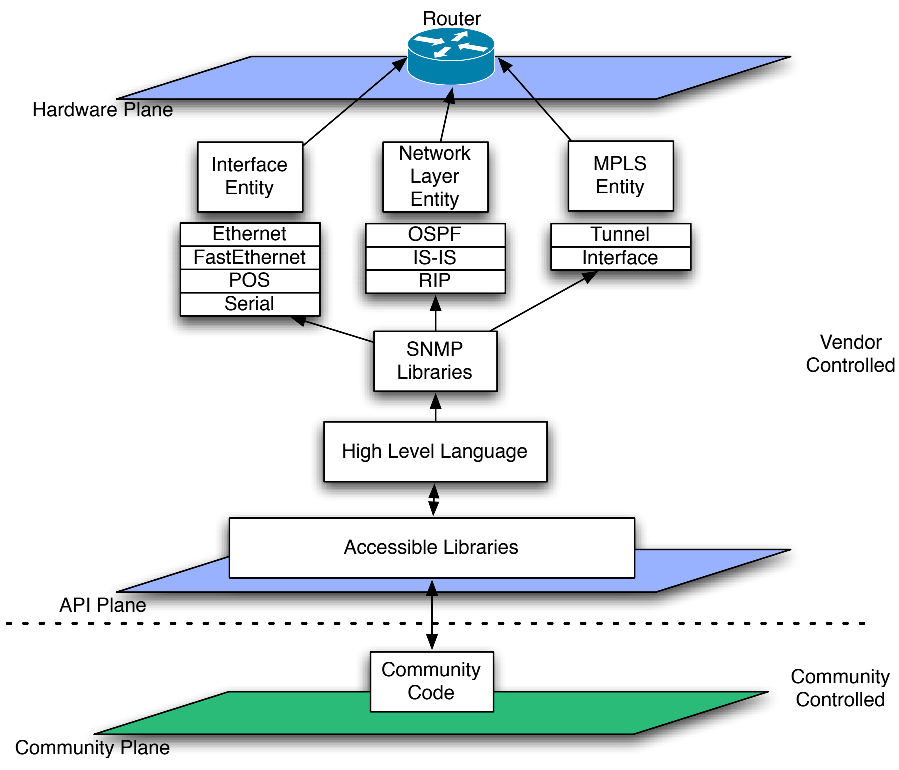


Figure 6.8: Structure of the model API.

6.3.1 Member Classes

Currently the API is structured logically into four classes: Router, Interface, IGP and MPLS.

1. **Router Class.** The Router class initializes the SNMP channel to a router through which all configuration initializations and updates take place. This class also contains methods for generating 'micro-configuration' files corresponding to the the configuration of hardware (interfaces) and protocols (OSPF, ISIS, RSVP, etc.) on the router.

2. **Interface Class.** This class allows the access and configuration of physical interfaces on a router and is divided into subclasses, each corresponding to the type of physical interface. Specific API methods are developed and associated to each type of interface as they have different configuration procedures. Similar methods also exist to get interface specific information once they are configured. This class uses the IF-MIB [76].

3. **IGP Class.** The IGP class allows the configuration of the underlying IGP. Once the interfaces are configured using the methods in the Interface Class, a network layer protocol such as OSPF or ISIS (corresponding subclasses of the IGP class) can be configured so that routing can take place in the network. Again, information about connectivity is gathered from the OSPF-MIB [77] and the ISIS-MIB.

4. **MPLS Class.** All MPLS functions in a router are enabled, configured, controlled and accessed through elements of this class. This class has two subclasses, the LSR and the Tunnel subclass. The LSR subclass handles the configuration required for setting up MPLS on a router whereas the Tunnel subclass handles all elements for creation, configuration and accessing of MPLS Tunnel

Table 6.1: Some typical API calls

Class	Function	API Call
Router	Initialize Router Connectivity	`routerObject.initRouter(snmpString,loopbackIP)`
Interface	Initialize Interface	`interfaceObject.initInterface(interFaceName,ipAddress,netMask)`
Interface	Configure Interface	`routerObject.configure(interfaceObject)`
IGP	Configure OSPF	`igpObject.confOSPF(procID,netIP,invMask,areaID)`
IGP	Add Network	`igpObject.addNetworkOSPF(netIP,invNetMask,areaID)`
MPLS	Enable on a router	`mplsObject.enableMPLS(igpObject)`
MPLS	Enable on an interface	`mplsObject.enableMPLSInterface(interfaceName)`
Tunnel	Initialize Tunnel	`tunnelObject.initTunnel(sourceIP,destIP,resvBandwidth,pathOption)`
Tunnel	Setup Tunnel	`routerObject.setupTunnel(tunnelObject)`

parameters. This class uses the MPLS-TE MIB [78] and the MPLS LSR MIB [79].

Though extensions are possibly limitless and are in progress, the four entities are sufficient to demonstrate the easy enabling of a new traffic engineering mechanism using the availability of an open API. Table 6.1 shows some basic API calls and the functions they perform.

6.4 Deployment Experiences

This section presents a performance evaluation of a novel traffic engineering mechanism that has been implemented using the model API.

6.4.1 Experimental Setup

1. **Testbed Configuration.** Testing the new bandwidth reservation mechanism involves two steps. The first one is to setup the network and the second one is to actually initiate the new mechanism. Instead of configuring the routers manually, the API was used in the first step to configure the network, the underlying routing protocol (OSPF) and setup MPLS based traffic engineering tunnels. Table 6.2 shows how API based code can setup the network. Due to

Table 6.2: API based implementation to setup network and initiate the new mechanism

Step	API Call	Function
1	`router1.initRouter("snmpString", "192.168.0.1");`	Initialize SNMP channel to router
2	`router2.initRouter("snmpString", "192.168.0.2");`	
3	`FE00.initInterface("FastEthernet0/0", "14.1.0.1", "255.255.255.0");`	Initialize Interface
4	`FE01.initInterface("FastEthernet0/1", "14.1.0.2", "255.255.255.0");`	
5	`router1.configure(FE00); router2.configure(FE01);`	Configure interface on router
6	`igp1.confOSPF(router1, 100, 14.1.0.0, 0.0.255.255, 0);`	Configure OSPF
7	`igp2.confOSPF(router2, 200, 14.1.0.0, 0.0.255.255, 0);`	
8	`igp1.addNetworkOSPF("12.1.0.0", "0.0.255.255", 1);`	Add a network to OSPF
9	`igp2.addNetworkOSPF("16.1.0.0", "0.0.255.255", 2);`	
10	`router1.configure(igp1); router2.configure(igp2);`	Configure OSPF on the routers
11	`mpls1.enableMPLS(igp1); mpls2.enableMPLS(igp2)`	Enable MPLS on a router
12	`mpls1.enableMPLSInterface(FE00)`	Enable MPLS on an interface
13	`mpls2.enableMPLSInterface(FE01)`	
14	`router1.configure(mpls1); router2.configure(mpls2);`	
15	`tunnel1.initTunnel(192.168.0.1, 192.168.0.2, 0, 1)`	Initialize 0 bandwidth Tunnel
16	`router1.configure(tunnel1)`	Setup Tunnel
17	`newMechanismObject.initiateMechanism(tunnel1)`	Initiate the new mechanism

the object oriented nature, it can be seen that configuring interfaces, routing, MPLS and MPLS Tunnels is very intuitive. From Table 6.2.

- **Step 1 and 2.** Initializes the SNMP communication channel and provides the Loopback IP addresses through which configuration changes will take place.
- **Step 3, 4 and 5.** Initialize and configure the interfaces on the routers.
- **Steps 6-10.** Configure OSPF processes and areas on the routers and add networks to the OSPF process. This ensures the configuring of the network in Figure 6.9.
- **Steps 11-14.** Enable MPLS for the routers and their interfaces.
- **Steps 15 and 16.** Configure an MPLS Tunnel and set it up. This causes traffic to flow through the MPLS tunnel.
- **Step 17.** Initiates the new reservation mechanism.

2. **Traffic Generation.**

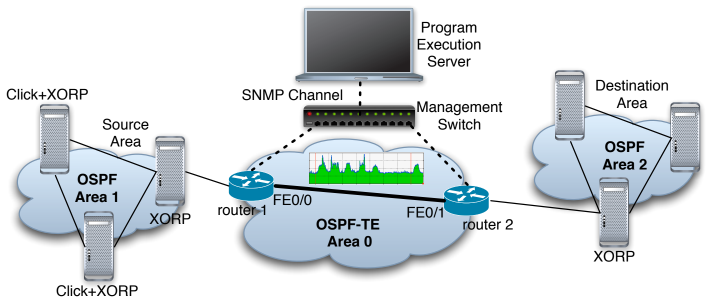


Figure 6.9: Scenario configured for the experiment.

6.4.2 API Based Implementation

To show an example how a simple yet functional API can help to implement a Traffic Engineering mechanism, we implement the Trend Based Bandwidth Reservation Mechanism, presented in great detail in Section X. In this mechanism, traffic traversing a tunnel is periodically sampled and, based on the trend of the flowing traffic, the reservation of the corresponding tunnel is resized. Moreover, the newly resized tunnel is routed along the shortest possible path corresponding to the reservation. Increasing and decreasing reservations are controlled by two parameters, underbooking (x_{ub}) and overbooking (x_{ob}) fractions. Underbooking occurs when the reservation is less than the traffic passing through the tunnel and conversely, overbooking occurs when the traffic traversing the tunnel is more than the reserved bandwidth. A memory also keeps track of the traffic trend at regular time instances (called seed points) over the period of a day. Two circular queues called Sample Queue (SQ) and Weighted Average Queue (WAQ) hold the samples and the weighted average of the samples. Rules are then applied to the information obtained from these queues, memory and current trend to take a resizing decision.

Table 6.3 shows the Java methods that make up the new reservation mechanism. For every Tunnel upon which the new mechanism acts, a new thread is started. This

Table 6.3: Methods in the new reservation mechanism

Step	**Java Method**	**Function**
1	`targetTunnel.getSample()`	API method to sample traversing traffic
2	`insertSampleIntoQueue()`	Method to insert sample into queue
3	`smoothSamples()`	Method to smooth information form samples
4	`returnSlopeBasedEstimate()`	Method that returns a slope based estimate of future traffic
5	`returnMemoryBasedEstimate()`	Method that returns a memory based estimate of future traffic
6	`decideResizing()`	Method that decides whether resizing is required
7	`resizeReservation()`	Method that configures the router to resize the tunnel

allows parallel and independent functioning of this mechanism on several tunnels at once.

- **Step 1.** Involves sampling the traffic flowing through the tunnel. Since an MPLS Tunnel is an interface (type 150), a simple SNMP execution can get the desired information.
- **Step 2-6.** Contains methods (emulating community code) that process the samples based on which decisions to resize the Tunnel reservation can be taken.
- **Step 7.** Writes a micro-configuration file that updates the concerned Tunnel head-end router and changes the reservation.

6.4.3 Performance Analysis

This section discusses the performance analysis of the bandwidth reservation mechanism and the overhead of using this API. The mechanism is executed on an Intel Pentium III 400Mhz with 256MB of RAM. Two tunnels, Tunnel 1 and Tunnel 2 respectively are setup carrying traffic from Area 1 (12.0.0.0) to Area 2 (16.0.0.0). Tunnel 1 has a sampling period of 5 minutes whereas Tunnel 2 has a sampling period of 5 seconds. Tunnel 1 uses 24 seed points to maintain memory (1 every hour) and Tunnel 2 uses 48 seed points to maintain memory (one every half hour). The mechanism is run for two days and the behavior observed. Maximum under-booking allowed is 5% ($x_{ub} = 0.05$) of the instantaneous reserved bandwidth and maximum over-booking allowed is 25% ($x_{ob} = 0.25$) of the same.

Figure 6.10 shows the behavior of reserved bandwidth corresponding to the traffic. Since the initial reserved bandwidth is 0, the mechanism over conservatively reserves a high bandwidth and later reduces it over time. After 24 hours, the reserved bandwidth changes are also influenced by the seed points taken over the previous day.

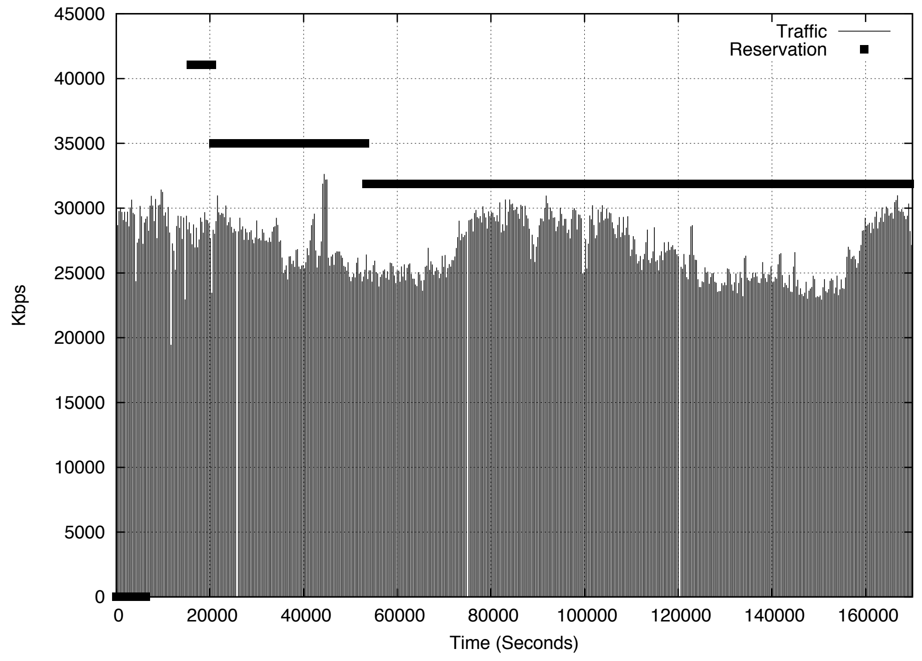


Figure 6.10: Traversing traffic and corresponding reservation for Tunnel 1.

Figure 6.11 shows the seed points taken at 1 hour intervals over the duration of the day.

Figure 6.12 shows how the reservation changes for Tunnel 2. Since the mechanism is sampling traffic every 5 seconds, the mechanism is over reactive. At the beginning

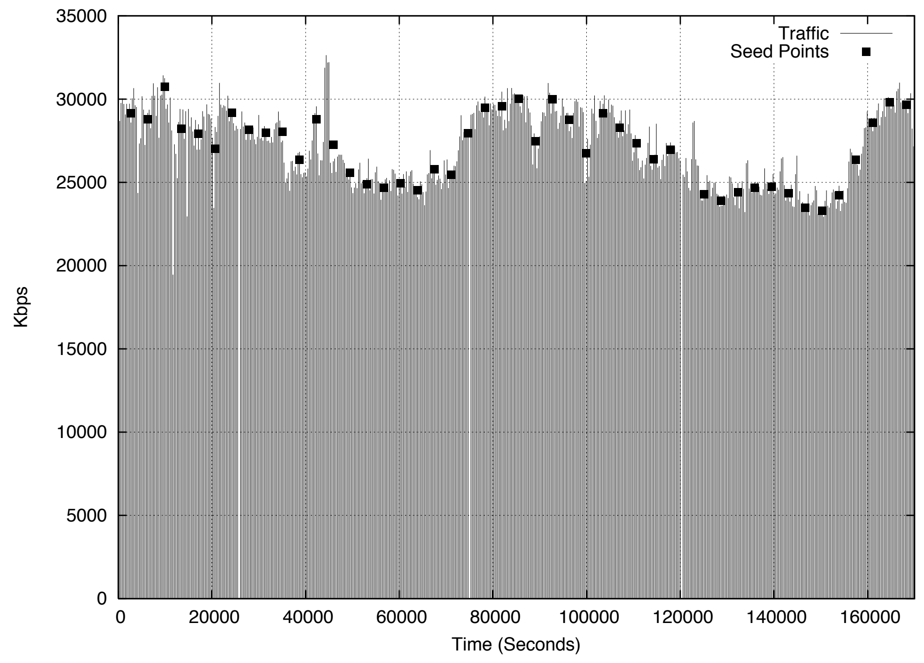


Figure 6.11: Traversing traffic and seed points for Tunnel 1.

it can be seen that the mechanism causes several changes before slowly stabilizing.

Figure 6.13 shows an MRTG [68] view of the traffic generated due to the SNMP queries for sampling traffic and the configuration files being uploaded to the routers during a configuration update of the Tunnels. It can be seen that traffic leaving the router (Blue) is lesser than that entering it (Green). This is because input traffic corresponds to uploading configuration files whereas output traffic corresponds only to sampling information.

The computational overhead of running this mechanism is dependent on the coding techniques and algorithms used. On the test machine described above, CPU usage was less than 3% and memory consumption was less than 1% on an average. The Java code corresponding to this mechanism, including managing of timers and threads was approximately 400 lines.

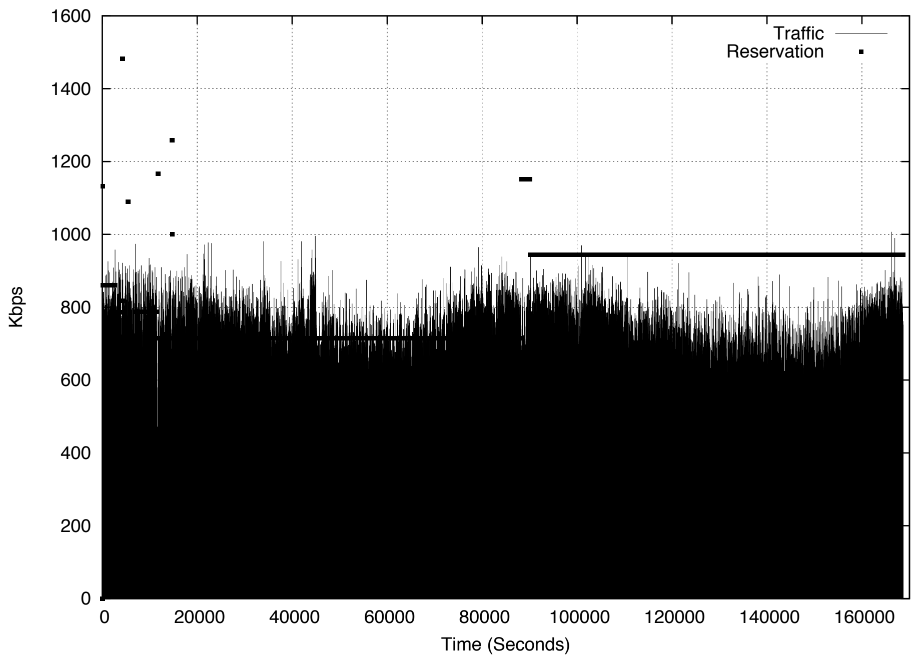


Figure 6.12: Traversing traffic and corresponding reservation for Tunnel 2.

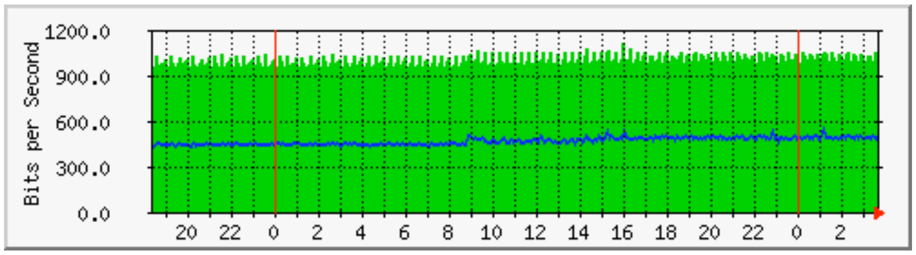


Figure 6.13: SNMP Traffic generated during the execution of the new mechanism. Green=input traffic, Blue=output traffic.

6.5 Summary and Future Directions

In this chapter we discussed and presented a testbed comprising of Cisco routers and PC/Sun workstation based routers. The development of this testbed has helped to build a platform for carrying out testing and analysis of new mechanisms and understanding the issues that plague deployment on live networks. We also presented the idea of open API based access to commercially available vendor based hardware. This API will be provided by the vendor itself and will serve several purposes. It will allow the vendor to insulate proprietary intellectual content and at the same time provide a means for the community to test and deploy new network mechanisms. To support this idea, a model API is developed and tested using a new bandwidth reservation mechanism based on it. The overhead associated with SNMP based communication and update of configuration is minimal. With access to APIs, compiled code could perhaps be directly flashed onto router memory for configuration updates and SNMP based communication could be prevented altogether.

The scope for furthering this work is immense. The primary goal is to enrich the API further by developing new methods that give more control to a developer. Currently, all SNMP communication is undertaken with the aim of either sampling or updating the running configuration on the routers. Since currently the testbed is manually manageable, methods that obtain information about interfaces before configuring them are not a part of the API. For larger scenarios where physical access might not be possible, a new family of methods are being developed to get current interface state information before configuring them. Discovery of interfaces and routers in the network before proceeding with configuration and testing is also a future direction. Strong error checking is also being developed to prevent mis-configurations or over-writing of existing configuration as much as possible. In terms of testing, newer and more varied mechanisms are currently being implemented to test the efficacy of the API in realms other than traffic engineering. As the API moves towards stabilization, an important future goal is to release it to the community as an open source project for adoption, testing and further development.

7. Conclusion and Future Directions

Efficient resource management in communication networks has become a very important issue in network management. Network level resources constitute of available bandwidth, paths amongst nodes, signaling noise in the network, etc. and it is always favorable to have even network utilization, paths with low latency and no unnecessary background signaling in the network. At the equipment level, resources are constituted by available processing capability and available memory for state maintenance. The optimization of these resources is a challenge as they are all related to each other and the relation is not always obvious. As time passes, networks are carrying larger volumes of diverse traffic and allowing more users to get online. With varied QoS requirements, optimizing and managing resources require dedicated mechanisms that take into account the behavioral relationship amongst them.

7.1 Overview and Contributions

This thesis has the following contributions to the existing literature on resource management in Traffic Engineered networks.

1. **Identification of key network entities and characteristics.** We first identify the important network entities that will be involved in development and deployment TE mechanisms. This is followed by the identification of measures that are used to quantify the performance of these entities. A set of these measures are selected to then quantify the performance of new TE mechanisms.

2. **Comparison and quantification of MPLS and IP TE.** The performance of IP and MPLS Traffic Engineering techniques has seldom been compared.

We identify performance measures that fairly quantify and compare this performance under normal and link failure scenarios.

3. **Dynamic bandwidth reservation.** MPLS-TE involves the static reservation of bandwidth to deliver QoS guarantees. We show that with the introduction of dynamic bandwidth reservation, the same network has enough available bandwidth to accommodate traffic growth and traffic shifts arising from link and node failures. The dependence of performance on traffic profiles and timezones is highlighted also. Finally, we show that information from traffic profiles can be used for better forecasting of future traffic demands and in the process design and develop a novel Trend Based Bandwidth Provisioning Algorithm.

4. **Performance of inter-domain path computation methodologies.** With traffic traversing multiple areas and provider domains, performance of TE mechanisms depend on the quality of the path computed. We identify key performance measures that quantify the performance of inter-domain path computation. We compare two technologies, the "Per-Domain" approach and the PCE based approach. We motivate the use of PCE based inter-domain path computation for TE mechanisms.

5. **MPLS testbed.** To test the feasibility and operational viability of the TE mechanisms developed and discussed in this thesis, we have built an MPLS TE testbed. It consists of 10 Cisco MPLS capable routers and 20 desktop computers consisting of PCs and Sun Workstations. An open API in Java has also been developed to provide a platform for the implementation of TE mechanisms.

7.2 Future Work

The scope for future work in this area is immense. As new technologies emerge, so will new demands and requirements for stringent QoS. Newer mechanisms that provide robust and finer TE will be the new order of the day.

Insight from research in dynamic bandwidth reservation has yielded several interesting avenues for progress. The Trend Based Bandwidth Provisioning Algorithm has only introduced the idea of using periodicity of profile characteristics for future bandwidth forecasting. Much work remains in making this idea go further. The usage of tunable knobs to customize the algorithm is one such possibility. The algorithm can also be coupled with priority based functioning such that different classes of traffic are treated differently. This will allow high priority traffic to follow better paths. Representation of the “trend” of the profile itself can be made much more efficient in terms of processing and storage overhead.

Coupling the dynamic bandwidth reservation mechanisms with PCE based path computation is a natural extension to the inter-domain scenario. Many other factors need to be kept in mind when extending functionality to this scenario. Resizing and rerouting constraints arise since paths could traverse multiple areas about which limited information could be available.

There also exists a lot of scope for the development of the MPLS testbed. With new TE mechanisms being developed in the research community, the need is felt for a platform that will allow their testing on a realistic environment. Successful deployment of TE mechanisms developed by the research community is only possible when thoroughly tested on commercial hardware. The testbed API has ample room for development is to be used in conjunction with the testbed hardware to provide such a platform. A very exciting phase then lies ahead when testing of TE mechanisms proven theoretically and through simulations is undertaken on this testbed.

Bibliography

[1] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, *"Overview and Principles of Internet Traffic Engineering"*, IETF RFC 3272, May 2002.

[2] J. Moy, *"OSPF Version 2"*, IETF RFC 2328, April 1998.

[3] R. Callon, *"Use of OSI IS-IS for Routing in TCP/IP and Dual Environments"*, IETF RFC 1195, December 1990.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

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

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

[7] D. Awduche, L.. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, *"RSVP-TE: Extensions to RSVP for LSP Tunnels"*, IETF RFC 3209, December 2001.

[8] D. Awduche, A. Hannan, and X. Xiao, *"Applicability Statement for Extensions to RSVP for LSP-Tunnels"*, IETF RFC 3210, September 1999.

[9] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, *"Resource ReSerVation Protocol"*, IETF RFC 2205, September 1999.

[10] J. Boyle, V. Gill, A. Hannan, D. Cooper, D. Awduche, B. Christian, and W. S. Lai, *"Applicability Statement for Traffic Engineering with MPLS"*, IETF RFC 3346, August 2002.

[11] L. Andersson, P. Doolan, N. Feldman, A. Fredette, and B. Thomas, *"LDP Specification"*, IETF RFC 3036, January 2001.

[12] Katz et al., *"Traffic Engineering (TE) Extensions to OSPF Version 2"*, IETF RFC 3630, September 2003.

[13] E. Osborne and A. Simha, *Traffic Engineeering with MPLS*, Cisco Press, 2002.

[14] B. Fortz, J. Rexford, and M. Thorup, "Traffic Engineering with Traditional IP Routing Protocols," *IEEE Communications Magazine*, 2002.

[15] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," in *Proceedings of IEEE INFOCOM*, 2002.

[16] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," *IEEE JSAC*, vol. 20, May 2002.

[17] B. Fortz and M. Thorup, "Robust optimization of OSPF/IS-IS Weights," in *Proceedings of INOC*, 2003.

[18] P. Francois and O. Bonaventure, ""Avoiding transient loops during IGP Convergence in IP Networks," in *Proceedings of IEEE INFOCOM*, 2005.

[19] S. Dasgupta, J. C. de Oliveira, and J.-P. Vasseur, "A Performance Study of IP and MPLS Traffic Engineering Techniques under Traffic Variations," in *Proceedings of IEEE GLOBECOM*, Washington D.C., USA, November 2007.

[20] H. Smit and T. Li, *IS-IS extensions for Traffic Engineering*, IETF Draft, Work in Progress, August 2005.

[21] Q. Zhang, W. Zhu, and Y.-Q. Zhang, "Resource Allocation for Multimedia Streaming Over the Internet," *IEEE Transactions on Multimedia*, vol. 3, no. 3, September 2001.

[22] F. Yu, Q. Zhang, W. Zhu, and Y.-Q. Zhang, "QoS-Adaptive Proxy Caching for Multimedia Streaming Over the Internet," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 3, March 2003.

[23] S. Weber and G. de Veciana, "Network design for rate adaptive media streams," in *Proceedings of the IEEE INFOCOM 2003*, San Francisco, CA, USA, 2003.

[24] A. Byde, M. Sall, and C. Bartolini, *Market-Based Resource Allocation for Utility Data Centers, Technical Report, HPL-2003-188*, HP Laboratories, 2003.

[25] A. AuYoung, B. N. Chun, A. C. Snoeren, and A. Vahdat, "Resource Allocation in Federated Distributed Computing Infrastructures," in *Proceedings of the First Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, 2004.

[26] D. P. Pazel, T. Eilam, L. L. Fong, M. Kalantar, K. Appleby, and G. Goldszmidt, "Neptune: a dynamic resource allocation and planning system for a cluster computing utility," in *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2002.

[27] X. Wang, R. Ramjee, and H. Viswanathan, "Adaptive and predictive downlink resource management in next generation CDMA networks," in *Proceedings of the IEEE INFOCOM 2004*, Hong Kong, China, 2004.

[28] C.-T. Chou and K. G. Shin, "Analysis of Combined Adaptive Bandwidth Allocation and Admission Control in Wireless Networks," in *Proceedings of the IEEE INFOCOM 2002*, New York, NY, USA, June 2002.

[29] I. Koutsopoulos and L. Tassiulas, "Adaptive Resource Allocation in SDMA-based Wireless Broadband networks with OFDM Signaling," in *Proceedings of the IEEE INFOCOM 2002*, New York, NY, USA, June 2002.

[30] G. B. Figueiredo, J. A. S. Monteiro, N. L. S. Fonseca, and A. A. A. Rocha, "Dynamic Sizing of Label Switching Paths In MPLS Networks," in *Proceedings of the IEEE International Telecommunications Symposium (ITS-2002)*, Natal, RN, Brazil, September 2002, pp. 593–598.

[31] W. Shen and M. Devetsikiotis, "A Self-Sizing Framework for Adaptive Resource Allocation in Label-Switched networks," in *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, Fort Worth, Texas, USA, October 2002.

[32] H. Levy, T. Mendelson, and G. Goren, "Dynamic Allocation of Resources to Virtual Path Agents," *IEEE/ACM Transactions on Networking*, vol. 12, no. 4, August 2004.

[33] Y. Watanabe and T. Oda, "Dynamic Routing Schemes for International Networks," *IEEE Communications Magazine*, vol. 28, no. 10, October 1990.

[34] A. N. Kashper and Y. Watanabe, "Dynamic Routing in the Multiple Carrier International Network," *IEEE Communications Magazine*, vol. 33, no. 7, July 1995.

[35] B. Krithikaivasan, Y. Zeng, K. Deka, and D. Medhi, "ARCH-based Traffic Forecasting and Dynamic Bandwidth Provisioning for Periodically Measured Nonstationary Traffic," *IEEE/ACM Transactions on Networking*, August 2006.

[36] T. Anjali, C. Bruni, D. Iacoviello, and C. Scoglio, "Dynamic bandwidth reservation for label switched paths: An on-line predictive approach," *Computer Communications*, vol. 29, 2006.

[37] S. Dasgupta, J. C. de Oliveira, and J.-P. Vasseur, "A New Distributed Dynamic Bandwidth Reservation Mechanism to Improve Resource Utilization," in *Proceedings of IEEE INFOCOM*, Barcelona, Spain, April 2006.

[38] S. Dasgupta, J. C. de Oliveira, and J.-P. Vasseur, "Dynamic Traffic Engineering for Mixed Traffic on International networks: Simulation and analysis on real network and traffic scenarios," *Computer Networks*, vol. 52, no. 11, August 2008.

[39] S. Dasgupta, J. C. de Oliveira, and J.-P. Vasseur, "Trend Based Bandwidth Provisioning: An Online Approach for Traffic Engineered Tunnels," in *Proceedings of IEEE Next Generation Internet*, Krakow, Poland, April 2008.

[40] N. Spring, R. Mahajan, and D. Wehterall, "Measuring ISP Topologies with Rocketfuel," in *Proceedings of ACM SIGCOMM*, 2002.

[41] J. Guichard, F. Le Faucheur, and J.-P. Vasseur, *Definitve MPLS Network Designs*, Cisco Press, 2005.

[42] "The Abilene Backbone Network," http://abilene.internet2.edu/.

[43] S. Yasukawa, A. Farrel, and O. Komolafe, *"An Analysis of Scaling Issues in MPLS-TE Backbone Networks"*, Work in progress, July 2007.

[44] C. Fraleigh et al., "Packet-Level Traffic Measurements from the Sprint IP Backbone," *IEEE Network*, 2003.

[45] J. W. Roberts., "Traffic Theory and the Internet," *IEEE Communications Magazine*, 2001.

[46] Q. Hao, S. Tartarelli, and M. Devetsikiotis, "Self-Sizing and Optimization of High-Speed Multiservice Networks," in *Proceedings of IEEE GLOBECOMM*, 2000.

[47] R. Guerin, H. Ahmadi, and M. Naghshineh, "Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Network," *IEEE JSAC*, vol. 9, no. 7, 1991.

[48] A. Farrel, J.-P. Vasseur, and J. Ash, *"A Path Computation Element (PCE) Based Architecture*, IETF RFC 4655, August 2006.

[49] J. Ash and J. L. Le Roux, *"A Path Computation Element (PCE) Communication Protocol Generic Requirements"*, IETF RFC 4657, September 2006.

[50] J. L. Le Roux, *"Requirements for Path Computation Element (PCE) Discovery"*, IETF RFC 4674, October 2006.

[51] J. L. Le Roux, J.-P. Vasseur, Y. Ikejiri, and R. Zhang, *"OSPF Protocol Extensions for Path Computation Element (PCE) Discovery"*, IETF RFC 5088, January 2008.

[52] J. L. Le Roux, J.-P. Vasseur, Y. Ikejiri, and R. Zhang, *"IS-IS Protocol Extensions for Path Computation Element (PCE) Discovery"*, IETF RFC 5089, January 2008.

[53] E. Oki, *"PCC-PCE Communication and PCE Discovery Requirements for Inter-Layer Traffic Engineering"*, Internet-Draft, Work in progress, April 2008.

[54] J. L. Le Roux and J.-P. Vasseur, *"Encoding of Objective Functions in Path Computation Element communication Protocol (PCEP)"*, Internet-Draft, Work in progress, March 2008.

[55] S. Dasgupta, J. C. de Oliveira, and J.-P. Vasseur, "Path Computation Element Based Architecture for Inter-Domain MPLS/GMPLS Traffic Engineering: Overview and Performance," *IEEE Network*, vol. 21, no. 4, July/August 2007.

[56] S. Dasgupta, J. C. de Oliveira, and J.-P. Vasseur, *"Performance Analysis of Inter-Domain Path Computation Methodologies"*, Internet-Draft, Work in progress, November 2007.

[57] J.-P. Vasseur, R. Zhang, N. Bitar, and J. L. Le Roux, *"A Backward Recursive PCE-based Computation (BRPC) procedure to compute shortest inter-domain Traffic Engineering Label Switched Paths"*, Internet-Draft, Work in progress, December 2006.

[58] A. Farrel et al., *"Crankback Signaling Extensions for MPLS and GMPLS RSVP-TE"*, Internet-Draft, Work in progress,, November 2005.

[59] J. V. der Merwe and C. Kalmanek, "Network Programmability is the answer! What was the question again?," in *PRESTO 2007, Princeton*, 2007.

[60] J. van der Merwe, S. Rooney, I. Leslie, and S. Crosby, "The Tempest, a Framework for Network Programmability ," *IEEE Network Magazine*, vol. 12, no. 3, 1998.

[61] S. Rooney, J. van der Merwe, S. Crosby, and I. Leslie, "The Tempest, a Framework for Safe, Resource Assured, Programmable Networks ," *IEEE Communications Magazine*, vol. 36, no. 10, 1998.

[62] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, S. Rao, and W. Aiello, "PRESTO: A tool for Configuration Management at Massive Scale," in *Proceedings of USENIX Annual*, 2007.

[63] "Ubuntu Linux," http://www.ubuntu.com/.

[64] M. Handley, E. Kohler, A. Ghosh, O. Hodson, and P. Radoslavov, "Designing Extensible IP Router Software," in *Proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[65] "XORP: Open Source IP Router," http://www.xorp.org.

[66] "The Net-SNMP Library," http://net-snmp.sourceforge.net/.

[67] J. Case, M. Fedor, M. Schoffstall, and J. Davin, *"Simple Network Management Protocol"*, IETF RFC 1157, May 1990.

[68] Tobi Oetiker, "Multi Router Traffic Grapher," http://oss.oetiker.ch/mrtg/.

[69] Tobi Oetiker, "RRDtool," http://oss.oetiker.ch/rrdtool/.

[70] "Xerces Java Parser," http://xerces.apache.org/xerces-j/.

[71] "The Apache Commons Project," http://commons.apache.org/net/.

[72] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Computer Communications Review*, 2006.

[73] "The Click Modular Router," http://www.read.cs.ucla.edu/click/.

[74] E. Kohler, R. Morris, B. Chen, and M. Poletto, "Modular components for network address translation," in *Proceedings of OPENARCH*, 2002.

[75] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI Veritas: Realistic and Controlled Network Experimentation," in *Proceedings of ACM SIGCOMM*, 2006.

[76] K. McCloghrie and F. Kastenholz, *"The Interface Group MIB using SMIv2"*, IETF RFC 2233, November 1997.

[77] F. Baker and R. Coltun, *"OSPF Version 2 Management Information Base"*, IETF RFC 1850, November 1995.

[78] C. Srinivasan, A. Viswanathan, and T. Nadeau, *"Multiprotocol Label Switching (MPLS) Traffic Engineering (TE) Management Information Base (MIB)"*, IETF RFC 3812, June 2004.

[79] C. Srinivasan, A. Viswanathan, and T. Nadeau, *"Multiprotocol Label Switching (MPLS) Label Switching Router (LSR) Management Information Base (MIB)"*, IETF RFC 3813, November 2004.

Appendix A. Publications

- Sukrit Dasgupta, Jaudelice C. de Oliveira and Jean-Philippe Vasseur. *"Dynamic Traffic Engineering for Mixed Traffic on International Networks". Computer Networks*, vol. 52(11), pp. 2237–2258, August 2008.

- Sukrit Dasgupta, Jaudelice C. de Oliveira and Jean-Philippe Vasseur. *"Trend Based Bandwidth Provisioning: An Online Approach for Traffic Engineered Tunnels".* In Proceedings of the Next Generation Internet Networks (Euro-NGI) 2008, Krakow, Poland, April 2008.

- Sukrit Dasgupta, Jaudelice C. de Oliveira and Jean-Philippe Vasseur. *"Performance Analysis of Inter-Domain Path Computation Methodologies, Version 01".* IETF Draft submitted to CCAMP Working Group, November 2007.

- Sukrit Dasgupta, Jaudelice C. de Oliveira and Jean-Philippe Vasseur. *"A Performance Study of IP and MPLS Traffic Engineering Techniques under Traffic Variations".* In Proceedings of the 50th Annual IEEE GLOBECOM Technical Conference, Washington, DC, USA, November 2007.

- Sukrit Dasgupta, Jaudelice C. de Oliveira and Jean-Philippe Vasseur. *"Path Computation Element Based Architecture for Inter-Domain MPLS/GMPLS Traffic Engineering: Overview and Performance." IEEE Network*, vol. 21(4), pp. 38–45 July/August 2007.

- Joshua Goldberg, Sukrit Dasgupta and Jaudelice C. de Oliveira. *"Bandwidth Constraint Models: A Performance Study with Preemption on Link Failures".* In Proceedings of the 49th Annual IEEE GLOBECOM Technical Conference, San Francisco, CA, USA, November 2006.

- Steven Weber, Jaudelice C. de Oliveira, Sukrit Dasgupta, Bryan Willman and Zhen Zhao. *"Combined Preemption and Adaptation in Next Generation Multi-Service Networks".* In Proceedings of the 41st IEEE International Conference on Communications (IEEE-ICC), Istanbul, Turkey, June 2006.

- Sukrit Dasgupta, Jaudelice C. de Oliveira and Jean-Philippe Vasseur. *"A New Distributed Dynamic Bandwidth Reservation Mechanism to Improve Resource Utilization".* In Proceedings of the 25th Conference on Computer Communications (IEEE-INFOCOM), Barcelona, Spain, April 2006.

Vita

Sukrit Dasgupta was born in Mumbai, Maharashtra, India. He received his B.Tech in Computer Engineering from Sikkim Manipal Institute of Technology, India, in July 2003. He joined the Department of Electrical and Computer Engineering at Drexel University in September 2003 and has been affiliated with the Applied Networking Research Laboratory ever since. His research has spanned several areas of Computer Networks including Routing, Traffic Engineering, Multi-Protocol Label Switching and the Path Computation Element. He has been a teaching assistant for the undergraduate Computer Networking sequence for a duration of four years. During this time, he also served as a research assistant in the Applied Networking Research Laboratory.

His research has led to the publication of 5 conference papers, 2 journal papers and an IETF Draft that is currently under review in the CCAMP working group. He has served as a reviewer for the Elsevier Computer Networking Journal. He is a student member of IEEE.