THESIS
2
2000

This is to certify that the

dissertation entitled

Providing QoS in the Internet

presented by

Xipeng Xiao

has been accepted towards fulfillment
of the requirements for

Doctoral degree in Computer Science & Engineering

Lionel M. Ni
Major professor

Date March 31, 2000

DATE DUE	DATE DUE	DATE DUE
OCT 1 7 2002		
JAN 1 5 2004 0 1 1 2 0 4		
AUG 2 5 2005		
0 2 0 7 0 6		

PROVIDING QUALITY OF SERVICE IN THE INTERNET

By

Xipeng Xiao

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science and Engineering

2000

ABSTRACT

PROVIDING QUALITY OF SERVICE IN THE INTERNET

By

Xipeng Xiao

This dissertation presents a framework for providing *quality of service* (QoS) in the Internet. This framework consists of *traffic directing* and *load balancing* at the application layer, *differentiated services* (Diffserv) at the transport layer, *traffic engineering* and *fast reroute* at the network layer.

Traffic directing is to utilize the high-performance part of a network as much as possible and avoid using the low performance part. Load balancing is to distribute client requests to multiple servers so that service availability and responsiveness is increased. Approaches for traffic directing and load balancing are briefly described.

Diffserv is to divide traffic into different classes and treat them differently, especially when there is a shortage of network resources. Mechanisms needed at the edge and at the core of the network are described. These mechanisms can be used to achieve the desired *per-hop behaviors* (PHBs). By concatenating all these PHBs together, a certain level of QoS can be provided end to end. A conflict between Diffserv and TCP is also described, and a solution for resolving this conflict is proposed. This solves a significant problem in the Internet.

Traffic engineering is an iterative process of *network planning* and *network optimization*. Network planning is to improve the architecture (topology and link capacity) of a network in a systematic way so that the network is easy to operate, robust, and adaptive. Network optimization is to control the mapping and distribution of traffic over the existing network infrastructure to avoid and/or relieve congestion, and thus to optimize resource efficiency. The issues of designing a traffic engineering systems are discussed. A national traffic engineering system with *multi-protocol label switching* (MPLS) is then presented, and its performance evaluated. Based on the experience of implementing this system, a generic procedure is proposed for deploying large-scale traffic engineering systems. An approach for performing inter-domain traffic engineering in a quantitative way is then described. We also propose an offline constraint-based routing algorithm for computing the paths for MPLS *label switched paths* (LSPs), and describe how to provide Diffserv in an MPLS environment. Two emerging technologies that are closely related to traffic engineering, *multi-protocol lambda switching* (MPLmS) and *fast reroute*, are also discussed.

Constraint-based routing is one of the most important tools for traffic engineering. The issues related to constraint-based routing are discussed in detail. Heuristics for computing paths with propagation delay constraint, and for QoS routing on the top of virtual networks constructed in the traffic engineering process, are proposed. In order to reduce the computation complexity of constraint-based routing, an algorithm is proposed for reducing the routing table computation cost for OSPF. The correctness of the algorithm is illustrated. Similar idea can also be applied to IS-IS.

In summary, this dissertation not only discusses on all major issues related to QoS, but also presents a systematic approach for providing QoS in the Internet.

To Mom, Dad, Xibin and Lihua

ACKNOWLEDGMENTS

I complete this dissertation with guidance, help and support from many people.

First, I would like to thank my dissertation advisor, Dr. Lionel M. Ni. Dr. Ni always gives me his attention, advice and guidance whenever I need them. I can talk to him whenever I need to, no matter it is early in the morning or late in the night, remotely or face to face, with or without appointments. Dr. Ni helped to revise my first paper in networking research – "A Comprehensive Comparison of IP Switching and Tag Switching" – 5 or 6 times. Without his guidance, I won't be able to publish that paper, and will not gain the enormous confidence that paper brought to me.

When I was relatively matured in networking research, Dr. Ni gave me the freedom to work in the networking industry while completing my dissertation research. In the industry, I implemented mechanisms for providing QoS, and built a large-scale MPLS system for Internet traffic engineering. Without this industry experience, my dissertation will not be as intensive.

I would like to thank my Ph.D. committee members, Dr. Abdol H. Esfahanian, Dr. Tien-Yien Li and Dr. Matt W. Mutka, and other faculty and staff in MSU. They teach me, advise me, and help me to develop my knowledge and skills. These are life-long gifts.

My wife, Lihua Chen, contributes significantly to the completion of this dissertation. She is emotionally supportive while pushing me to get things done. She does lots of work so that I can concentrate on my dissertation research. I want to thank her wholeheartedly for her support.

I would like to thank Alan Hannan of Global Crossing Ltd., Yakov Rekhter of Cisco Inc., and Dave Wang of WANDL Inc. Alan mentored me in Global Crossing, trusted me to build the industry's first large-scale MPLS system, and pushed me to complete my dissertation. Yakov provided lots of guidance during these years and helped me to revise a number of papers. Dave generously provided me the WANDL simulator. These are essential to the completion of my dissertation.

Last but not the least, I would like to thank all my friends, especially Vasu Vuppala and Manqing Huang, my colleagues in the Multi-Gigabit Routing Division of Ascend Communications Inc. and in Global Crossing Ltd. for their friendship, help and support.

This dissertation is dedicated to my Mom, Dad, brother and my wife!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Images in this dissertation are presented in color.

Chapter 1 INTRODUCTION: TRENDS AND ISSUES IN THE INTERNET

Nothing has changed the world as rapidly as the Internet [1]. Today, the Internet has become one of the most important carriers of information. People also start to use the Internet to receive education, shop for grocery and do banking and stock transactions.

As the Internet becomes more and more important, requirement for the Internet also becomes higher. Driven by the need for more bandwidth, *quality of service* (QoS) and security, the Internet has evolved rapidly [2][3]. In this chapter, we discuss trends in the Internet and the related issues.

1.1 A Big Picture of the Internet

The Internet consists of many *local area networks* (LANs) and *metropolitan area networks* (MANs) interconnected by a backbone. The LANs and WANs can be university networks, corporate networks, or regional service providers, etc. The backbone of the Internet consists of a number of national and global *Internet service providers* (ISPs) such as AT&T WorldNet, Global Crossing, Sprint and UUNET. Networks of large ISPs are usually geographically overlapped. ISP networks are interconnected at public exchange points called *network access points* (NAPs) such as Mae-East and PAIX. In order to avoid congestion at the NAPs, ISP networks are also connected by private peering links.

An ISP network consists of *points of presence* (POPs) and links interconnecting the POPs. A sample part of an ISP network is shown in Fig. 1-1. Simply speaking, a POP consists of a combination of one or more *access routers* (ARs) connected to remote access customers, *border routers* (BRs) connected to other ISPs, *hosting routers* (HRs) connected to Web servers of companies such as Yahoo, and *core routers* (CRs) connected to other POPs. Traffic from ARs, BRs and HRs must first be sent to the CR. The CR will relay the traffic to another CR, until the traffic reaches the destination POP. The architecture of POPs is usually symmetric, as showed in Fig. 1-1. Different POPs are normally connected in a ring fashion to increase reliability. A large ISP may have more than 50 POPs.

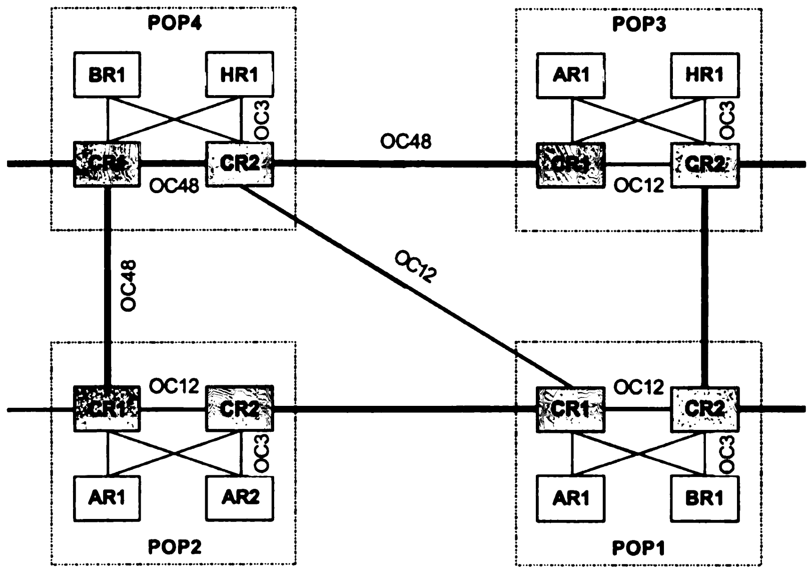


Figure 1-1. A sample part of an ISP network

1.2 Trends and Issues in the Internet Backbone

In this section, we discuss trends and issues in the Internet backbone.

1.2.1 Higher link speed

During the past four years, typical backbone links have evolved from DS-3 (45Mbps) to OC-12c (622 Mbps). This trend is even stronger now. ISPs such as Global Crossing and Qwest have deployed many OC-48c links (2.5Gbps) and even some OC-192c links (10Gbps).

There are two driving factors for higher link speed in the backbone. The first factor is the phenomenal growth of Internet traffic, which typically doubles every six months. With more DSLs and cables to homes, and more multimedia applications appearing on the Web, traffic may grow faster in the future. ISPs must provide sufficient bandwidth to meet the traffic demand. Another driving factor for higher link speed is cost and efficiency. For example, an OC-12c link (622 Mbps) does not cost four times as much as an OC-3c link (155 Mbps). In other words, the cost of per unit of bandwidth is lower for higher speed links. Besides, less effort is required to manage a single link OC-12c link than four OC-3c links.

Issues related to high-speed links are:

1. Propagation delay becomes a significant issue;

Propagation delay is the time it takes signal, usually in the form of light, to propagate across the geographic distance between traffic source and destination. Depending on transmission media, it takes light about 8ms to travel 1,000 miles. This means that the one-way propagation delay between the two coasts of the US (assumed to be 3,000 miles) is about 24 ms. With high-speed routers/switches and links, the queueing delay in each router/switch is usually well below 1ms in normal case. Assuming that traffic has to be forwarded by 10 routers/switches, then the propagation delay is approximately two-thirds of the total delay. This has a direct impact on how to provide QoS in the Internet. It means that complex mechanisms for reducing queueing delay are less important, unless the network is congested and the queueing delay becomes very long. Proper control of traffic mapping to avoid congestion can be more productive.

Since the propagation delay for a link is constant, higher link speed means more data on-the-fly. For example, the amount of data on the fly inside a cross-country OC-3c link is 155Mbps $\times$ 25 ms = 0.47 MB. With an OC-192c link, the amount of data on the fly becomes 29.76 MB. This is not a problem by itself, but if there is a link problem, more data need to be buffered or will be lost. This factor has to be considered in the design of the buffering system of high-speed routers and switches.

2. Network survivability becomes critical.

Network survivability is a network's capability to continue to provide a certain level of quality of service during link or router failure. With higher router/link speed, the importance of each router and link increases. A traditional approach for providing high network survivability is to have physical router and link redundancy. One example is SONET *automatic protection switching* (APS) [4]. If one router/link fails, traffic can be switched to another router/link within 50ms.

SONET APS is physical layer approach for providing high network survivability. Survivability can also be provided at the network layer. Because routers are connected in a ring fashion, there are multiple paths between each source-destination pair. When there is a router/link failure along the original path, traffic can be switched to the alternative path. The problem with this approach is that when there is a failure, it takes *interior gateway protocols* (IGPs) such as OSPF or IS-IS some time to propagate the failure information and for other routers to re-compute their routes. Depending on network size, it can take several seconds to several minutes for routing to converge. During this time interval, lots of traffic sent along an affected path will be lost because the intermediate routers cannot buffer that many data. In order to solve this problem, fast reroute is introduced. Fast reroute mechanism temporarily repairs the affected path so that it can continue to carry traffic, and notifies the affected sources to re-compute new paths for their traffic. Fast reroute will be further discussed in Chapter 4.

1.2.2 QoS

Traditionally, the Internet only provides best effort service. Traffic is processed as quickly as possible, but there is no guarantee as to timeliness or actual delivery.

With the flourishing of e-commerce, QoS has become much needed. There are two driving forces for QoS. First, companies that do business on the Web need QoS for better delivery of their content and/or services to attract more customers. Second, ISPs need the value-added services in their networks to increase revenue.

Integrated services (Intserv) and *differentiated services* (Diffserv) are the two models for providing QoS in the Internet. The essence of Intserv is to reserve resources (link bandwidth and buffer space) for each individual flow so that the service quality can be guaranteed if needed. The essence of Diffserv is to divide traffic into different classes and give them differentiated treatment [2].

In order to provide QoS in the Internet, some mechanisms need to be deployed. These mechanisms can be at the transport layer or at the network layer. How to provide QoS in the Internet is the theme of this dissertation. Recently, some mechanisms also appear in the application layer, they will also be briefly discussed in this dissertation.

Aside from the enabling mechanisms, monitoring, accounting and billing are the most important issues related to QoS. Both customers and providers need to know whether the contracted quality of service is delivered or not, and ISPs need to know how the users should be billed in either case. Future edge devices must have rich software features and traffic metering capability. Billing applications can then generate reports based on the statistics collected by the edge devices [23].

1.2.3 IP Networks with traffic engineering

The Internet backbone used to consist solely of IP routers. Several years ago, some ISPs decided to introduce ATM cores into their backbones. In such a network, IP routers were used at the edge, and were connected to a mesh of ATM switches. Through ATM *permanent virtual circuits* (PVCs), the routers were logically fully meshed [6][7].

The reasons why an ATM core was introduced are:

1. ATM switches were faster than IP routers at that time;
2. ATM networks can provide some QoS and traffic engineering capability.

However, ATM also introduces the following problems:

1. *Segmentation and reassembly* (SAR) is difficult to implement at high speed;

ATM is cell-based. IP packets have to be segmented into fixed-sized cells (53 bytes) at the ingress of an ATM network. At the egress, the cells must be reassembled into packets. Because of cell-interleave, SAR must perform queueing and scheduling for a large number of VCs. This makes it difficult to implement SAR at OC-48c or higher speed.

2. ATM header overhead is high.

Of the 53 bytes of an ATM cell, 5 byte is used for header. This makes header overhead of ATM relatively high.

Because of the above two problems, and because IP routers are now as fast as ATM swatches or even faster, ISPs that have ATM cores start to migrate their networks back to IP networks.

No matter in ATM networks or IP networks, traffic engineering is useful. In normal IP networks, shortest paths are always used to forward traffic. This may cause congestion on some links while leaving some other links under-utilized. Traffic engineering is an iterative process of network planning and network optimization. The purpose of traffic engineering is to optimize resource efficiency and network performance [8][9][10].

In order to do traffic engineering effectively in IP networks, network administrators must be able to control the paths of packets. This calls for some kind of connections in the connectionless IP networks. *Multi-protocol label switching* (MPLS) [11] can provide the connections with *label switched paths* (LSPs). This is one of the reasons why MPLS is useful for IP networks.

1.2.4 Security

With more and more companies relying on the Internet for their business, and more and more sensitive data being transmitted through the Internet, security has become critical.

Security is provided in the Internet by data encryption/decryption, packet filtering, user authentication, and resource usage authorization [12].

1.2.5 Optical switching

Currently, traffic is transmitted in optical form inside links. However, at every intermediate node from the source to the destination, light signal must first be converted into digital signal, and then routed or switched by the node. For transit traffic, the digital signal must be converted back to optical signal before being transmitted again.

There are two disadvantages associated with this. First, if the routing or switching function of a node fails, transit data will be affected. Second, the unit that does the optical-electrical-optical (O-E-O) conversion is very complex. Such unit will eventually become the bottleneck. Therefore, it is very desirable for transit lambda (a light channel in a fiber with DWDM) to be switched pure optically, bypassing the O-E-O conversion and routing/switching completely. This is called optical switching. Equipments performing optical switching are called optical cross-connects (OXCs). Only traffic that is destined to this router/switch is converted to digital signal and processed. In the next couple of years, OXCs will be tightly integrated with routers. They will make up the core of the Internet [13][14][15].

Optical switching will be further discussed in Chapter 4.

1.3 Organization of this Dissertation

This dissertation discusses how to provide QoS in the Internet. We focus on approaches and the corresponding mechanisms needed by these approaches in the backbone, i.e., ISP networks. These approaches are grouped according to the layers they belong to. We focus on approaches at the transport layer and at the network layer. For completeness, some approaches that are used at the application layer are also discussed.

In Chapter 2, we give a brief introduction to QoS, and propose a framework for providing QoS in the Internet. In Chapter 3, we discuss two common models for providing QoS, i.e., Intserv and Diffserv, and focus on the mechanisms needed for providing Diffserv. In Chapter 4, we discuss the most important network layer approach for providing QoS, i.e., traffic engineering, and present an approach for performing traffic engineering in the Internet with constraint-based routing, MPLS, enhanced IS-IS and RSVP as the path signaling protocol. The design and implementation of a national traffic engineering system is presented. Although we focus on the intra-domain aspect of traffic engineering, a systematic approach for doing inter-domain traffic engineering is also proposed. Two emerging technologies that are closely related to traffic engineering, multi-protocol lambda switching and fast reroute, are also discussed in this chapter. Constraint-based routing is discussed in Chapter 5. It is one of the most important tools for traffic engineering. Because constraint-based routing usually has high computation complexity, an approach for reducing the routing table computation cost for OSPF, one of the most important link state protocols, is described. Application layer approaches for providing QoS are briefly discussed in Chapter 6. Finally, the dissertation is concluded in Chapter 7.

1.4 Summary of Contributions

In this dissertation, a framework for providing QoS in the Internet is proposed. The major components of this framework are Diffserv, traffic engineering, traffic directing and load balancing.

For Diffserv, the mechanisms needed at the edge and at the core of the network are described. These mechanisms can be used to achieve the desired per-hop behaviors (PHBs). By concatenating all these PHBs together, a certain level of QoS can be provided end to end. A conflict between Diffserv and TCP is described, and a solution for resolving this conflict is proposed. This solution solves a significant problem in the Internet.

For traffic engineering, the design issues are first discussed. A national traffic engineering system with MPLS is then presented, and its performance is evaluated. Based on the experience of implementing this system, a generic procedure is proposed for deploying large-scale traffic engineering systems. An approach for performing inter-domain traffic engineering in a quantitative way is described. We also propose an offline constraint-based routing algorithm for computing the paths for LSPs, and describe how to provide Diffserv in an MPLS environment. The importance of fast reroute and multi-protocol lambda switching is also discussed.

Constraint-based routing is one of the most important tools for traffic engineering. The issues related to constraint-based routing are discussed in detail. Heuristics for computing paths with propagation delay constraint, and for QoS routing on the top of virtual networks constructed in the traffic engineering process, are proposed. In order to reduce the computation complexity of constraint-based routing, an algorithm is proposed for reducing the routing table computation cost for OSPF. The correctness of the algorithm is illustrated. Similar idea can also be applied to IS-IS.

Traffic directing and load balancing have become very useful for improving performance for Web traffic. Such approaches, and their interactions with Diffserv and traffic engineering, are briefly discussed.

In summary, this dissertation not only presents a comprehensive discussion on all major issues related to QoS, but also describes a solution for providing QoS in the Internet. The traffic engineering part of this solution has been implemented and proved to work well.

Chapter 2 A FRAMEWORK FOR PROVIDING QOS IN THE INTERNET

The service quality of a network is reflected by the extent of user satisfaction of the network. Although also depending on application types and user perception, QoS can usually be measured by packet latency, jitter, packet loss rate and application throughput [6].

In this chapter, we first introduce two service models for providing QoS in the Internet. They are *integrated services* (Intserv) with the *resource reservation protocol* (RSVP) and *differentiated services* (Diffserv). They have been the traditional approaches for providing QoS in the Internet. We then argue that Intserv/Diffserv do not solve all problems, and additional approaches are useful or even necessary. A framework for providing QoS is proposed. This framework consists of traffic directing and load balancing at the application layer, Diffserv at the transport/network layer, traffic engineering and fast reroute at the network layer.

2.1 Introduction

Today's Internet only provides best effort service. With the rapid transformation of the Internet into a commercial infrastructure, demands for QoS have rapidly developed.

It is likely that several types of services will be demanded. One type will provide predictable service for companies that do business on the Web. Such companies are willing to pay a certain price to make their services reliable and to give their users a fast feel of their Web sites. This type of services may contain a single service class, or it may contain multiple classes such as *gold* and *silver* with decreasing quality. Another service type will provide low delay and low jitter services to real-time applications. Finally, best effort service will remain for those customers who only need connectivity [2].

Whether mechanisms are even needed to provide QoS is a debated issue. One opinion is that fiber and *dense wavelength division multiplexing* (DWDM) technologies will make bandwidth so abundant and cheap that high quality of service will be automatically delivered. The other opinion is no matter how much bandwidth the Internet can provide, new applications will be created to consume it. Therefore, mechanisms will still be needed to provide QoS. This argument is beyond the scope of this dissertation. Here we simply note that even if bandwidth will eventually become abundant and cheap, it is not the reality now. For now, some simple mechanisms are definitely needed in order to provide QoS on the Internet. All major router/switch vendors support this view by providing some QoS mechanisms in their products [17][18][19][20][21].

The *Internet engineering task force* (IETF) has proposed many service models and protocols for providing QoS in the Internet. Notably among them are the Intserv/RSVP and Diffserv. They are briefly reviewed in the subsequent sections.

2.2 Integrated Services and RSVP

The *integrated services* (Intserv) model [24] proposes two service classes in addition to best effort service. They are: 1) *guaranteed service* [26] for applications requiring fixed delay bound; and 2) *controlled load service* [27] for applications requiring reliable and enhanced best effort service. The philosophy of this model is that "there is an inescapable requirement for routers to be able to reserve resources in order to provide special QoS for specific user packet streams, or flows. This in turn requires flow-specific state in the routers" [24].

RSVP was invented as a signaling protocol for applications to reserve resources [28]. The signaling process is illustrated in Fig. 2-1. The sender sends a PATH message

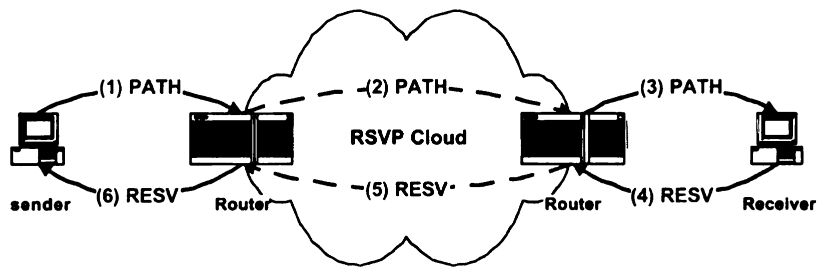


Figure 2-1 RSVP signaling

to the receiver specifying the characteristics of the traffic. Every intermediate router along the path forwards the PATH message to the next hop determined by the routing protocol. Upon receiving a PATH message, the receiver responds with a RESV message to request resources for the flow. Every intermediate router along the path can reject or accept the request of the RESV message. If the request is rejected, the router will send an error message to the receiver, and the signaling process will terminate. If the request is accepted, link bandwidth and buffer space are allocated for the flow and the related flow state information will be installed in the router.

Intserv is implemented by four components: the signaling protocol (e.g. RSVP), the admission control routine, the classifier and the packet scheduler. Applications requiring guaranteed service or controlled-load service must set up the paths and reserve resources before transmitting their data. The admission control routines will decide whether a request for resources can be granted. When a router receives a packet, the classifier will perform a *multi-field* (MF) classification and put the packet in a specific queue based on the classification result. The packet scheduler will then schedule the packet accordingly to meet its QoS requirements.

The Intserv/RSVP architecture is influenced by the work of Ferrari et al. [29]. It represents a fundamental change to the traditional Internet architecture, which is founded on the concept that all flow-related state information should be in the end systems [32].

Problems with the Intserv architecture are: 1) the amount of state information increases proportionally with the number of flows. This places a huge storage and processing overhead on the backbone routers. Therefore, this architecture does not scale well in the Internet core; 2) the requirement on routers is high. All routers must support RSVP, admission control, MF classification and packet scheduling; 3) ubiquitous deployment is required for guaranteed service. Incremental deployment of controlled-load service is possible by deploying controlled-load service and RSVP functionality at the bottleneck nodes of a domain and tunneling the RSVP messages over other part of the domain.

2.3 Differentiated Services

Because of the difficulty in implementing and deploying Intserv and RSVP, *differentiated services* (Diffserv) [38][39] is introduced. The essence of Diffserv is to divide traffic into multiple classes, and treat them differently, especially when there is a shortage of resources.

IPv4 header contains a *type of service* (TOS) byte. Its meaning is previously defined in [33]. Applications can set the left three bits in the TOS byte to indicate the need for low delay or high throughput or low loss rate service. However, choices are limited. Diffserv renames the TOS octet as *differentiated services field* (DS field) [43] and uses it to indicate the forwarding treatment a packet should receive. Diffserv also standardizes a number of *per-hop behavior* (PHB) groups [44][45]. Using different classification, policing, shaping and scheduling rules, several classes of services can be provided.

In order for a customer to receive differentiated services from its ISP, it must have a *service level agreement* (SLA) with its ISP. A SLA may explicitly or implicitly specify a *traffic conditioning agreement* (TCA) which defines classification rules as well as metering, policing, marking, and shaping rules [38]. A SLA can be static or dynamic. Static SLAs are negotiated on a regular basis, e.g. monthly and yearly. Customers with dynamic SLAs use a signaling protocol such as RSVP to request for services on demand.

Customers can mark the DS fields of their packets to indicate the desired service or have them marked by the ISP edge routers based on MF classification.

At the ingress of the ISP networks, packets are classified, policed and possibly shaped. The classification, policing and shaping rules used at the ingress routers are specified in the TCA. The amount of buffering space required is usually not considered explicitly. When a packet enters one domain from another domain, its DS field may be re-marked, as determined by the SLA between the two domains.

Diffserv only defines DS fields and PHBs. It is ISP's responsibility to decide what services to provide. Using the classification, policing, shaping and scheduling mechanisms, many services can be provided. For example,

1. *premium* service with reliable, low delay and low jitter delivery;
2. *gold* and *silver* services with reliable and timely delivery.

Diffserv is significantly different from Intserv. First, there are only a limited number of service classes indicated by the DS field. Since resources are allocated in the granularity of class, the amount of state information is proportional to the number of classes rather than the number of flows. Diffserv is therefore more scalable. Second, sophisticated classification, marking, policing and shaping operations are only needed at the edge of the networks. ISP core routers need only to implement *behavior aggregate* (BA) classification. Therefore, it is easier to implement Diffserv.

In the Diffserv model, incremental deployment is possible for gold and silver services. Diffserv-incapable routers simply ignore the DS fields of the packets and give all packets best effort service. Even in that case, because gold and silver packets are less likely to be dropped by Diffserv-capable routers, the performance of gold and silver traffic will still be better than best effort traffic.

2.4 A Framework for Providing QoS in the Internet

Diffserv is considered by many people as the scheme for providing QoS in the Internet. Traffic engineering, constraint-based routing, MPLS, and fast reroute are considered as something independent.

In this dissertation, we argue that Diffserv alone is not sufficient for providing QoS in the Internet, and propose a framework for doing so. This framework consists of traffic directing and load balancing at the application layer, Diffserv at the transport/network layer, and traffic engineering and fast reroute at the network layer. Here we briefly explain why the additional schemes are needed.

Diffserv essentially provide differentiated performance degradation (or no degradation) for different traffic when there is network congestion. If congestion can be avoided, performance of all traffic will be good even without Diffserv. Traffic engineering can avoid congestion caused by uneven traffic distribution. This is why traffic engineering is useful for providing QoS. There is also a problem that Diffserv does not solve. That is, even though policing and shaping are done at the edge, uneven distribution of traffic in the network may still cause concentration of high priority traffic at some routers. This can excessively affect performance of low priority traffic, and can even cause performance degradation of high priority traffic at those routers. This problem must be solved by traffic engineering. Therefore, traffic engineering is also necessary for providing QoS in the Internet.

Diffserv is only concerned with networks in normal condition. It does not adequately address issues caused by link or router failure. When a router or a link fails, traffic sent along an affected path will be lost. Therefore, it is critical to temporarily repair the affected path so that it can continue to carry traffic. This is the task of fast reroute [51][52][53].

Traffic directing is to avoid using the congested part of a network by choosing the sources of traffic from multiple geographically distributed alternatives based on some dynamically measured statistics. This is complementary to Diffserv. Load balancing is to use multiple servers (geographically close to each other) and distribute requests among them. The purpose is to increase service availability and to reduce the load of each server.

The framework for providing QoS in the Internet is summarized in Table 2.1.

Table 2-1 A framework for providing QoS

Location	**QoS scheme**	**Mechanisms**	**Purpose of the QoS scheme**
Application Layer	Traffic directing and load balancing	URL redirecting, load balancing	Direct traffic away from congesting part of a network or server
Transport/ Network Layer	Diffserv	Classification, policing, shaping, marking, class-based queueing and scheduling, Random Early Detection (RED)	Provide differentiated services for different classes of traffic, especially during network congestion
Network Layer	Traffic engineering	MPLS, constraint-based routing, LSP path signaling, and enhanced link state IGPs	Avoid congestion in the network
	Fast reroute	Local repair	Avoid packet loss during link and/or router failure

Chapter 3 DIFFERENTIATED SERVICES

In this chapter, we discuss how to provide *differentiated services* (Diffserv) in IP networks. We also propose a solution for resolving a conflict between Diffserv and TCP.

3.1 Introduction to Assured Service and Premium Service

In this section, a service architecture for Diffserv is presented [5]. This architecture provides three classes of services:

- *best effort*,
- *assured*, and
- *premium*.

Best effort is the traditional Internet service. Assured service and premium service are described next.

3.1.1 Assured service

Assured service provides reliable and predictable packet delivery. It is intended for non-real-time interactive applications such as Web browsing. Traffic in the assured class will exhibit the *assured-forwarding* (AF) *per-hop behavior* (PHB) [44]. Assured service can be further divided into two sub-classes, gold and silver.

Customers that need assured service should have *service level agreements* (SLAs) with their ISPs. The SLAs will specify the amount of bandwidth allocated for the customers. Customers are responsible for deciding how their applications share that amount of bandwidth. One possible service allocation process is described in the Section 3.2.2. SLAs for assured service are usually static, i.e., customers can start data transmission whenever they want without signaling their ISPs.

3.1.2 Premium service

Premium service provides reliable, low-delay and low-jitter service such as *virtual lease line* (VLL) service. It can be used for real-time applications or mission critical traffic such as network control. Traffic in the premium class will exhibit the *expedited-forwarding* (EF) PHB [45].

Each customer that subscribes to the premium service will have a SLA with its ISP. The SLA specifies a desired peak bit-rate from that customer. The customer is responsible for not exceeding the peak rate. Otherwise, excess traffic may be dropped. Alternatively, if an ISP is certain that there are enough resources to carry the extra premium traffic, the extra premium traffic can still be transmitted. The customer will be charged accordingly. Because ISP networks are usually over-provisioned, the second case is more likely.

Because premium service is more expensive than assured service, it is desirable for ISPs to support both static SLAs and dynamic SLAs. Dynamic SLAs allow customers to request for premium service on demand without subscribing to it. Signaling and admission control are needed for dynamic SLAs [40].

3.1.3 Terminology

The frequently used terms in this chapter are defined below.

Table 3-1 Frequently used terms in Diffserv

Flow	A stream of packets with the same source IP address, source port number, destination IP address, destination port number and protocol ID.
Service Level Agreement (SLA)	A service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization or another provider domain (upstream domain).
Traffic Profile	A description of the properties of a traffic stream such as rate and burst size.
Precedence Field	the three leftmost bits in the TOS octet of an IPv4 header. Note that in Diffserv, these three bits may or may not used to denote the precedence of the IP packet.
TOS Field	bits 3-6 in the TOS octet of IPv4 header [34].
Differentiated Services field (DS field)	the TOS octet of an IPv4 header, or the traffic class octet of an IPv6 header, is renamed the differentiated services field by Diffserv. It is the field where service classes are encoded
Per-Hop-Behavior (PHB)	The externally observable forwarding treatment of a class of packets at a Diffserv-compliant node
Mechanism	A specific algorithm or operation (e.g., queuing discipline) that is implemented in a router to realize a set of one or more per-hop behaviors.
Admission Control	The decision process of whether to accept a request for resources (link bandwidth plus buffer space)
Classification	The process of sorting packets based on the content of packet headers according to defined rules.
Behavior Aggregate (BA) Classification	The process of sorting packets based only on the contents of the DS field.
Multi-Field (MF) Classification	The process of classifying packets based on the content of multiple fields such as source address, destination address, TOS byte, protocol ID, source port number, and destination port number.
Marking	The process of setting the DS fields of packets
Policing	The process of handling out of profile traffic, e.g., discarding excess packets
Shaping	The process of delaying packets within traffic stream to cause it to conform to some defined traffic profile.

Scheduling	The process of deciding which packet to send first in a system of multiple queues
Queue Management	Controlling the length of packet queues by dropping packets when necessary or appropriate

The structure of the TOS octet is depicted below:

0	1	2	3	4	5	6	7
Precedence			TOS				MBZ

MBZ field: Must Be Zero, the right-most bit in the TOS octet.

The structure of the DS field is depicted below:

0	1	2	3	4	5	6	7
DSCP						CU	

DSCP: Differentiated Service Code Point, the leftmost 6 bits in the DS field.

CU: currently unused.

3.2 An Approach for Providing Assured Service and Premium Service

In this section we describe how to provide assured and premium services. We first describe the mechanisms needed in ISP networks. We then describe an approach for customers to allocate the service they get from their ISPs to their hosts/applications, and an approach for ISPs to allocate resources in their network so that the contracted service can be delivered. The requirements on the routers, and the scalability of this approach, are then discussed.

Because there are only 3 classes of traffic (or 4 classes if assured service is divided into gold and silver), only the first 3 bits of the DSCP, i.e., the former precedence field, are used to denote the traffic class that each packet belongs to and the drop preference of the packet. The other 3 bits of the DSCP are set to 0. In other words, only the 8 class selector code points are used. For each output port, one queue is used for each class. Of the 3 bits used, the first two bits are used to denote traffic classes (and thus for selecting the queue associated with those classes), the third bit is used to indicate the drop preference inside each class.

3.2.1 Mechanisms for providing assured service and premium service

In this approach, packets are classified, policed, marked and possibly shaped at the ISP edge routers. Besides, all routers (edge and core) queue and schedule the packets according to their DSCPs. Traffic may be shaped again before being sent out to the next domain. These processes are described in this section.

3.2.1.1 Classification at the ingress

Classification is mainly based on the incoming interface. This incoming interface can be physical like a *packet-over-SONET* (POS) interface, or it can be logical like an ATM VC or an Ethernet *virtual LAN* (VLAN). Besides the incoming interface, source and destination IP addresses, source and destination port numbers, protocol ID and the TOS octet can be further used in the classification. This is called *multi-field* (MF) classification. By considering the incoming interface first, the classification rules for traffic from one interface can be separated from rules for traffic from other interfaces. This greatly reduces the number of rules that need to be considered, and makes the MF classification much faster. From the classification result, traffic profiles and the corresponding policing, marking and shaping rules of the incoming packets can be derived. Policing, marking and shaping can then be performed on the traffic based on the SLA.

3.2.1.2 Policing, marking and shaping at the ingress

Policing is to determine whether traffic from a particular customer conforms to the traffic profile specified in the SLA, and take action accordingly. Marking is to set the DSCPs of the packets. The policing and the marking processes of assured and premium traffic are described below.

For customers with assured service, policing should be implemented with a token bucket so that some kind of burst is allowed. When a packet arrives and there are tokens in the bucket, the packet is considered as in profile. The DSCP of the packet is set to 101000 by the marking process. If a packet arrives and there is no token in the bucket, then the packet is considered as out-of-profile. The marking process will set the DSCP of the packet to 100000. Note that the first two bits are the same for both in-profile packets and out-of-profile packets, because both types of packets belong to the same traffic class. But the third bit is different, so that in-profile and out-of-profile packets will have different drop preference. The *random early detection* (RED) algorithm then determines whether to drop a particular packet or to queue it. No packets with DSCP 101000 will be dropped before all 100000 packets are dropped. RED will be detailed in the next section. All the packets that are not dropped, no matter they are in or out of profile, are put into the same queue to avoid out of order delivery. We call this queue *assured queue* (AQ).

For customers with premium service, policing should be implemented with a leaky bucket so that burst will not be introduced, or with a token bucket that allows only small burst. When a packet arrives and there are tokens in the bucket, the packet is considered as conformant. The DSCP of the packet is set to 111000. If a packet arrives and there is no token in the bucket, then the packet is considered as non-conformant. Depending on the SLA between the ISP and the customer (can be another ISP), non-conformant packets may be dropped immediately or may be transmitted and accounted for extra charge. If non-conformant packets are to be transmitted, their DSCPs will also be set to 111000. All packets are put into a queue called *premium queue* (PQ).

Best effort traffic may also be policed, depending on the SLAs between customers and providers. In-profile packets will have DSCP 001000, and out-of-profile packets will have DSCP 000000. Alternatively, no policing is done for best effort traffic and all packets have their DSCPs set to 000000. RED is applied, and packets that are not dropped enter a queue called *default queue* (DQ).

Because the PQ, AQ and DQ all have fixed output rate, premium, assured and best effort traffics are all shaped.

3.2.1.3 Queue management and packet scheduling at the ingress and the core

After a packet is marked, its treatment is solely determined by its DSCP, no matter at an edge router or at a core router. The handling of each packet is described below.

For assured and best effort traffic, RED is first applied to determine whether to drop the packet. RED is a buffer management scheme. RED drops packets randomly based on their DSCPs and the average queue length. The purpose of RED is to avoid queue overflow and tail-drop (when a queue overflows, all arriving packets are dropped) at each router. By dropping packets randomly, packets of different TCP connections are likely to be dropped at different time. Therefore, the TCP flow control mechanism for these connections will reduce their send rate at different time. This helps to prevent the router's queues from overflowing, and thus avoid the tail-drop. Otherwise, tail-drop will trigger many TCP flows to decrease and later increase their rate simultaneously. It causes traffic load to oscillate and can hurt performance significantly. RED has been proved to be useful.

RED with in and out (RIO) is a more advanced buffer management scheme. It basically maintains two RED algorithms, one for in-packets and one for out-packets. There are two thresholds for each queue. When the average queue length is below the first threshold, no packets are dropped. When the queue length is between the two thresholds, only out-packets are randomly dropped. When the queue length exceeds the second threshold, indicating possible network congestion, all out-packets are dropped, and then some in-packets may also be randomly dropped. The packet drop probability can be a step function or a linear function of the queue length.

In addition to preventing the TCP flow-control synchronization, RIO prevents, to some extent, greedy sources from hurting the performance of other assured traffic by dropping the out-packets more aggressively. However, RED/RIO is not effective in controlling unresponsive sources like applications using UDP.

For premium traffic, neither RED nor RIO is applied, because it is undesirable to drop premium packets. All packets are put into the PQ.

The PQ, AQ and DQ are scheduled by an algorithm called *weight fair queueing* (WFQ) or a variant of it. The rate of these queues is set by network administrators based on traffic statistics and domain policy. If we define the provisioning factor of a queue, *pf*(queue), as the ratio between the configured output rate and the actual input traffic rate, then the following order must be maintained:

$pf(\text{PQ}) > pf(\text{AQ}) > pf(\text{DQ}) > 1.0.$

How to set the provisioning factor for these queues is described in Section 3.2.3.

3.2.1.4 Re-shaping of premium traffic at the egress

Depending on the SLA between an ISP and its downstream domain (can be another ISP), the current ISP may need to shape its premium traffic going to the downstream domain. The downstream domain will police the premium traffic from the current ISP. Excess traffic may be dropped, or be transmitted and accounted for extra charge. Alternatively, the current ISP may send its premium traffic downstream without shaping it, if the SLA allows it to do so. The downstream domain will then shape the premium traffic from the current ISP.

3.2.1.5 Realization of assured service and premium service

This section explains how assured and premium services can be realized by the classification, policing, marking, buffer management and packet scheduling mechanisms described above.

For assured traffic, because it is separated from best effort traffic, in-profile packets will have low loss rate (or no loss) and timely delivery, even when the network is congested with best effort traffic. Therefore, customers will perceive a reliable and responsive network service if they keep their traffic conformant to the traffic profile.

When there is no congestion, out-of-profile packets will also be delivered. Network resources are thus better utilized.

Assured service can be replaced by gold and silver services for more service classes. The handling of gold and silver traffic will be exactly the same as the handling of assure traffic, except that

- the DSCPs of the gold and silver packets are marked differently. Gold packets will have DSCP 10x000, silver packets will have DSCP 01x000, where x can be either 1 or 0, depending on whether the packet is in profile or out of profile.
- Gold packets will enter a *gold queue* (GQ) and silver packets a *silver queue* (SQ). The following order will be maintained: *pf*(GQ) > *pf*(SQ) > *pf*(DQ) > 1.0.

For premium traffic, first, traffic is shaped or dropped at the ingress routers of the networks. Non-conformant flows cannot affect performance of conformant flows. Second, the output rate of the PQ is configured to be significantly higher than the input rate (It is advisable to have *pf*(PQ) > 2.0). Therefore, arriving premium packets will find the PQ empty or very short most of the time. The delay or jitter experienced by premium packets will be very low. But it should be noted that premium service provides only probabilistic guarantee on the delay or jitter bound.

However, uneven distribution of premium or assured traffic may cause a problem for Diffserv. In ISP networks, aggregation of traffic from the boundary routers to a core router, e.g. CR1 in Fig. 3-1, is inevitable. But this is not a problem because the output link to the core is usually much faster than the input links from customers. However, aggregation of premium/assured traffic that is caused by unexpected traffic shift, as showed at CR4 in Fig. 3-1, may invalidate the fundamental assumption that the arrival rate of premium/assured traffic is well below the service rate, and cause performance of the premium/assured traffic to degrade. Diffserv alone cannot avoid this problem. Traffic engineering must be used to avoid such congestion caused by uneven traffic distribution. It is discussed in Chapter 4.

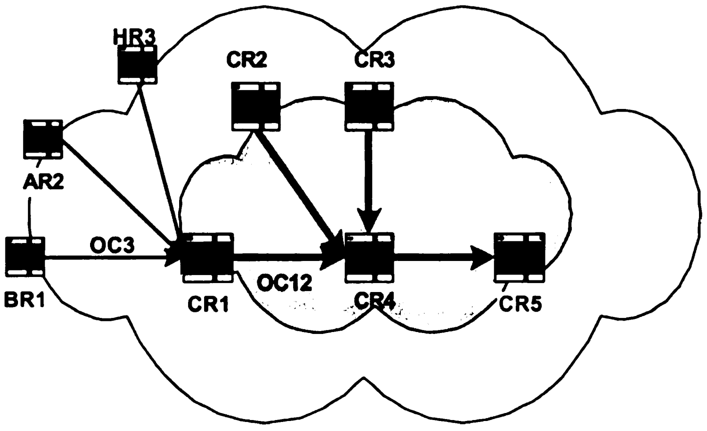


Figure 3-1 Uneven traffic distribution

3.2.2 Service allocation in customer domains

Given a SLA, it is the customer's responsibility to decide how its hosts/applications share the service (e.g., 10 Mbps of premium traffic) specified in the SLA. This process is called *service allocation*.

At the early stage of Diffserv deployment, hosts in customer domains will send their traffic in the usual way, i.e., unmarked. All traffic is treated equal inside the customer domain, and by the ISP before policing and marking, regardless of the sending hosts and applications. The treatment that a packet from that customer will receive is determined by the policing result (i.e., in or out of profile) at the ISP ingress router. At this stage, a customer has no capability to allocate its service differently to different hosts/applications.

If a customer domain decides to differentiate traffic from different hosts and applications, there are basically two choices: 1) Each application makes its own decision on which service to use, and set the DSCPs of its packets. There is no central controller. 2) A resource controller coordinates the resource usage of all applications. A resource controller is a device that grants or denies requests for services based on domain policies. The resource controller in a customer domain is most likely the exit router (CPE) itself. The domain policies can be stored in the resource controller itself, or they can be stored in a separate directory server. In the second case, the resource controller will use the *light-weight directory access protocol* (LDAP) [55] to retrieve the policies periodically. Since all hosts must cooperate to share the service specified in the SLA, it is technically better to have a resource controller. A service allocation scheme is described below.

Each user/application will decide which service to use and set the DSCPs of its packets. Since many different applications may need the capability of setting the DSCPs of their packets, it is recommended that the *operating system* (OS) of the hosts provide the *application programming interface* (API) to all applications. If the application or the OS does not have the capability to set the DSCPs of its packets, then this step can be skipped. The CPE will classify the packets and act as the resource controller to decide which service the application flow should receive. The DSCPs of the packets can be reset to a new value by the CPE if necessary. The advantage of this approach is that it is simple and easy to implement. A slight disadvantage of this approach is that applications do not know whether the CPE will reset the DSCPs of their packets or not. This can be a problem for real-time applications.

In order to solve the above problem, signaling capability must be added between the applications and the resource controller. Because many applications may need this signaling capability, it is recommended that OS provide such mechanism. The signaling protocol can be RSVP or some other protocols. With the signaling capability, when the resource controller decides to reset the DSCPs of an application's packets, it can signal the application. The user can then decides whether to stop or continue with best effort service.

Alternatively, signaling can be done to decide which service an application can use before actual data transmission. For example, if an application needs premium service, it must request premium service from the resource controller first. The resource controller can then grant or deny the request. Whether signaling is needed before data transmission depends on how fast the resource controller can make the admission control decision. If the admission control decision can be made very quickly, then the first data packet can serve as signaling. Otherwise, signaling must be done before data transmission, or the first few data packets may be significantly delayed. From a practical perspective, because admission control decision is made by the control card in the CPE instead of a line card, it is advisable to do signaling before data transmission.

If the SLA between a customer domain and its ISP is dynamic, the customer domain must signal its ISP for service. This can be done by communication between administrators in both domains or by some signaling protocol.

3.2.3 Resource provisioning in ISP domains

Given the SLAs, an ISP must decide how to configure its routers so that they know how to handle the incoming traffic. One important part of this task is to configure the rate (in terms of percentage of link capacity) and size for the PQ, AQ, DQ, and the RIO parameters for the AQ and DQ. We call this process *resource provisioning.*

We first discuss resource provisioning in ISPs that only support static SLAs.

For edge routers, classification, policing and shaping rules can be derived from the traffic profiles in the SLAs. However, rate and size for the PQ, AQ, DQ and parameters for RIO cannot be derived analytically from the traffic profiles. The reason is that the edge router may have multiple interfaces to multiple core routers. Therefore, even though the total amount of incoming traffic in each class can be derived by adding all customer traffic in that class together, the distribution of traffic in each class among the multiple links to the core routers is unknown. For example, even if it is known that there is a total of 100 Mbps of premium traffic, it is generally unknown how much of this 100 Mbps is destined to Seattle and how much is destined to New York City. Therefore, the rate, size, and RIO parameters of the queues for interfaces to the core routers may have to be set based on some measured statistics. Because of this, it is very desirable that routers be equipped with commands for showing the instantaneous or average (over a configurable time period) rate and length of the queues. When there is a topology change in the network, the distribution of traffic among these interfaces may change. Even when there is no topology change, traffic distribution can change over time. All these make the task of setting the parameters complicated.

For core routers, the task of resource provisioning is more difficult, because even the amount of incoming traffic in each class is unknown. Again, the problem arises because traffic distribution among destinations is unknown and can change over time. Therefore, the rate and size for the queues, and the parameters for RIO, must be configured based on some measured statistics.

One approach to setting the rate, size and RIO parameters is as follows. The arrival rate of each queue is polled via the *simple network management protocol* (SNMP) every 5 minutes. The 95th-percentile of the all measured rates over the past 2 hours, inflated by the provisioning factor *pf*, is then used as the output rate of the queue. The value of *pf* is determined by the corresponding class and maybe by the domain policy as well. For example, *pf* can be set to 2.0 for PQ, 1.5 for AQ, and 1.2 for DQ. From the three inflated output rate r(q) of the three queues, the percentage of link capacity each queue should get can be computed in the following way:

Percentage of link capacity of queue q = r(q) / [r(PQ) + r(AQ) + r(DQ)],

where q is either PQ or AQ or DQ.

From the traffic profile, the output rate of the queue, and other requirements such as maximum queueing latency, the maximum size of the queue can then be determined as follows:

Max queue size = output rate of the queue × Max queueing latency.

From the maximum queue size and average queue size (known from statistics), RIO parameters can be derived [124].

The above parameters may be changed periodically, e.g. hourly, to track normal traffic shift.

The above resource-provisioning scheme requires the measured arrival rates of the queues. If the arrival rates are not available but the current output rate and average queue length are available, then the new rate of the queue can be determined in the following way.

If the average queue length of a queue is longer than desired, then a factor larger than 1.0 should be multiplied to the measured rate. The value of the factor should be determined by how much we want to accelerate the speed of that queue. If the average queue length of a queue is shorter than desired, then a factor smaller than 1.0 can be applied. After this normalization,

Percentage of link capacity of queue q = r(q) / [r(PQ) + r(AQ) + r(DQ)],

where q is either PQ or AQ or DQ, and r(q) is the normalized output rate of queue q.

It should be noted that it is not always good to give the PQ a high percentage of link capacity, because it makes the premium traffic more bursty and add to the burden of the downstream nodes.

If dynamic SLAs are also supported, resource provisioning becomes even more complicated.

First, resource provisioning becomes related to the signaling process. When a request for resources arrives, the ISP edge router makes an admission control decision based on domain policy and resource availability. If the request is granted, the edge routers will be configured with corresponding classification, policing, marking and shaping rules. The rate, size and RED/RIO parameters for the queues may have to be changed more frequently, but not every time a request arrives or departs.

Second, because additional traffic shifts are introduced by dynamic SLAs, estimating the exact amount of traffic in each class becomes more difficult. However, the configuration parameters for core routers should still be based on measured statistics, e.g. the 95th-percentile of measured rates over the past two hours, and should not be changed frequently. Specifically, core routers should not have to change its configuration parameters with the arrival and departure of each dynamic resource request.

Resource provisioning is a difficult task. From the above discussions, it is clear that Diffserv does not provide sufficient mechanisms to prevent hot spots of high priority traffic. Therefore, if a network is not planned and controlled carefully, Diffserv alone may not always be able to deliver the contracted quality of service for high priority traffic. This process will be even more complex if the number of traffic classes is increased.

If a network has sufficient capacity, then setting the queue rate, size and RED parameters accurately becomes less important. Besides, the network can handle bursty premium/assured traffic and traffic shift more easily. This makes network operation easy and reduces the associated operation cost. This factor should be considered in the network planning process.

3.2.4 Requirements on routers for providing assured and premium services

The requirements on routers for providing assured and premium services are summarized below.

1. ISP edge routers (PEs) need to support MF classification, policing, marking and shaping.

2. All ISP routers need to implement BA classification, RED/RIO, and at least three queues with WFQ scheduling. Some commands for showing the input/output rate and the average length of the queues are highly desirable.

Optionally,

3. If dynamic SLAs are supported, signaling and admission control mechanisms are needed.

If assured service is replaced by gold and silver services, then the AQ must be replaced by a GQ and a SQ. WFQ is needed to schedule the PQ, GQ, SQ and DQ.

3.2.5 Scalability of Diffserv in the Internet core

The scheme described in this section for providing assured service and premium service is a scalable scheme. This is because:

1. The amount of state information that core routers need to maintain is proportional to the number of traffic classes, not the number of application flows;
2. Complex classification, policing, marking and shaping only happen at the edge of the network;
3. Core routers only need to classify packets based on their DSCPs (BA classification), perform RED/RIO if necessary, and use WFQ to schedule the PQ, AQ and DQ.

At the edge of the network, link speed is usually low. Therefore the edge routers can spend more time on complex classification, policing, marking and shaping. At the core of the network, link speed is high. The core routers must be kept simple in order to be able to forward packets quickly. The scheme described in this section meets these requirements well.

3.2.6 Examples of End-to-End Service Delivery

Example 1: delivery of assured service with a static SLA

In Fig. 3-2, host S in customer network #1 (CN1) wants to use assured service to send data to host D in customer network #2 (CN2). CN1 has a static SLA with ISP1. The service delivery process is described below. The numbers inside the circles are the step numbers in the service delivery process.

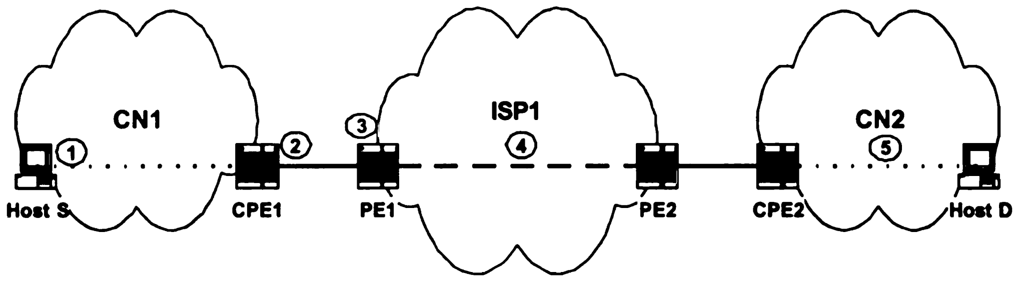


Figure 3-2 The delivery process of assured service with a static SLA

1. Host S sends its packets to the local exit router CPE1. Optionally, the DSCPs of the packets can be set by Host S.
2. CPE1 classifies the packets based on source and destination IP addresses, source and destination port numbers, protocol ID and possibly the incoming interface. CPE1 then checks domain policies to decide whether to reset the DSCPs of the packets in that stream to a new value. If service allocation is not needed, then these classification and policing processes can be skipped, and PE1 will allocate the assured service to CN1's

traffic on a first come first serve basis in Step 3. CPE1 then forwards the packets to ISP edge router PE1.

3. PE1 performs a MF classification based on the incoming interface, DSCPs, and possibly other fields of the packets. PE1 then polices the traffic stream and re-marks the packets if necessary. In-profile packets will have DSCP 101000, out-of-profile packets will have DSCP 100000. RIO is applied to the packets. If the packets are not dropped by RIO, they are put into the AQ and scheduled accordingly. PE1 also does the metering/accounting so that billing can be done later.
4. All routers between PE1 (excluded) and PE2 (included) perform BA classifications, apply RIO, and perform WFQ on the queues.
5. CPE2 forwards the packets to host D.

Note that:

- If there are multiple ISPs between CN1 and CN2, then Step 4 will be repeated multiple times, once per ISP.
- If CN1 does not have a SLA with ISP1, it can only send traffic as best effort. No matter what DSCPs are set inside CN1, they will be reset by PE1.

Example 2: Delivery of premium service with a dynamic SLA

In Fig. 3-3, host S in customer network #1 (CN1) wants to use premium service to send

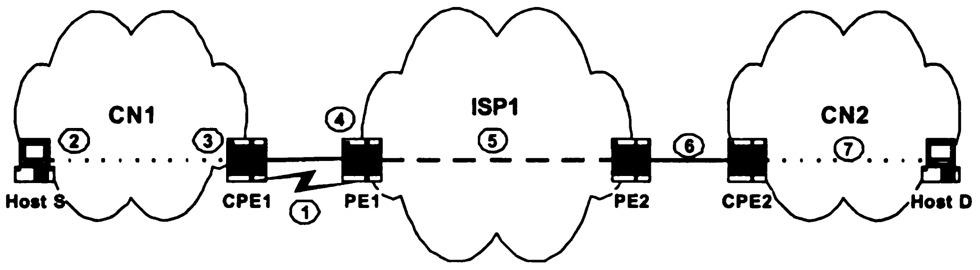


Figure 3-3 The delivery process of premium service with a dynamic SLA

data to host D in customer network #2 (CN2). CN1 has a dynamic SLA with ISP1.

Phase 1: signaling

1. Either the administrator in CN1 communicates with the administrator in ISP1 to request for the service, or CPE1 (acting as the resource controller in CN1) sends a signaling message to PE1 to request for the service. Theoretically, the request can be granted or denied. If the request is granted, some policy rules will be added to CPE1 specifying which applications are allowed to use premium service. Some rules will also be added to PE1 so that it knows how to police the premium traffic from CPE1. If the request is denied, the whole service process stops. In reality, a dynamic request for premium service may never be denied because this is an important profit source for ISPs. The main purpose of signaling is to notify the ISP to configure its edge router.

Note that:

- It is assumed that static SLAs will always be used between ISPs. Therefore, even if there are multiple ISPs between CN1 and CN2, there is no need for the signaling process to proceed end to end.
- If the SLA between CN1 and ISP1 is static, then the signaling process is not needed.

Phase 2: Data Transmission:

2. Host S sends its packets to the local exit router CPE1. Optionally, the DSCPs of the packets can be set by Host S.
3. CPE1 classifies the packets based on source and destination IP addresses, source and destination port numbers, protocol ID and possibly the incoming interface.

CPE1 then checks domain policies to decide whether to reset the DSCPs of the packets in that stream to new values. CPE1 then forwards the packets to ISP edge router PE1.

4. PE1 performs a MF classification based on the incoming interface, DSCPs, and possibly other fields of the packets, and polices the traffic. If the premium traffic is non-conformant to the profile, either the stream is shaped, or excess premium packets are dropped, or excess premium packets are sent and accounted for extra charge. After that, all premium packets will have their DSCPs set to 111000, be put into the PQ, and be scheduled accordingly. PE1 will also do the metering/accounting so that the information can be used for billing later.
5. All routers between PE1 (excluded) and PE2 (included) perform BA classifications and WFQ on the queues.
6. PE2 forwards the packets to CPE2.
7. CPE2 forwards the packets to host D.

3.3 Resolving a Conflict between Diffserv and TCP

This section describes a conflict between TCP [56] and Diffserv on the use of the three leftmost bits in the TOS octet of an IPv4 header [57]. In a network that contains Diffserv capable nodes, such a conflict can cause failures in establishing TCP connections or can cause some established TCP connections to be reset undesirably. A solution to this problem is provided by modifying TCP to ignore the precedence field. This solution has been published by the IETF as a proposed standard [59].

3.3.1 Introduction to the conflict between Diffserv and TCP

In TCP, each connection has a set of states associated with it. Such states are reflected by a set of variables stored in the TCP control block (TCB) of both ends. Such variables may include the local and remote socket number, precedence of the connection, security level and compartment, etc. Both ends must agree on the setting of the precedence and security parameters in order to establish a connection and keep it open [60].

There is no field in the TCP header that indicates the precedence of a segment. Instead, the precedence field in the header of the IP packet is used as the indication. The security level and compartment are likewise carried in the IP header, but as IP options rather than a fixed header field. Because of this difference, the problem with precedence discussed in this dissertation does not apply to them.

TCP requires that the precedence (and security parameters) of a connection must remain unchanged during the lifetime of the connection. Therefore, for an established TCP connection with precedence, the receipt of a segment with different precedence indicates an error. The connection must be reset [RFC793, page 37].

With the advent of Diffserv, intermediate nodes may modify the differentiated services code-point (DSCP) of the IP header to indicate the desired per-hop behavior (PHB). The DSCP includes the three bits formerly known as the precedence field. Because any modification to those three bits will be considered illegal by endpoints that are precedence-aware, they may cause failures in establishing connections, or may cause established connections to be reset.

It should be noted that this conflict exists well before Diffserv is standardized, because it is a common practice for routers to manipulate the precedence field to achieve some forwarding behavior before Diffserv was introduced [60].

3.3.2 Problem description

The manipulation of the DSCP to achieve the desired PHB by Diffserv-capable nodes may conflict with TCP's use of the precedence field. This conflict can potentially cause problems for TCP implementations that conform to RFC 793. First, page 36 of RFC 793 states:

"If the connection is in any non-synchronized state (LISTEN, SYN-SENT, SYN-RECEIVED), and the incoming segment acknowledges something not yet sent (the segment carries an unacceptable ACK), or if an incoming segment has a security level or compartment which does not exactly match the level and compartment requested for the connection, a reset is sent. If our SYN has not been acknowledged and the precedence level of the incoming segment is higher than the precedence level requested then either raise the local precedence level (if allowed by the user and the system) or send a reset; or if the precedence level of the incoming segment is lower than the precedence level requested then continue as if the precedence matched exactly (if the remote TCP cannot raise the precedence level to match ours this will be detected in the next segment it sends, and the connection will be terminated then). If our SYN has been acknowledged (perhaps in this incoming segment) the precedence level of the incoming segment must match the local precedence level exactly, if it does not a reset must be sent."

This leads to problem #1:

For a precedence-aware TCP module, if during TCP's synchronization process, the precedence fields of the SYN and/or ACK packets are modified by the intermediate nodes, resulting in the received ACK packet having a different precedence from the precedence picked by this TCP module, the TCP connection cannot be established, even if both modules actually agree on an identical precedence for the connection.

Then, on page 37, RFC 793 states:

"If the connection is in a synchronized state (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), ... [unrelated statements snipped] If an incoming segment has a security level, or compartment, or precedence which does not exactly match the level, and compartment, and precedence requested for the connection, a reset is sent and connection goes to the CLOSED state."

This leads to problem #2:

For a precedence-aware TCP module, if the precedence field of a received segment from an established TCP connection has been changed en route by the intermediate nodes so as to be different from the precedence specified during the connection setup, the TCP connection will be reset.

Each of problems #1 and #2 has a mirroring problem. They cause TCP connections that must be reset according to RFC 793 not to be reset.

Problem #3: A TCP connection may be established between two TCP modules that pick different precedence, because the precedence fields of the SYN and ACK packets are modified by intermediate nodes, resulting in both modules thinking that they are in agreement for the precedence of the connection.

Problem #4: A TCP connection has been established normally by two TCP modules that pick the same precedence. But in the middle of the data transmission, one of the TCP modules changes the precedence of its segments. According to RFC 793, the TCP connection must be reset. In a Diffserv-capable environment, if the precedence of the segments is altered by intermediate nodes such that it retains the expected value when arriving at the other TCP module, the connection will not be reset.

3.3.3 Proposed modification to TCP

The proposed modification to TCP is that TCP must ignore the precedence of all received segments. More specifically:

(1) In TCP's synchronization process, the TCP modules at both ends must ignore the precedence fields of the SYN and ACK packets. The TCP connection will be established if all the conditions specified by RFC 793 are satisfied except the precedence of the connection.

(2) After a connection is established, each end sends segments with its desired precedence. The two ends' precedence may be the same or may be different (because precedence is ignored during connection setup time). The precedence fields may be changed by the intermediate nodes too. They will be ignored by the other end. The TCP connection will not be reset.

Problems #1 and #2 are solved by this proposed modification. Problems #3 and #4 become non-issues because TCP must ignore the precedence. In a Diffserv-capable environment, the two cases described in problems #3 and #4 should be allowed.

The proposed modification to TCP is in conformance with TCP's design philosophy. In RFC 793, page 36, it is stated that:

"As a general rule, reset (RST) must be sent whenever a segment arrives which apparently is not intended for the current connection. A reset must not be sent if it is not clear that this is the case."

With the deployment of Diffserv, the precedence field can be modified by intermediate network nodes. A change in the precedence of a received segment does not necessarily indicate that the segment is not intended for the connection. Therefore, a RST must not be sent based solely on the change of the precedence of a received segment, no matter whether the TCP connection is in a non-synchronized state or synchronized state.

3.3.4 Security impact of the proposed modification to TCP

The RST generation rules given on page 37 of RFC 793 appear to indicate that the reception of any segment with an incorrect precedence field terminates a connection. If this is true regardless of the correctness of the sequence numbers in the segment's header, then the RFC 793 rules present a serious denial-of-service threat, as all an attacker must do to terminate a connection is guess the port numbers and then send two segments with different precedence values; one of them is certain to terminate the connection. Accordingly, the change to TCP processing proposed in this memo would yield a significant gain in terms of TCP resilience.

On the other hand, the stricter processing rules of RFC 793 in principle make TCP spoofing attacks more difficult, as the attacker must not only guess the victim TCP's initial sequence number, but also its precedence setting.

3.4 Conclusion

In this chapter, we propose a framework for providing differentiated services in IP-based networks. Three classes of services are provided, i.e., premium service, assured service and best effort service. In this approach,

1. Customers negotiate SLAs with ISPs. The SLAs specify what services the customers will receive. SLAs can be static or dynamic. For static SLAs, customers can transmit data at any time. For dynamic SLAs, customers must request for the desired service before transmitting data.
2. With service allocation, a customer domain decides how its hosts and applications share the services specified by the SLAs. With resource provisioning, an ISP domain decides how its routers should be configured so that the contracted services can be delivered.
3. Packets are classified and have their DSCPs set at the edge of a network. Subsequent forwarding treatment is solely based on the DSCPs of the packets. In order to deliver premium service and assured service, the following mechanisms are needed in ISP networks:
    - At the ingress routers: classification, policing, marking and shaping;
    - At all routers: BA classifications, RED/RIO and WFQ scheduling for the PQ, AQ and DQ.

One of the most important tasks in providing Diffserv is to set the output rate and size for the queues. Although the amount of ingress traffic in each class is known at the network edge, the distribution of the traffic in the network is generally unknown, unless some kinds of statistics are available. Besides, the traffic distribution can vary over time.

Therefore, it is difficult to set the output rate for the queues, especially at the core. In this dissertation, we propose that the output rate of the queues are set according to measured statistics, and updated periodically. From the output rate, the input rate, and the desired queueing delay of the packets in each queue, the queue size can then be derived. It is not advisable to set the service rate of a high priority queue much higher than its input rate, because this can introduce burst of high priority traffic for the downstream router.

Another important task of providing Diffserv is to avoid hot spots of high priority traffic. Since Diffserv has to provide preferable treatment for high priority traffic at the expense of low priority traffic during traffic congestion, the service quality of high priority traffic may be affected if there is a hot spot of high priority traffic. Diffserv does not provide sufficient mechanisms to avoid hot spots. Therefore, traffic engineering is necessary for providing QoS in the Internet.

Diffserv has a conflict with TCP on the use of the precedence field in the IP header. Because Diffserv-capable routers may change the precedence fields of packets en route, TCP connections can be reset undesirably for TCP stacks that strictly implement RFC793. We describe the problem and propose a solution to resolve this conflict. The solution is to modify TCP to ignore the checking of the precedence field.

Chapter 4 TRAFFIC ENGINEERING AND FAST REROUTE

This chapter discusses *traffic engineering* and *fast reroute*.

Traffic engineering is an iterative process of network planning and network optimization [8]. The purpose of traffic engineering is to optimize resource efficiency and network performance. Traffic engineering is important for providing QoS in the Internet because, Diffserv essentially provides differentiated performance degradation for different classes of traffic during network congestion. When there is no congestion, network performance will be good even without Diffserv. This means that traffic engineering is very useful for providing QoS. Traffic engineering is also needed to prevent concentration of high priority traffic. If there is concentration of high priority traffic, when there is network congestion, Diffserv cannot provide high priority traffic graceful performance degradation (or no degradation) at the expense of low priority traffic. Therefore, the contracted quality of high priority traffic may be violated [5]. Constraint-based routing, *multi-protocol label switching* (MPLS), an enhanced link state IGP and a path signaling protocol are useful tools for traffic engineering [63].

Fast reroute is to temporarily repair an affected path so that it can continue to carry traffic before a more optimal path is computed and used to carry traffic. Fast reroute is important for providing QoS because, when a router or a link fails, it takes *interior* *gateway protocols* (IGPs) from seconds to minutes to propagate information and re-compute paths. During this time interval, traffic sent along an affected path will experience high latency because of buffering, or will be lost. Higher link speed means more lost traffic. Therefore, it is important that the affected path is temporarily repaired and used to carry traffic before a more optimal path is computed [51][52][53]. This is especially important for providing QoS for real-time and other high priority traffic.

In this chapter, we first briefly review MPLS, constraint-based routing and enhanced link state IGPs to provide a background for traffic engineering. We then discuss the general issues of designing an MPLS system for traffic engineering. The design of GlobalCenter's MPLS system is presented and its performance is evaluated. Based on the experiences, a generic procedure for deploying MPLS system is proposed [9]. We then discuss the importance of network planning [10] and *multi-protocol lambda switching* (MPLmS) [16]. The inter-domain issues of traffic engineering are discussed, and an approach for systematic inter-domain traffic engineering is described.

Besides traffic engineering, an approach for providing fast reroute for high priority traffic is briefly discussed. And an MPLS-based service architecture for providing Diffserv is described.

4.1 Introduction

Traffic engineering is an iterative process of network planning and network optimization so as to optimize resource efficiency and network performance. Network planning is to improve the architecture (topology and link capacity) of a network in a systematic way so that the network is robust, adaptive and easy to operate. Network optimization is to control the mapping and distribution of traffic over the existing network infrastructure to avoid and/or relieve congestion, to assure satisfactory service delivery, and to optimize resource efficiency. In the first four sections of this chapter, we focus on network optimization. Network planning is discussed in Section 4.4.

Network optimization is needed because, no matter how well a network is designed, random incidents such as fiber cuts or changes in traffic demand will occur. When they occur, they can cause congestion and other problems to manifest in an operational network. This is aggravated by the fact that current IGPs always use the shortest paths to forward traffic. Using shortest paths conserves network resources, but it may also cause the following problems.

1. If traffic from a source to a destination exceeds the capacity of the shortest path, the shortest path will become congested while a longer path between these two nodes is under-utilized;
2. At a link that multiple shortest paths from different sources overlap, if the total traffic from different sources exceeds the capacity of the link, congestion will occur.

Such problems occur because traffic demand changes over time but network topology cannot be changed as rapidly, causing the network architecture to become sub-optimal over time.

Traffic engineering is difficult to do with IGP in large networks because:

1. Among the *equal-cost multi-paths* (ECMPs) [81] from a source, every path will have an equal share of load. This equal ratio cannot be changed. Therefore, one of the

paths may end up carrying significantly more traffic than other paths because it also carries traffic from other sources.

2. Load sharing cannot be done among multiple paths of different cost.
3. Modifying IGP metric to trigger some traffic shift tends to have side effects, and undesirable traffic shifts may also be triggered.

In order to do traffic engineering effectively, the *Internet engineering task force* (IETF) introduces MPLS, constraint-based routing and enhanced link state IGPs. They are briefly reviewed in this section.

4.1.1 MPLS

MPLS is an advanced forwarding scheme. It extends routing with respect to packet forwarding and path controlling [11].

Each MPLS packet has a header. In a non-ATM environment, the header contains a 20-bit label, a 3-bit *experimental* field (formerly known as *class of service*, or CoS, field), a 1-bit label stack indicator and an 8-bit TTL field. In an ATM environment, the header contains only a label encoded in the VCI/VPI field. An MPLS capable router, termed *label switching router* (LSR), examines the label and possibly the experimental field in forwarding the packet.

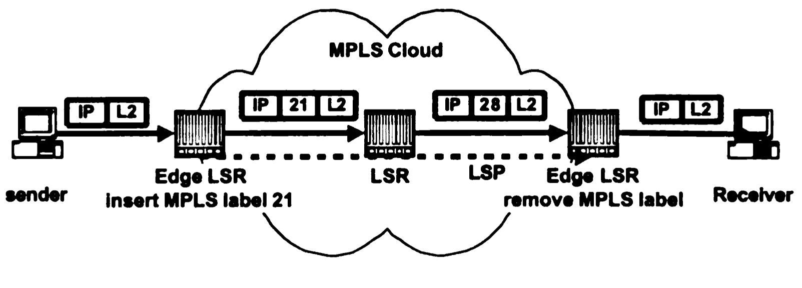


Figure 4-1 MPLS

At the ingress LSRs of an MPLS-capable domain IP packets are classified and routed based on a combination of the information carried in the IP header of the packets and the local routing information maintained by the LSRs. An MPLS header is then inserted for each packet. Within an MPLS-capable domain, each LSR will use the label as the index to look up the forwarding table of the LSR. The packet is processed as specified by the forwarding table entry. The incoming label is replaced by the outgoing label and the packet is switched to the next LSR. This label-switching process is very similar to ATM's VCI/VPI processing. Before a packet leaves a MPLS domain, its MPLS header is removed. This whole process is showed in Fig. 4-1. The paths between the ingress LSRs and the egress LSRs are called *label switched paths* (LSPs). MPLS uses some signaling protocol like RSVP [65][66], LDP [66] or CR-LDP [68] to set up LSPs.

Table 4-1 LSP attributes

Attribute Name	**Meaning of the attribute**
Bandwidth	The minimum requirement on the reservable bandwidth of a path for the LSP to be set up along that path
Path attribute	An attribute that decides whether the path of the LSP should be manually specified or dynamically computed by constraint-based routing
Setup Priority	The attribute that decides which LSP will get the resource when multiple LSPs compete for it
Holding Priority	The attribute that decides whether an established LSP should be preempted the resource it holds by a new LSP
Affinity (color)	An administratively specified property of an LSP (see Section 4.2.1 for detail)
Adaptability	Whether to switch the LSP to a more optimal path when one becomes available
Resilience	The attribute that decides whether to reroute the LSP when the current path is affected by failure

In order to control the path of LSPs effectively, each LSP can be assigned one or more attributes. These attributes will be considered in computing the path for the LSP [63]. The attributes are summarized in Table 4.1.

4.1.2 Constraint-based routing

Constraint-based routing computes routes that are subject to constraints such as bandwidth and administrative policy. Because constraint-based routing considers more than network topology in computing routes, it may find a longer but lightly loaded path better than the heavily loaded shortest path. Network traffic is hence distributed more evenly [83].

For example in Fig. 4-2, the shortest path between router A and router C is through link A-C with IGP metric *m*=1. But because the reservable bandwidth on the shortest path is only (622-600)=22 Mbps (see the figure), when constraint-based routing tries to find a path for an LSP of 40 Mbps, it will select path A-B-C instead, because the shortest path does not meet the bandwidth constraint.

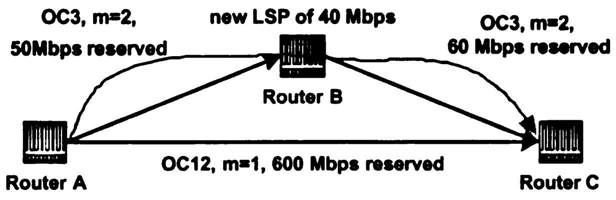


Figure 4-2 Constraint-based routing

It should be noted that the reservable bandwidth of a link is equal to the maximum reservable bandwidth set by network administrators minus the total bandwidth reserved by LSPs traversing the link. It does not depend on the actual amount of available bandwidth on that link. For example, if the maximum reservable bandwidth of a link is

155 Mbps, and the total bandwidth reserved by LSPs is 100 Mbps, then the reservable bandwidth of the link is 55 Mbps, regardless of whether the link is actually carrying 100 Mbps of traffic or more or less. In other words, constraint-based routing does not compute LSP paths based on instantaneous residual bandwidth of links. This reduces the probability of routing instability [5][83].

Constraint-based routing can be online or offline. With online constraint-based routing, routers may compute paths for LSPs at any time. With offline constraint-based routing, an offline server computes paths for LSPs periodically (hourly/daily). LSPs are then configured to take the computed paths.

4.1.3 Enhanced link state IGPs

In order for constraint-based routing to compute LSP paths subject to constraints, an enhanced link state IGP must be used to propagate link attributes in addition to normal link state information [69][71]. Common link attributes include:

1. reservable bandwidth, and
2. link affinity (color), that is, an administratively specified property of the link.

Enhanced link state IGPs will flood information more frequently than normal IGP. This is because, even without topology changes and hence no normal IGP flooding, changes in reservable bandwidth or link color can trigger the enhanced IGP to flood information. Therefore, a tradeoff must be made between the need for accurate information and the need for avoiding excessive flooding. Change in reservable bandwidth is flooded only when it is significant. And a timer should be used to set an upper bound on flooding frequency.

When the enhanced IGP builds LSR's forwarding table, it will take into account LSPs originated by the LSR, so that the LSPs can actually be used to carry traffic. One of such approaches is described in [72]. In a network with MPLS, constraint-based routing enhanced IGP and a path signaling protocol, the process of forwarding table construction is showed in Fig. 4-3.

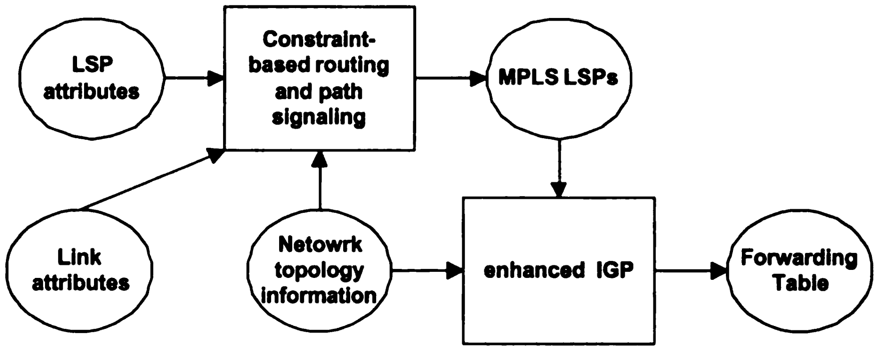

LSP attributes
Constraint-based routing and path signaling
MPLS LSPs
Link attributes
Netowrk topology information
enhanced IGP
Forwarding Table

Figure 4-3 Forwarding table construction

4.1.4 Summary

With MPLS, constraint-based routing and an enhanced link state IGP, traffic engineering can be done much more effectively. The two problems discussed in the introduction can be solved.

First, by setting the maximum reservable bandwidth of each link (e.g., to the physical bandwidth of the link), and by setting the bandwidth requirement for each LSP, constraint-based routing will automatically avoid placing too many LSPs on any link. This solves the first problem. For example in Fig. 4-4, constraint-based routing will automatically place LSP B→E on a longer path to avoid congestion in link C→E. Every link in the figure is an OC3 link with 155 Mbps of bandwidth and has IGP metric 2.

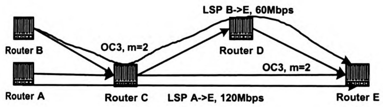


Figure 4-4 Congestion avoidance

Second, if traffic from router C1 to router B1 exceeds the capacity of any single path from C1 to B1 (Fig. 4-5), then multiple LSPs can be configured from C1 to B1, and load splitting ratio of these two LSPs can be specified as desired, so that load can be distributed optimally. This solves the second problem. For example in Fig. 4-5, if the total traffic from router C1 to router B1 is 160 Mbps, and the traffic from C2 to B1 is 90 Mbps, then two LSPs can be configured from C1 to B1. Their load ratio is automatically derived from their bandwidth specification. Here we clearly see that with MPLS, constraint-based routing, and enhanced IGP, not only can load sharing be done among multiple paths of different cost, but also load splitting ratio can be specified as desired. With IGP shortest path routing and equal-cost multi-paths (ECMPs), the problem described in the example cannot be solved.

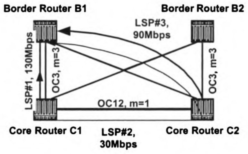


Figure 4-5 Load sharing

In addition to the advantages mentioned above, MPLS also provides the followings:

1. Explicit routes (ERs) can be specified for LSPs. Network administrators can use ERs to control traffic trajectory precisely.
2. Per-LSP statistics can provide accurate end-to-end traffic matrix.
3. Backup LSPs can be used to provide graceful degradation in the case of router or link failure.

In particular, end-to-end traffic matrix makes network planning possible in an IP network without a dedicated connection-oriented network layer such as ATM or frame relay.

4.2 Design of A National MPLS System

In this section, the generic design issues of an MPLS system are discussed, and the design of GlobalCenter's MPLS system is presented.

4.2.1 Generic issues of designing an MPLS system for traffic engineering

To build an MPLS system for traffic engineering, the following design parameters must be determined:

1. the geographical scope of the MPLS system;
2. the participating routers;
3. the hierarchy of MPLS system;
4. the bandwidth constraint of the LSPs;
5. the path attribute of the LSPs;

6. the priority of the LSPs;
7. the number of parallel LSPs between each endpoint pair;
8. the affinity of the LSPs and the links;
9. the adaptability and resilience attributes of the LSPs.

The process of deciding the scope of an MPLS system is driven by administrative policy. Specifically, if the network architecture of a region is irregular, or the capacity of a region is tight, then the region should be included in the MPLS system.

The second step is to decide the participating routers in the MPLS system, i.e., the ingress LSRs, the transit LSRs and the egress LSRs. This should also be guided by the administrative policy. Network administrators may want to forbid some routers from participating in the MPLS system for some reasons such as those routers cannot be trusted or those routers do not have enough processing power and/or memory capacity. Another factor for consideration is the tradeoff between the number of LSPs and efficiency of the links. More ingress and egress LSRs mean more LSPs and thus higher LSP-routing complexity. But because the average size (bandwidth requirement) of the LSPs is smaller, constraint-based routing has more flexibility in routing the LSPs. Higher link efficiency can be achieved.

After the LSRs are decided, network administrators need to decide the hierarchy of the MPLS system. One alternative is to fully mesh all LSRs, resulting in a single layer of LSPs. For large ISPs, there can be hundreds of LSRs. A full mesh will result in a huge MPLS system. Another alternative is to divide one's network into multiple regions. LSRs in each region are meshed. This forms the first layer of LSPs. Some selected LSRs from each region, for example the core routers, are also fully meshed to form the second layer of the LSPs. This hierarchical design can significantly reduce the number of LSPs in the network, and thus the associated processing and managing overhead.

Unless an end-to-end traffic matrix is available beforehand, the bandwidth requirement of the LSPs is usually unknown and has to be guessed when the LSPs are deployed for the first time. Later, the measured rate of the LSPs can be used as the bandwidth requirement of the LSPs. This process is detailed in the next section.

LSP paths can be manually specified or dynamically computed. Unless offline constraint-based routing is used to compute the paths, manually specifying paths for LSPs is difficult. Therefore, LSPs are usually dynamically computed by an online constraint-based routing algorithm in the routers.

Important LSPs, such as those carrying large amount of traffic, can be given a higher priority than other LSPs. In this way, these LSPs are more likely to take the optimal paths. This will result in higher routing stability and better resource utilization network-wide.

Multiple parallel LSPs can be configured between an ingress-egress pair. These LSPs can be placed on different physical paths, so that the traffic load from the source to the destination can be distributed more evenly. By using multiple parallel LSPs, the size of each LSP is also smaller. These LSPs can be routed more flexibly. These are the primary motivations for parallel LSPs. It is recommended that parallel LSPs be used to keep the size of each LSP below 40 Mbps.

Affinity, or color [63], can be assigned to LSPs and links to achieve some desired LSP placement. For example, if network administrators want to prevent a regional LSP from traversing routers or links outside the region, color can be used to achieve the goal.

All regional links can be colored green, and all inter-region links can be colored red. Regional LSPs are constrained to take only green links. In this way, regional LSPs can never traverse any inter-region link. The process of assigning color to LSPs and links is again guided by administrative policy.

Depending on the stability of the network, when better paths become available, network administrators may or may not want to switch LSPs to the more optimal paths. The switching of LSPs to better paths is called LSP *reoptimization* [63]. Reoptimization is not always desirable because it may introduce routing instability. In the case that reoptimization is allowed, it should not occur too frequently. Performing reoptimization once per hour may be a good choice. As to the resilience attribute, LSPs are generally allowed to be rerouted when failure occurs along their paths. In the cases of failure, it may even be desirable to reroute LSPs regardless of their bandwidth and other constraints.

4.2.2 The MPLS system in GlobalCenter's US national network

GlobalCenter is one of the 10 largest ISPs in the US. GlobalCenter's US national network consists of approximately 50 POPs of more than 300 routers. An enhanced IS-IS [70] is used as the IGP. All routers are in the same IS-IS level. All routers also run BGP.

In GlobalCenter, it is decided that MPLS will be deployed nation-wide, for both traffic engineering and *virtual private networks* (VPNs) [74]. Approximately 200 routers with interfaces ranging from DS3 through OC-48c participate in the MPLS system. Fully meshing these 200 routers would result in an MPLS system of about 40,000 LSPs. Although it is not a problem for the network to support this many LSPs, it is decided that better performance and manageability can be achieved by deploying a hierarchical MPLS system of 2 layers of LSPs.

The nation-wide network is divided into 9 regions. All routers in each region are meshed to form the first layer of LSPs. Core routers in all regions are also fully meshed to form the second layer of LSPs. The number of LSRs in a region ranges from 10 to 50. Therefore, the number of LSPs in a region ranges from 100 to 2500. Intra-region traffic, ranging from several Mbps to several Gbps, is transported by these regional LSPs. The core LSP mesh consists of approximately 2500 LSPs. Inter-region traffic, in the magnitude of several Gbps, traverses first the regional LSPs in the source region, then the core LSPs, and last the regional LSPs in the destination region.

It is envisioned that LSPs for VPN will be built between ARs, BRs and HRs across the network in the near future. Such LSPs will be built on the top of the current LSPs that are used for traffic engineering. In other words, a current LSP will be treated as a link in building the LSPs for VPN. Only the endpoints of current LSPs will be involved in the signaling process of building new LSPs for VPN. LSPs are therefore stacked. Such a scheme can significantly reduce state information in the core.

4.3 Performance Evaluation of the Traffic Engineering System

In this section we evaluate performance of the traffic engineering system by comparing the number of congested links and packet loss in the network with and without the traffic engineering system.

With the traffic engineering system, GlobalCenter is able to avoid congestion and relieve congestion immediately after link/router failure. In order to compare the performance of the network with and without traffic engineering, simulation is used. The simulator used is the BBDsgn package of the Wide-Area Network Design Laboratory (WANDL) [75]. This simulator is capable of simulating a traffic engineering system with MPLS, constraint-based routing and an enhanced link state IGP. The simulator and the simulation methodology are briefly introduced below.

4.3.1 Introduction to the WANDL simulator

The WANDL simulator reads four input files:

1. Network node file: this file describes all the nodes (routers/switches) in the network. The fields contained in this file are the names and the geographic locations of all nodes.
2. Network link file: this file describes all the network interfaces in the network. Because all links are full duplex, each link actually consists of two interfaces. These two interfaces can have different IGP metrics. The important fields contained in this file are: interface name, source node, destination node, IGP metric, a Boolean variable for denoting whether the interface is MPLS capable or not, and if the interface is MPLS capable, the maximum reservable bandwidth of the interface.
3. Traffic demand file: this file describes the end-to-end traffic demand matrix of the network. The fields in this file are: source node, destination node and the amount of traffic from the source that sinks at or exits from the destination node.
4. LSP file: this file describes all the LSPs in the MPLS system. The important fields are the name of each LSP, its bandwidth requirement, setup and holding priorities, and the complete path of the LSP. This file is not needed for simulating networks without LSPs.

Given these inputs, the WANDL simulator will simulate traffic mapping in the network, visualize condition of the network, and report all the congested links (if any) whose traffic demand is higher than its physical capacity and the corresponding packet loss.

4.3.2 Simulation methodology and result

In this section, we describe how information in the input files is extracted.

The node file and link file are extracted from the link state database (LSDB) of a router in the network. Because all routers in the network are in the same IS-IS level, their LSDBs are identical. Therefore, extracting the node and link information from any one of these routers is sufficient to provide the node and link information.

The traffic demand file is constructed from the traffic statistics of the LSPs. Basically, the statistics of the full mesh of core LSPs provide the inter-region traffic demand, and the statistics of the regional LSPs provide the intra-regional traffic demand. Specifically, the 95th-percentile rate of an LSP from the source node to the destination node, measured every 5 minutes for the past 7 days, is considered as the traffic demand from the source to the destination.

The traffic demand file described above is different from a complete end-to-end traffic demand file, but it is sufficient to describe the actual traffic demand in the network.

To simulate the network with MPLS and constraint-based routing, the LSP file is supplied by listing all the MPLS LSPs in the network. To simulate the network without MPLS and constraint-based routing, the tunnel file is simply not supplied. The simulator will then map all traffic from a particular source to a particular destination to the shortest path(s) between the source and the destination. To simulate failure of a link, the line that corresponds to the link in the link file is simply removed.

The simulation result is summarized in Table 4.2.

Table 4-2 Differences in network condition with and without traffic engineering

Network Scenario		Congested Links and Data Loss	
		With MPLS	Without MPLS
1	No Link Failure	None	CR2.POP7→CR2.POP8, OC12: 26.98Mbps CR2.POP8→WR2.POP5, OC12: 6.93Mbps WR2.POP10→CR1.POP1, OC3: 30.58Mbps CR1.POP1→BR1.POP1, OC3: 28.11Mbps
2	cr1.POP8-wr1.POP5 (OC48) fails	None	CR2.POP7→CR2.POP8, OC12: 417.75Mbps CR2.POP8→WR2.POP5, OC12: 629.2Mbps WR2.POP10→CR1.POP1, OC3: 30.58Mbps CR1.POP1→BR1.POP1, OC3: 28.11Mbps
3	Cr1.POP4-cr2.POP8 (OC12) fails	None	CR1.POP8→CR1.POP4, OC3: 336.78Mbps CR2.POP8→WR2.POP5, OC12: 1.77Mbps WR2.POP10→CR1.POP1, OC3: 30.58Mbps CR1.POP1→BR1.POP1, OC3: 28.11Mbps
4	Traffic growth of 200Mbps from cr1.POP7 to cr1.POP9	None	CR1.POP7→CR1.POP8, OC12: 22.82Mbps WR1.POP10→WR1.POP2,OC12: 175Mbps WR2. POP10→CR1.POP1, OC3: 30.58Mbps CR1.POP1→BR1.POP1, OC3: 28.11Mbps
5	Cr1.POP8-wr1.POP5 (OC48) Cr2.POP8-wr2.POP5 (OC12)	CR2.POP8 →CR1.POP4, OC12: 44.43Mbps	CR2.POP8→CR1.POP4, OC12: 98.16Mbps CR1.POP7→CR3.POP3, OC12: 738.25Mbps CR3.POP3→ CR2.POP3, OC12: 700.18Mbps CR2.POP3→CR1.POP6, OC12: 750.61Mbps CR1.POP6→WR1.POP5, OC12: 432.88Mbps WR2.POP10→CR1.POP1, OC3: 30.58Mbps CR1.POP1→BR1.POP1, OC3: 28.11Mbps

The tunnels in the network are showed in Fig. 4-6.

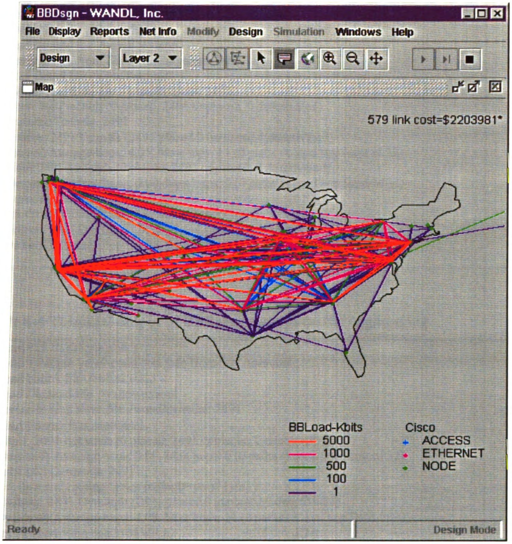


Figure 4-6 Tunnels (LSPs) in the network

The simulation results of the network with and without traffic engineering in normal condition (no failure) are showed below in Tables 4.3 and 4.4. Pictures of link utilization with and without traffic engineering for the part of the network in the California Bay area are showed in Fig. 4-7 and Fig. 4-8. Please note the difference in link utilization.

Table 4-3 Terminal output of BBDsgn, with traffic engineering

```
--- Read 315 Backbone nodes
--- Read 1 default vendor and cost info from file ./usercost
--- Read bblink file ./bblink.0213
--- Read Demand file ./wandldemand
--- Demands read from file ./wandldemand: 2491
--- Read Tunnel file ./wandltunnels
--- Tunnels read from file ./wandltunnels: 2491
--- Initial: 2491 tunnels (0 placed, 2491 unplaced, 0 deactivated)
--- Unplaced Tunnels: 2491
--- Iteration: 2491 Tunnels (2491 placed,0 unplaced,0 deactivated)
--- (Tunnel) Average hops: 4.05, Max hops: 11 (0 paths exceed hop limit of 32)
--- Initial: 2491 demands (0 placed, 2491 unplaced, 0 deactivated)
--- (Demand) Average hops: 0.00, Max hops: 0 (0 paths exceed hop limit of 32)
--- Unplaced Demands: 2491
Update demand routing tables (default=yes)? (y/n) y
--- Iteration: 2491 Demands (2491 placed,0 unplaced,0 deactivated)
--- (Demand) Average hops: 4.05, Max hops: 11 (0 paths exceed hop limit of 32)
```

Table 4-4 Terminal output of BBDsgn, without traffic engineering

```
--- Read 315 Backbone nodes
--- Read 1 default vendor and cost info from file ./usercost
--- Read bblink file ./bblink.0213
--- Read Demand file ./wandldemand
--- Demands read from file ./wandldemand: 2491
--- Read Tunnel file ./notunnel
--- Initial: 2491 demands (0 placed, 2491 unplaced, 0 deactivated)
--- (Demand) Average hops: 0.00, Max hops: 0 (0 paths exceed hop limit of 32)
--- Unplaced Demands: 2491
Update demand routing tables (default=yes)? (y/n) y
--- Iteration: 2491 Demands (2491 placed,0 unplaced,0 deactivated)
--- (Demand) Average hops: 4.09, Max hops: 11 (0 paths exceed hop limit of 32)
# Over Subscription occurred at:                      A2Z        Z2A
#     CR2.POP7-CR2.POP8:    DEF   OC12 AvailBw=  -26.98M   471.384M
#     CR2. POP8-WR2.POP5:   DEF   OC12 AvailBw=   -6.93M   449.206M
#     WR2.POP10-CR1.POP1:   DEF   OC3 AvailBw=   -30.58M    71.211M
#     BR1.POP1-CR1.POP1:    DEF   OC3 AvailBw=  155.000M   -28.11M
```

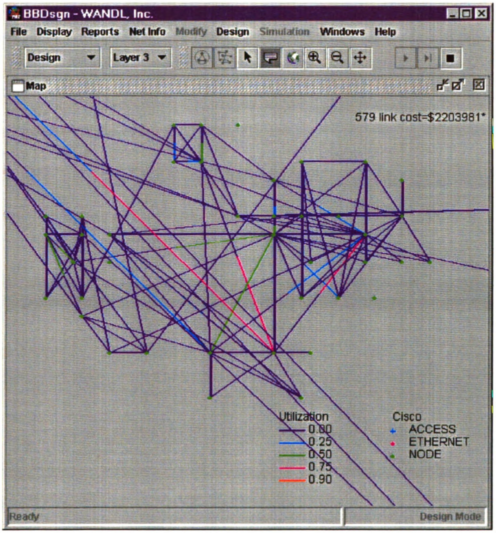
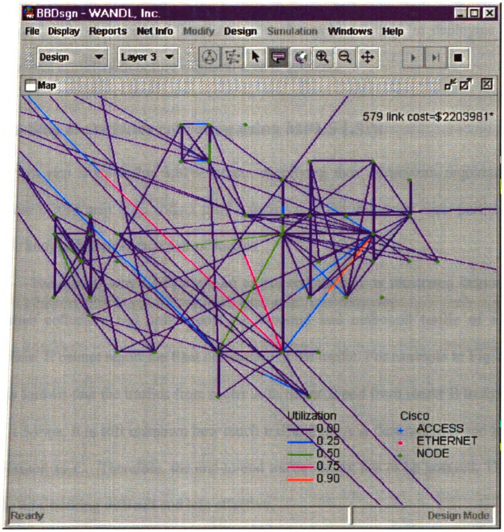


Figure 4-7 Network condition with traffic engineering



579 link cost=$2203981*

Figure 4-8 Network condition without traffic engineering

Deploying an MPLS System for Traffic Engineering

In this section, a procedure for deploying MPLS systems for traffic engineering is proposed. It is recommended that a non-critical area be selected for deployment to gain operational experiences before a critical area is selected.

4.3.3 Step 1: statistics collecting using MPLS LSPs

The first step is to deploy LSPs without specifying their bandwidth requirement. The purpose of this step is to collect traffic statistics in the interested area, particularly the traffic amount between each pair of routers.

The reason why LSPs are used to collect statistics is illustrated below. Current statistics collectors can only report the inbound and outbound traffic of a network interface. It cannot report the final destination of the traffic. For example in Fig. 4-9, even if it is known that the traffics from router A to router B and from router B to router C are both 1 Mbps, it is still unknown how much traffic from A is destined to B and how much is destined to C. Therefore, the end-to-end traffic matrix has to be guessed. This makes traffic engineering difficult and less effective.

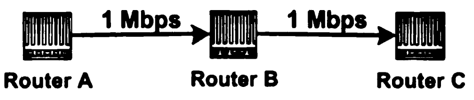


Figure 4-9 Statistics collecting

However, if two LSPs are configured from A to B and from A to C, then the traffic from A to C will not enter the LSP from A to B. Therefore, the statistics of these two LSPs will tell network administrators precisely how much traffic from A is destined to B and how much is destined to C.

An end-to-end traffic matrix is a two-dimensional array. Each row of the array is indexed by a router in the network. Each column is indexed either by a router in the network, or by an IP address prefix. Each element of the array is the amount of traffic from the router in that row to the router or the address prefix in that column. Most traffic engineering mechanisms are effective only when an end-to-end traffic matrix is available. The first form of traffic matrix is useful for both intra-domain and inter-domain traffic engineering. The second form of traffic matrix is useful mainly for inter-domain traffic engineering.

4.3.4 Step 2: deploy LSPs with bandwidth constraint

The second step is to deploy LSPs with their bandwidth requirement specified. Usually, the measured rate of an LSP is used as the bandwidth requirement of that LSP. Because the measured rate of an LSP can vary significantly at different time, it must be decided which value to use. One common choice is the 95th-percentile of all rates measured every 5 minutes over a period of time, e.g., the past 7 days. This value is usually close to the real peak value (as opposed to a traffic spike).

Online constraint-based routing will compute paths for the LSPs such that for every link, the maximum reservable bandwidth of the link is greater than or equal to the sum of the specified bandwidth of all LSPs traversing the link. Under this premise, whether high utilization will occur in a link depends on how close the total actual traffic rate of all the LSPs matches the total specified bandwidth. If the total rate is greater than or equal to the total specified bandwidth, high utilization may occur in the link.

There are two approaches to avoid the above problem. The first approach is to under-subscribe the links. For example, the maximum reservable bandwidth of an OC-3c link that tends to be highly utilized can be set to 60% of the link capacity, i.e. 100Mbps. This approach is useful for controlling the utilization of a specific link. The second approach is to inflate the bandwidth requirement of the LSPs by a factor. For example, if the desired utilization of all links is 60%, then a factor of 1/0.6=1.67 can be used. More specifically, the required bandwidth of every LSP is set to the 95th-percentile of the measured rate multiplied by the factor. The philosophy of this approach is to reserve some headroom for each LSP to grow. It is useful for controlling the utilization of all links.

With either approach, a tradeoff must be made between the need to avoid congestion and the need for link efficiency. Under-subscription or a large inflation factor can both cause some LSPs to take sub-optimal paths, even though the optimal paths have enough physical bandwidth (but not enough reservable bandwidth) for these LSPs. This reduces efficiency of the links.

No matter what has been done, it is still possible that the utilization of a specific link become too high. In that case, network administrators can either manually move some LSPs away from the congested link by using explicit routes, or lower the reservable bandwidth of the congested link to force some LSPs to reroute, or add parallel LSPs along some lightly loaded alternative paths to offload some traffic from the congested link. Load splitting ratio of the parallel LSPs is determined by their specified bandwidth. In order do these effectively, tools for showing the paths of the LSPs, the reservable bandwidth of the links, and for suggesting the alternative paths, are very useful.

A simulation tool like the BBDsgn of WANDL can be used before the actual deployment to make sure that the constraint-based routing algorithm can find paths for all the LSPs. Otherwise, more capacity has to be added to the network. If the simulator shows that some links will have very high utilization, the congestion control approaches described above can be used to solve the problem.

4.3.5 Step 3: periodic update of LSP bandwidth

The third step is to further collect statistics and adjust the bandwidth of the LSPs if necessary. This is needed because network traffic is growing and traffic distribution is changing all the time. This step should be done periodically, e.g. daily or weekly, by some scripts. It is wise not to adjust the bandwidth requirement of all LSPs at the same time. Otherwise, significant change may be observed during the bandwidth adjustment process.

4.3.6 Step 4: offline constraint-based routing

If optimal link efficiency is desired, offline constraint-based routing can be used to compute the paths for the LSPs. By taking all the LSP requirements, link attributes and network topology information into consideration, an offline constraint-based routing server may be able to find better LSP placement than online constraint-based routing, where every router in the network finds paths for its LSPs separately based on its own information. Therefore, links can be used more efficiently. Offline constraint-based routing will compute paths for LSPs periodically, e.g. daily. Then the path information of the LSPs is downloaded into the routers. Online constraint-based routing is still used in the routers, so that if some routers or links fail, new paths will be computed for those affected LSPs immediately. A simple offline constraint-based routing algorithm is described below.

Compute paths for LSPs one by one, starting from high priority LSPs. For LSPs with the same priority, compute paths for them in the order of decreasing bandwidth requirement. The goal of this algorithm is to minimize the sum of the (*reserved bandwidth* $\times$ *path metric*) product for all LSPs. That is, the goal is to let the largest LSP take the best path, the second largest LSP take the next best path, and so on, inside each priority class. The algorithm is briefly described below.

1) Sort the LSPs in decreasing order of importance as described above;
2) For a particular LSP, first prune all the unusable links;

   A link can be unusable for an LSP because:

   - the reservable bandwidth of the link is not sufficient for the LSP or the delay of the link is too high (e.g., satellite links);
   - the link is administratively forbidden for the LSP, e.g., *red* links cannot be used for a *green* LSP.
3) On the remaining graph, compute the optimal path for the LSP;
4) For those links used by the LSP, deduct the resources (e.g., link bandwidth) used by the LSP;
5) Repeat steps 2-4 for the next LSP until all are done.

If backup LSPs need to be computed, repeat the above procedure on the remaining graph, that is, with resources used by the primary LSPs deducted. The only difference is in step 2. Now all links and routers used by the corresponding primary LSP of this backup LSP are also pruned. This is to avoid any single point of failure for both the primary and the backup LSPs.

This algorithm may not find the globally optimal layout for LSPs. But it is simple. The problem of optimizing the (*reserved bandwidth* $\times$ *path metric*) product for all LSPs is NP-complete [76], because the BIN-PACKING [76] problem can be reduced to it. Therefore, finding the optimal solution is not practical except for small networks.

4.4 Network Planning

Although network optimization with MPLS, constraint-based routing and an enhanced link state IGP is very useful in avoiding and/or relieving congestion, they should never be considered as a substitute for network planning.

Network planning is to initiate actions to evolve the architecture, topology, and capacity of a network in a systematic way. When there is a problem in the network, network optimization must provide an immediate fix. Because of the need to respond promptly, the fix may not be the optimal solution. Network planning may subsequently be needed to improve the situation. Network planning is also needed to expand the network to support traffic growth and changes in traffic distribution over time. Through network planning, network administrators may decide to add more routers/links to eliminate single points of failure, add capacity to the network to remove bottlenecks and to accommodate traffic growth, remove unused routers/links to reduce cost, and change IGP metrics systematically to make the network robust, adaptive and easy to operate.

A network simulator is very useful for network planning. First, during the process of network planning, a simulator may reveal pathologies such as single points of failure and bottlenecks. Second, a simulator can be used for failure analysis. Appropriate actions can then be taken to make the network more robust. Third, a network simulator can be used to investigate and identify how to evolve and grow the network in order to accommodate projected future demands. Last, a simulator can be used to validate the effectiveness of planned solutions without the need to tamper with the operational network. As a result, they may help to avoid commencing an expensive network upgrade that may not achieve the desired objectives.

With bandwidth becoming cheaper and more abundant, providing a certain amount of extra capacity becomes more practical. With some extra capacity, the need for complex network optimization mechanisms that are designed for achieving maximum link efficiency is reduced. This helps to keep a traffic engineering system simple. Extra capacity also makes it easy to provide QoS by over-allocating bandwidth to those flows with QoS requirement such as delay bound. The cost of extra bandwidth can be well compensated from the reduction of network operation cost. This has to be carefully considered in network planning.

Network planning and network optimization are mutually complementary. A well-planned network makes network optimization easier, while the process of network optimization provides statistics and insights for network planning, and allows network planning to focus on long term issues rather than tactical considerations.

4.5 Multi-protocol Lambda Switching

Currently, routers in different POPs are connected by links in a ring fashion. Traffic has to be routed/switched at each intermediate node. With optical switching, it is relatively easy to establish a link mesh (not necessarily a full mesh) among the nodes.

Optical switching is very useful for traffic engineering for two reasons. First, capacity between two nodes can be easily added by turning up a link between these two nodes in a very short time, e.g., minutes. This can dramatically change the scenario of network planning. Second, because two nodes that are not adjacent at physical layer can become adjacent at the link layer, routing/switching can be done between these two nodes directly. This makes traffic control much simpler.

In order to switch lambda without converting them into digital signal, optical cross-connects (OXCs) must exchange information so that the mapping from the input channels to the output channels can be established. This process is very similar to the signaling process that LSRs use to establish the mapping from the incoming MPLS labels to the outgoing labels. Besides, because a fiber can only carry a certain number of lambda, network administrators cannot put too many lambda into the same fiber. This is very similar to the scenario where network administrators cannot put too many LSPs into the same link because each link has a certain capacity. In fact, the lambda switching process is very similar to the label processing process. Specifically, a lambda in lambda switching is like a labeled packet in label switching, and an optical channel trail (OCT, a concatenation of multiple channels from the source to the destination) is like an LSP. Therefore, instead of using two separate control planes -- one for lambda switching and one for label switching -- it is more appropriate to extend the control plane of label switching and use it for lambda switching [16]. Specifically, the OXCs will use an enhanced IS-IS or OSPF to distribute attribute of the fibers. For an OXC that needs to set up an OCT, the OXC will use constraint-based routing to compute the path for the OCT based on the constraint of the OCT and the attributes of the fibers. A signaling protocol such as RSVP will then be used to set up the OCT.

4.6 Inter-domain traffic engineering

Sections 1-6 focus on the intra-domain aspect of traffic engineering. This section focuses on the inter-domain aspect.

4.6.1 Introduction to inter-domain traffic engineering

Inter-domain traffic engineering is to balance egress traffic among multiple peering circuits to other domains. This is the focus of this section. How to balance incoming traffic among multiple peering circuits is not the focus of this section because:

1. Some well known techniques exist for a domain to influence other domains on their selection of traffic egress points. These techniques include:
   1) announcing route to the same IP address prefix with different BGP MULTI_EXIT_DISCs (MEDs) at different peering points;
   2) pre-pending the local *autonomous system* (AS) number different number of times at different peering points;
2. Such influence can be completely ignored by the upstream domain. In other words, balancing the ingress traffic can be beyond the control of the current

domain. The receiving domain should be prepared to handle all possible ingress traffic distribution.

It should be noted that even if there is only one peering circuit between two domains D1 and D2, there can still be inter-domain traffic engineering issues between D1 and D2. The reason is that D1 can use another domain D3 as transit domain for some traffic destined to D2, if

- the peering circuit from D1 to D2 is congested;
- the peering circuit from D1 to D3 is idle; and
- D3 has connectivity to D2.

4.6.2 A simple introduction to BGP

Because inter-domain traffic engineering is closely related to BGP, we give a simple introduction to BGP here.

BGP is an inter-domain routing protocol. It is used to construct the routing table for destinations outside the AS. BGP is a path vector protocol [73].

BGP can be used between two ASs, so that each AS can learn routes to destinations in the other AS. In this case, BGP is called *external BGP* (eBGP). Or, BGP can be used among routers in the same AS to exchange the routes they have learned from other ASs. In this case, BGP is called *interior BGP* (iBGP).

In a domain, each router will announce routes to all directly connected networks, and to networks it learns from other domain via eBGP. Network administrators can also enumerate networks to announce. These are the sources of BGP routes. BGP routers exchange routes between peers. The result is each BGP router will have a routing table. The entries in the table specify routes to all destinations in the Internet.

A BGP process consists of three sub-processes.

1. The input sub-process;

When a BGP router receives routes from its peers, the input process will decide which routes to keep and which to discard. The input process may also modify the attributes of the routes.

2. The decision process;

A destination network may be reached via multiple routes. The decision process decides which route to choose. Only the chosen route will be installed in the routing table.

Each BGP route may have multiple attributes, e.g., local preference, AS path and MED. Local preference and MED are optional attributes while AS path is mandatory attribute. Simply speaking, when there are multiple routes to a destination, the route is selected first by local preference, then by AS path length, and then by MED.

3. The output process.

Of all the routes in the routing table, the output process decides which routes to advertise to its peers.

The most important features of BGP include:

1. supporting *classless inter-domain routing* (CIDR). CIDR helps to prevent IP addresses from depleting;

2. supporting route aggregation, which significantly reduces the routing table size;

3. providing policy and other control mechanisms for network administrators to select routes.

4.6.3 Inter-domain traffic engineering with BGP

Inter-domain traffic engineering is traditionally done by manipulating the BGP route metrics. A typical approach is illustrated below with Case 1.

Case 1:

- Domain D1 peers with domain D2 at border router BR_A (site A) and border router BR_B (site B). Both border routers in D1 receive identical routes to some 35K prefixes in D2; BR_A and BR_B announce these 35K routes to other routers in D1 with identical metric *x*.
- Problem: BR_A has too much egress traffic while BR_B is relatively idle;
- Solution: some egress traffic needs to be moved from BR_A to BR_B.

Today, the solution is provided by dis-preferring routes for some of the 35K prefixes at site A. More specifically, if it is decided that the outgoing traffic at site A should be reduced by 20%, BR_A can announced 35K $\times$ 20% = 7K routes with a higher metric y, $y>x$. Traffic to such prefixes that used to exit at BR_A will then exits at BR_B. Assuming that traffic to every prefix is equal, then 20% of the traffic exiting at BR_A will be moved to BR_B. In reality, this may not be the case. Then, if too little traffic is moved, more routes can be dis-preferred at BR_A; if too much traffic is moved, fewer routes can be dis-preferred. Without knowing the amount of traffic to each prefix, this process is trial-and-error in nature.

4.6.4 Inter-domain traffic engineering with MPLS

With MPLS, a new approach can be used to balance the egress traffic at different sites, which is to set the administrative weight of the LSPs to the multiple exit routers so as to prefer one or more exit routers to others. The solution in Case 1 can be provided as follows.

Assuming that currently router R in domain D1 is sending egress traffic to domain D2 through BR_A, and we want to move R's egress traffic from through BR_A to through BR_B. Since the BGP attributes of all routes are identical announced by BR_A and BR_B, the IGP route from R to BR_A must be shorter than the one from R to BR_B. By establishing two LSPs from R to BR_A and BR_B, and setting the administrative weight of the LSP to BR_A to a value larger than that of the LSP to BR_B, the egress traffic from R will be moved from through BR_A to through BR_B.

The major limitation of the above approach is that it is unknown how much traffic will be moved from through BR_A to through BR_B, because it is unknown how much egress traffic R originates.

Without MPLS, by tuning the IGP metrics of the links to make the IGP route from R to BR_A longer, traffic that used to exit at BR_A can also be moved. But the problem with this approach is that when the IGP metrics of the links are changed, the effect is global to domain D1. Other undesirable traffic shift may be triggered.

4.6.5 An quantitatively approach for inter-domain traffic engineering

In order to perform inter-domain traffic engineering quantitatively, an end-to-end traffic matrix is needed.

We again use Case 1 as an example. We further assume that the amount of egress traffic is 160Mbps at site A and 100 Mbps at site B, and that 30Mbps of egress traffic is to be moved from site A to site B.

An approach based on BGP is described as follows.

First, a traffic matrix in the form of router-prefix (described in Section 4.3.3) is needed. In the traffic matrix, select a number of prefixes in the row of router BR_A such that the total traffic to these prefixes is close to 30Mbps, and dis-preferred the routes to those prefixes at router BR_A. Let's denote these routes as set {R1}. These 30Mbps of traffic will then be moved away from site A. In Case 1, there are only two possible egress points. Therefore the traffic must go to site B.

If there are more than two egress points to domain D2, i.e., sites A, B, C, ..., K, then the 30Mbps of traffic that is moved away from A may go to B, or C, ..., or K. Let's further assume that too much of this 30Mbps of traffic (say 25 Mbps) is moved to B, and we want to move some out of it. First, an updated router-prefix traffic matrix must be obtained. Then a subset of {R1}, denoted as {R2}, can be dis-preferred at router B such that the total amount of traffic carried by {R2} is close to the amount of traffic that needs to be moved away. Because {R2} is a subset of {R1}, and {R1} has already been dis-preferred at site A, the traffic that is moved away from router BR_B cannot go back to router BR_A. Therefore, with each step, desired amount of egress traffic can be achieved for at least one border router. Assuming that there are totally *n* egresses to D2, the above process needs to be repeated at most (n-1) times to achieve the desired egress traffic distribution.

Another approach based on MPLS can also be used.

First, a traffic matrix in the router-router form (described in Section 4.3.3) is needed. Let's further assume that there is no customer networks directly attached to the border routers, which is generally true for ISP networks. Then all traffic from other routers destined to a border router (from D1's perspective) must be egress traffic. Therefore, a number of rows in the traffic matrix can be selected, such that the total amount of traffic from these routers to router BR_A adds up to 30Mbps. We then make the administrative weights of all LSPs from these routers to BR_B smaller than those to BR_A. Because the BGP attributes of all prefixes are identical, and the intra-domain routes to BR_B are shorter than the administrative weights of LSPs to BR_A, all egress traffic of the selected routers will be moved from through router BR_A to through router BR_B.

Even if there are more than two egress routers, the above process is guaranteed to move traffic to BR_B, not any other egress routers, as long as the administrative weights of the LSPs from the selected routers to BR_B are set smaller than the administrative weights of LSPs to all other egress routers.

With the MPLS approach, a single step is sufficient to move traffic away from router BR_A, and the traffic that is moved away can be distributed to other egress routers as desired by manipulating the administrative weights of the LSPs from the selected routers to other egress routers accordingly. However, it requires that the selected routers have LSPs to the egress routers. Therefore, the MPLS approach is appropriate to control egress traffic distribution among multiple egress routers in the same MPLS region. To control the egress traffic distribution among multiple routers in different MPLS region, the BGP approach can be used.

4.7 Fast Reroute

Because the speed of links and routers is becoming higher and higher, when a link or a router fails, large amount of data will be buffered for a long time or even dropped. Therefore, it is very important to temporarily repair the affected paths so that they can continue to carry traffic before more optimal paths are computed and used to carry traffic. This is the task of fast reroute. It is especially important for providing QoS for high priority traffic. In this section, we describe a fast reroute approach for protecting segments of paths.

LSP1 starts from LSR(0) and ends at LSR(*n*). The segment to be protected is from LSR(*i*) to LSR(*k*). Note that LSR(*i*) and LSR(*k*) can be adjacent or non-adjacent, LSR(*i*) can be LSR(0), and LSR(*k*) can be LSR(*n*).

In order to protect the segment between LSR(*i*) and LSR(*k*), a local-repair LSP is created from LSR(*i*) to LSR(*k*). The requirement on the local-repair LSP is that it must be disjoint from the protected segment. The local-repair LSP and LSP1 merge at LSR(*k*).

In normal case, LSR(*i*) will swap the incoming label of a packet in LSP1 with the outgoing label for LSP1 and switch the packet to the next LSR. When a router or a link along the protected segment fails, LSR(*i*) will detect the failure (from link layer information or something else). Instead of swapping the incoming label of LSP1 with the outgoing label for LSP1, LSR(*i*) will swap the incoming label of LSP1 with the outgoing label for the local-repair LSP. Traffic is carried by the local-repair LSP to LSR(*k*). At

LSR(k), the local-repair LSP merges into LSP1. This process is transparent for LSRs between LSR(k+1) and LSR(n).

When a router or a link along the protected segment fails, it takes LSR(i) time to detect that. The longer the protected segment, potentially the longer it takes LSR(i) to detect the failure. In order for fast reroute to take effect within 50ms (comparable to SONET's protection mechanism), LSR(i) and LSR(k) should be just one hop or two hops away from each other.

During fast reroute, the path of an LSP from the source to the destination can be sub-optimal. Therefore, the LSR(i) should also send a signal to notify LSR(0) that the protected segment fails. Upon receiving the signal, the LSR(0) will switch all the traffic to LSR(n) from LSP1 to a backup LSP. The backup LSP can be pre-computed or can be computed upon receiving the signal.

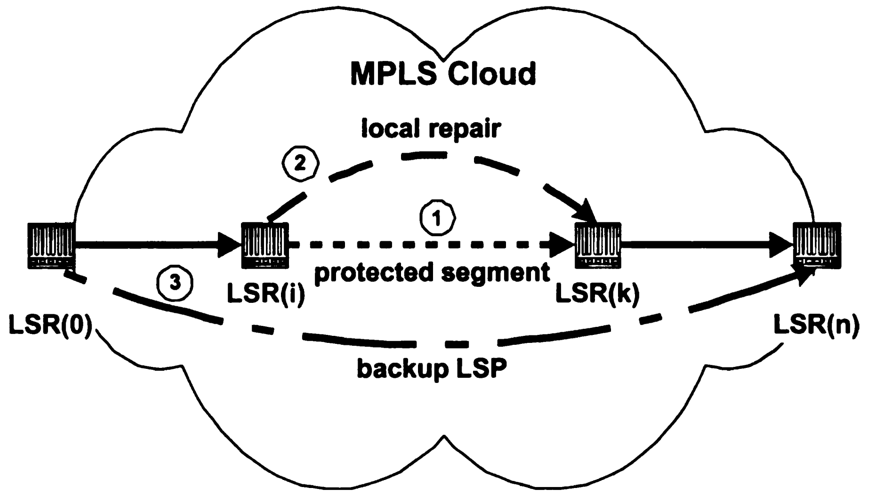


Figure 4-10 Fast reroute

Fast reroute adds significant complexity to the network. Although it is useful for providing QoS to high priority traffic during router or link failure, it may not be worthwhile for best effort traffic.

If fast reroute is used for high-priority traffic, resource should be reserved for the local-repair LSP. Besides, it is advisable to pre-compute the backup LSP and reserve resources for it. But local-repair LSPs that protect different nodes can share resources under the assumption that no different links/nodes will fail at the same time. If backup LSPs or local-repair LSPs are used for best effort traffic, then resources need not to be reserved.

4.8 Providing Diffserv in an MPLS-based Network with Traffic Engineering

In Chapter 3, we described an approach for providing Diffserv in an ISP network. Three services, premium, assured and best effort, are provided. Premium service is for real-time and mission-critical traffic. Assured service is for non-real-time interactive traffic. Best effort service is the traditional Internet service.

In that approach, packets from a customer domain are classified, policed and possibly shaped at the edge of the network. This is done according to the traffic profile specified in the SLA between the customer and the ISP. The DSCPs of the packets are then set accordingly. After the marking, packets are buffered and scheduled based solely on their DSCPs by RIO and WFQ. Because the provisioning factors *pf*, i.e. the ratio of the configured output rate to the actual input traffic rate, of the premium queue (PQ), assured queue (AQ) and the default queue (DQ) are different,

$$pf(\mathrm{PQ}) > pf(\mathrm{AQ}) > pf(\mathrm{DQ}) > 1.0,$$

and packets that are in or out of profile are dropped by RIO with different probability, the service quality of packets in different classes are differentiated.

With MPLS in the network, IP headers are not examined when the packets are MPLS-labeled. Therefore, packets cannot be differentiated based on their DSCPs. Diffserv must be provided in a different way, as described below.

4.8.1 Major differences between an MPLS-based service architecture and an IP-based service architecture

The main differences in the processing of a packet in an MPLS-based service architecture and in an IP-based service architecture that is described in Chapter 3 are summarized below.

1. At the ingress LSR, an MPLS header is inserted into the packet. The DSCP in the IP header is mapped to the *experimental field* (EXP field) in the MPLS header.
2. BA classification is based on the EXP field rather than on the DSCP.

3. At the egress LSR, the MPLS header is removed.

4.8.1.1 Mapping the DSCP to the EXP field

Packets still have their DSCPs set at the edge of the network. In addition, when MPLS headers are inserted at the ingress LSRs, the DSCPs in the IP headers are mapped to the EXP fields in the MPLS headers. Because the DSCP is 6-bit long, and the EXP field is only 3-bit long, some information in the DSCP may be lost in the mapping. But in the framework presented in Chapter 3, only the leftmost 3 bits (the former precedence field) contain useful information, these 3 bits can be copied to the EXP field. Therefore, the leftmost 2 bits denote the service class, and thus the service queue. The 3rd bit denotes whether the packet is in-profile or out-of-profile, and thus the drop probability.

4.8.1.2 BA classification, buffer management and queue scheduling

In the middle of an LSP, BA classification is based on the EXP field rather than the DSCP. Buffer management and queue scheduling are identical with or without MPLS. Therefore, the per-hop behavior (PHB) of the packets is also identical. This is desirable because it means that:

- ISPs that provide QoS with MPLS can easily inter-operate with ISPs that provide QoS without MPLS;
- Whether MPLS is involved or not in providing QoS is transparent to end-users.

4.8.2 Additional features of an MPLS-based service architecture

Besides the three main differences stated at the beginning of the section, an MPLS-based service architecture can provide some additional features to make QoS provisioning easier, compared to an IP-based service architecture.

4.8.2.1 Class-based routing with LSPs

Sometimes, it is desirable to let different classes of traffic take different paths. In order to do so, multiple LSPs can be created from the same source to the same destination, one LSP per class. LSPs for different classes can have different constraints. Each physical link can also be divided into multiple virtual links for different classes of traffic. Therefore, the physical network is divided into multiple virtual networks, one per class. These virtual networks may have different topology and resources if desired.

For example, if different LSPs are used for different classes of traffic from the same source to the same destination, the premium LSP can be given high priority, the assured LSP can be given middle priority, and the best effort LSP can be given low priority. Then premium LSPs can take the shortest paths. This can be useful because with high link speed, propagation delay has become a significant part of the total delay; by taking the shortest paths, the propagation delay of premium traffic can be reduced as much as possible. In the case of link or router failure, premium traffic will also have high priority in getting the backup resources than other classes of traffic. This process can be automatic, driven by constraint-based routing.

Although class-based routing can also be done in an IP network without MPLS, a source router cannot control the paths of its traffic to each destination, unless the complete path is carried in the header of every packet.

4.8.2.2 Resource allocation

Using an MPLS-based architecture to providing Diffserv can make resource allocation in ISP domain significantly easier. First, by setting the maximum reservable bandwidth for the high priority traffic at a desired level, constraint-based routing can be used to prevent the concentration of high priority traffic. This solves the problem that an IP-based service architecture for providing Diffserv cannot solve. Second, the rate of all the LSPs traversing through an interface provide the needed statistics for setting the rates of the queues. From the queue rates and the desirable queueing delay, the size of the queues and the RED/RIO parameters can then be derived.

4.8.3 Scalability of the MPLS-based service architecture

In this approach, as the number of transit flows increases, the number of flows in each LSP increases, but the number of LSPs need not increase. Therefore, this approach is scalable to large ISP networks.

4.9 Conclusion

In this chapter, we first briefly review MPLS, constraint-based routing and enhanced link state IGPs to provide a background for traffic engineering. We then discuss the general issues of designing an MPLS system for traffic engineering. The design of an MPLS system is presented and its performance is evaluated. Based on the experiences, a generic procedure for deploying MPLS systems is proposed. We then discuss the importance of network planning and MPLmS. The inter-domain issues of traffic engineering are discussed, and an approach for systematic inter-domain traffic engineering is described.

Besides traffic engineering, an approach for providing fast reroute for high priority traffic is briefly discussed. And an MPLS-based service architecture for providing Diffserv is described.

Deploying an MPLS system for traffic engineering in a large ISP network is feasible. MPLS, constraint-based routing and an enhanced IGP have to be deployed in every router that participates in the MPLS system. Because everything is done by software upgrade, such a wide-range deployment turns out not to be as difficult as many people might think.

A hierarchical MPLS system consisting of a core LSP mesh and several regional LSP meshes is a good design. By first introducing LSPs to collect statistics, then deploying LSPs with bandwidth requirement, and updating the bandwidth requirement of the LSPs periodically, an MPLS system can be deployed in an evolutionary fashion in an ISP network.

An MPLS system is very useful for traffic engineering. It automatically achieves desired traffic behavior and significantly relieves network administrators from the tedious work of manipulating routing metrics. It is especially useful in regions of irregular network topology or tight capacity.

The importance of network planning cannot be over-emphasized. It is through network planning that a network evolves. Because the time scale of network planning is weeks or months, when there is a problem in the network, network optimization must be used to provide an immediate fix. That is why network optimization is useful. But network optimization is never meant to replace network planning. Because of the time constraint, the fix may not be the optimal solution. Network planning may subsequently be needed to improve the situation. Network planning also makes a network more robust and makes network operation simpler.

MPLmS is a very important emerging technology. The essence of MPLmS is to use a similar control plane for both optical cross-connects (OXCs) and label switching routers (LSRs). More specifically, the control plane consists of an enhanced link state IGP, constraint-based routing and a signaling protocol. Optical channel trails (OCTs) can then be computed and set up in a similar way as MPLS LSPs. MPLmS shifts some of the traffic engineering work from the network layer to the physical layer.

Inter-domain traffic engineering can be done quantitatively if an end-to-end traffic matrix is available. MPLS provides a new approach for inter-domain traffic engineering. Because an end-to-end traffic matrix is critical for both intra-domain and inter-domain traffic engineering, it is imperative that router vendors provide some mechanisms for constructing traffic matrices. Providing counters to record the amount of traffic from each router to all IP address prefixes is a simple and effective approach.

Many new tools are needed for managing a traffic engineering system. Among them are:

- simulators like WANDL for ensuring that the design of an MPLS system is technically sound before actually deploying it;
- scripts for automatically configuring the routers and their interfaces;
- new SNMP *management information bases* (MIBs) and applications for monitoring the condition of the MPLS system and collecting statistics;
- tools for auditing the system and reporting potential problems.

Fast reroute is very useful for providing QoS for high priority traffic. Because fast reroute introduces significant complexity into a network, it may not be necessary for best effort traffic.

An MPLS-based service architecture can be used to provide Diffserv. In fact, it has some advantages over an IP-based service architecture, because it makes class-based routing and resource allocation in ISPs easier.

Chapter 5 CONSTRAINT-BASED ROUTING WITH ENHANCED LINK STATE IGPS

Constraint-based routing computes paths that are subject to constraints [83]. These constraints can be bandwidth, QoS requirements such as delay and jitter, or other policy requirements on the paths. Constraint-based routing is an important tool for traffic engineering and for providing QoS.

Because network topology information is needed in constraint-based path selection, constraint-based routing must be used together with a link state IGP, and the IGP must also be enhanced to provide the link attribute information as well. In this chapter, we will use the term constraint-based routing to mean constraint-based routing with an enhanced link state IGP.

The organization of this chapter is as follows. We describe issues related to constraint-based routing in Section 5.1. Because many applications have delay requirement, constraint-based routing with delay constraint can be useful. A path selection algorithm that can find paths with delay and jitter constraints is presented. We also discuss the computation complexity of this algorithm and the impact of inaccurate bandwidth information on the result. A simple approach for finding path with delay requirement is then proposed. All these are covered in Section 5.2. In Section 5.3, we describe how QoS routing can be used on the top of the virtual network topology provided by the traffic engineering process to provide QoS in a finer granularity than the class-based approach without significantly increase the amount of state information in the core of networks. In Section 5.4, we describe an algorithm for reducing the routing table computation cost in OSPF. Similar idea can be applied to IS-IS. The inter-domain issue of constraint-based routing, and the relationship between constraint-based routing and MPLS, are addressed in Section 5.5 and Section 5.6, respectively. Finally, the chapter is concluded in Section 5.7.

5.1 Issues of Constraint-based Routing

Constraint-based routing computes routes that are subject to constraints such as bandwidth and administrative policies. Because constraint-based routing considers more than network topology in computing routes, it may find a longer but lightly loaded path better than the heavily loaded shortest path. Congestion can thus be avoided, and QoS requirements of applications can be met.

Important issues of constraint-based routing are discussed below.

5.1.1 Link state information and communication overhead

In addition to network topology information, A router needs link attributes in order to find paths with constraint. Because buffer space can be derived from the input and output rates of the queues and the delay/jitter requirement of the packets, buffer space is usually not considered in constraint-based routing. Therefore, link attributes usually refers to the reservable bandwidth of the links and affinity of the links.

The available bandwidth of a link is equal to the maximum reservable bandwidth of a link (set by network administrators) minus the total bandwidth reserved. It does not depend on the actual amount of available bandwidth on that link. For example, if the maximum reservable bandwidth of a link is 155 Mbps, and the total bandwidth reserved is 100 Mbps, then the reservable bandwidth of the link is 55 Mbps, regardless of whether the link is actually carrying 100 Mbps of traffic or more or less. In other words, constraint-based routing does not compute paths based on instantaneous residual bandwidth of links. This reduces the probability of routing instability.

Affinity, or color, of a link is an administrative property of the link. It is used for network administrators to control whether a link should be considered in LSP path computation. An example on the use of link color is described in Section 4.2.1.

One approach to distribute link attributes is to extend the *link state advertisements* (LSAs) of protocols such as OSPF [71] and IS-IS [70], so that they can carry both topology information and link attributes.

Because link available bandwidth is frequently changing, a tradeoff must be made between the need for accurate information and the need to avoid frequent propagation of LSAs. To reduce the frequency of LSA propagation, changes in link available bandwidth will be announced only when they are significant, e.g., more than 5% of link capacity or more than 10 Mbps. A hold-down timer should also be used to set an upper bound on the frequency on LSA propagation. A recommended minimum interval is 30 seconds [87].

5.1.2 Path constraint and path computation complexity

The commonly used path constraints in constraint-based routing are bandwidth, hop count, delay, jitter, or monetary cost. The path computation algorithms select paths that optimize one or more of these constraints. If multiple constraints are to be optimized simultaneously, the complexity of the algorithms usually becomes very high. Some of them may even be *NP-complete* [85] [89].

Let $d(i,j)$ be a constraint for link (i,j). For any path $P = (i, j, k, \ldots, l, m)$, where $i, j, \ldots, l, m$ are the nodes along the path, constraint d is:

- *additive* if $d(P) = d(i,j) + d(j,k) + \ldots + d(l,m)$;
- *multiplicative* if $d(P) = d(i,j) * d(j,k) * \ldots * d(l,m)$;
- *concave* if $d(P) = min\{d(i,j), d(j,k), \ldots, d(l,m)\}$.

For example, constraints such as delay, jitter, cost and hop count are additive, reliability (1.0 - packet-loss-probability) is multiplicative, and bandwidth is concave.

It has been proved that finding an optimal path subject to constraints on two or more additive and/or multiplicative constraints in any possible combination is NP-complete [89]. As a result, any algorithms that selects any two or more of delay, jitter, hop counts, loss-probability as constraints and tries to optimize them simultaneously are NP-complete. The only computationally feasible combinations of constraints are bandwidth and any of the above.

This conclusion is widely known. However, the proof of NP-completeness is based on the assumptions that: 1) all the constraints are independent; and 2) the delay and jitter of a link are known a priori. But because bandwidth, delay and jitter are not independent in packet networks, polynomial algorithms for finding paths with hop count, delay, jitter and buffer space constraints exist. The complexity of such algorithms is $O(N*E*e)$, where N is the hop count, E is the number of links of the network, and $e \leq E$ is the number of distinct reservable bandwidth values among all links [85]. The algorithm will be discussed in the next section. Nevertheless, the work in [89] can serve as a good indication of the complexity of a path computation algorithm -- even if an algorithm is not NP-complete, it is still very complex.

It should also be noted that if different constraints have different importance, and can be optimized sequentially, then the path selection algorithm becomes much simpler, because only a single constraint needs to be considered at each round of path selection, starting from the most important one. If multiple paths exist after a particular round, the path selection process can be repeated again with the next most important constraint, and so on.

Of all the constraints, bandwidth and hop count are most useful because:

1. Algorithms for finding paths with bandwidth and hop count constraints are simple. The Bellman-Ford (BF) Algorithm can be used. For example, to find the shortest path between two nodes with path bandwidth greater than 1 Mbps and hop count less than h, all links with available bandwidth less than 1 Mbps can be pruned first. The BF Algorithm can then be iterated up to h times. The shortest path found will be the optimal path subject to the constraints [87]. The complexity of such algorithms is $O(N*E)$.
2. Bandwidth constraint is useful because network administrators can use bandwidth constraint to control link utilization. Hop count constraint is also important because the more hops a flow traverses, the more resources it consumes. For example, a 1-Mbps flow that traverses two hops consumes twice as many resources as one that traverses a single hop. If this flow does not use the bandwidth of the second hop, other flows can use it.

3. Delay and jitter constraints can be mapped to bandwidth and hop count constraints [36], if needed, with the queueing delay bound set to (application's delay bound - propagation delay). Propagation delay can be estimated from the distance between the source and the destination, and an inflation factor can be used to account for any detour.

No matter what path constraints are used, routers with constraint-based routing will have to compute its routing table more frequently. This is because, even without topology changes, routing table computation can still be triggered by significant resource availability changes or changes in link affinity. Therefore, even if the complexity of a constraint-based routing algorithm is comparable with the complexity of normal routing algorithms, the computation load on routers with constraint-based routing can be significantly higher. This can hurt stability of the networks.

Common approaches for reducing the computation load of routers with constraint-based routing include:

- using large hold-down timers to reduce routing table computation frequency;
- using administrative policy to prune unsuitable links before computing the routing table. For example, if the bandwidth request of a flow is 1 Mbps, all the links with residual bandwidth less than 1 Mbps can be pruned before the routing table computation. Or, if the flow has delay requirement, high propagation delay links such as satellite links can be pruned before the path computation.

Besides reducing the frequency of routing table computation, another approach for reducing the computation load is to reduce the intensity of each round of routing table computation. One of such approaches is described in Section 5.4.

5.1.3 Routing table structure and storage overhead

With constraint-based routing, the routing table structure needs not be different from normal routing. But if class-based routing is desired, then multiple routing tables need to be maintained, one per class (assuming that every class of traffic can take a different path). In this case, multiple forwarding tables are also needed, one per class.

If multiple copies of routing tables need to be maintained, storage overhead for each router will be significantly higher.

5.1.4 Selection of optimal route

When there are multiple paths satisfying the constraints, several criteria can be used for selecting the optimal route. The key issue is to make a good tradeoff between resource conservation and load balancing.

The first and most important criterion is IGP metric. The shortest path should always be used because this is what network administrators set the IGP metrics for – to control the traffic path.

The second criterion is hop count. A flow traversing a path of fewer hops consumes less link bandwidth, buffer space and processing power. Therefore, minimum hop-count paths should be used as much as possible.

The third criterion is the residual bandwidth on the path. The residual bandwidth of a path is the minimum of the reservable bandwidth of all links along the path. Taking the widest paths as much as possible help to keep the highest link utilization low in the network. A slightly more sophisticated approach is to use the maximum link utilization as the criterion. An example can clearly illustrate the difference between using the two criteria. Assuming that there is a request of 20 Mbps of bandwidth, an empty DS-3 (45Mbps) link and an OC-3c link with 50Mbps of reservable bandwidth (the other 105 Mbps has been reserved) both meet the constraint. If residual bandwidth is used as the criterion, then the OC-3c link will be selected. If the maximum link utilization is used, then the DS-3 link will be used, because the resulted maximum link utilization is smaller.

The order of the second and the third criteria can be switched.

5.1.5 Stability

Network stability has been a major concern of constraint-based routing. The factors that affect stability are discussed below [5][83][84][85][86][87][88][89][90][91][92][93].

The first factor is path re-optimization. For a traffic trunk with established path, when another path with the same IGP metric and hop count but more bandwidth becomes available, if the traffic trunk is shifted to the new path, then stability of a network can be compromised. This is because after the path change, the original path may have more available bandwidth than the new path and cause the traffic trunk to be shifted back again. Therefore, the path re-optimization process should not consider link available bandwidth as a factor. However, a traffic trunk should be allowed to switch to a new path with smaller IGP metric, or with the same IGP metric but fewer hops, when such a new path becomes available.

Routing table computation frequency is also a factor that can impact stability. With constraint-based routing, not only topology changes but also resource changes can trigger a new round of routing table computation. Path re-optimization also makes routing table computation more frequent, because it is usually done on a regular basis, e.g., hourly. More frequent routing table computation means potentially more frequent traffic shift. Therefore, routing table computation (including re-optimization) frequency should be limited. One possible approach is that a router needs not re-compute its routing table every time a change occurs. If a change does not cause the constraints of any path to be violated, the router can simply change its link state database, and wait until the next scheduled re-optimization time to re-compute its routing table.

Computation complexity of constraint-based routing is another factor that can impact stability. With constraint-based routing, the complexity of path computation becomes higher. This is especially true if both IGP route and LSP path need to be computed. This may cause slow routing convergence upon topology changes. Transition period caused by changes therefore becomes longer. The solution to this problem is to use simple path selection algorithm.

5.1.6 Scalability

The scalability issue of constraint-based routing is similar in nature to normal link state IGP. The scalability of a system is determined by requirements on link bandwidth, CPU and memory, and the stability of the system. Because constraint-based routing consumes more link bandwidth, CPU and memory, and can be less stable than normal link state IGP, the scalability issue is more critical for constraint-based routing than for normal link state IGP.

With normal link state IGP such as IS-IS and OSPF, the approach to improve scalability is to introduce multiple areas/levels. Each router only maintains topology information for the area it belongs to. The topology information of each area is summarized and announced to a backbone area (Area 0 in OSPF or Level-2 in IS-IS), and then announced from the backbone area to other areas. Because of the summarization of link state information, a link state change only directly affects routers in that area. Outside the area, the impact is much smaller. Therefore, the link state IGP with multiple areas can scale to larger networks.

The multi-area scheme is essentially a hierarchical routing scheme, with the backbone area at a higher hierarchy than other areas.

The multi-area scheme can still be applied to constraint-based routing, but only to a certain extent. The problem is that source routing is used in constraint-based routing (mainly to avoid routing loops), i.e., every path is computed and specified by the source. Without knowing the complete topology and link attributes of the network, the source cannot determine the complete path. Instead, the source can only determine the part of the path that is inside the area. The area border router that is selected by the source, or by the previous area border router, has to compute the next part of the path, and so on, until the complete path is determined. Path computation thus becomes much more complex. If constraints other than bandwidth and hop count (e.g., delay) are used, then feasible paths can be missed because a wrong area border router is selected. This causes unnecessary path setup failure.

A simple and effective solution to the above problem is to design the network carefully, and provide a certain amount of extra capacity in the network, so that the shortest paths can meet the constraints most of the time. In other words, this is to make the network less dependent on constraint-based routing.

5.2 Constraint-based Routing with Delay and Jitter Constraints

Because some applications have delay and jitter requirements, algorithms for selecting path with delay and jitter are perceived as important by many. In this section, a polynomial algorithm for selecting path with delay and jitter constraints is described [85].

As we can see from this section, algorithms for selecting path with delay and jitter constraints are complex. Because queueing delay and jitter of packets in a flow can be determined by the amount of bandwidth reserved by the flow and the number of hops the flow traverses, algorithms for selecting path based on bandwidth and possibly hop count is practically more useful, except when propagation delay becomes a very significant part of the total delay. But because propagation delay is static for a link, algorithms for selecting path with propagation delay constraint is much simpler than algorithms that take queueing delay into consideration. An algorithm for selecting path with propagation delay is briefly discussed at the end of this section.

5.2.1 Description of the algorithm

Delay, jitter and buffer space can be determined by the bandwidth reserved for a flow. For a flow that conforms to a leaky bucket $<\sigma, b>$ description,

The end-to-end delay bound is:

$D(p,r,b) = b/r + n*L_{max}/r + \sum_{i=1 \text{ to } n} L_{max}/C_i + \sum_{i=1 \text{ to } n} d_i$ (1)

The end-to-end jitter bound is:

$J(p,r,b) = b/r + n*L_{max}/r$ (2)

The buffer space requirement at the h-th hop is:

$$B(p,r,b) = b + h^* L_{max} \qquad (3)$$

Here p is the path for the flow, r is the bandwidth reserved for the flow, n is the hop count of p, L_{max} is the max packet size of the network, C_i is the bandwidth of the i-th link, d_i is the propagation delay of the i-th link .

If the bandwidth to reserve is given, then a single pass of Bellman-Ford (BF) algorithm (up to N hops) will solve the problem. The complexity of BF algorithm is $O(N^*E)$, here E is the number of links in the network [85].

However, if the amount of bandwidth to reserve is unknown, then every possible reservable bandwidth value must be tried. This can be done as follows.

Sort the reservable bandwidth of all links in ascending order $r_1 < r_2 < \ldots < r_k < \ldots < r_e$, $e \le E$ is the number of distinct bandwidth values among all links in the network. For k=1 to e, delete all links with bandwidth less than r_k so that at least r_k amount of bandwidth can be reserved in the remain network. The BF algorithm is then run up to N hops. If there is any feasible path with delay smaller than d and hop count smaller than N, the BF algorithm will find the path. If the path with smallest delay is desired, the BF algorithm can be run e times (for k=1 to e). All feasible paths found are then sorted by delay to produce the one with the smallest delay. This algorithm is called *iterative Bellman-Ford* (IBF) algorithm. Its complexity is $O(N^*E^*e)$ [85].

In order to find feasible paths constrained with delay and jitter, the jitter constraint can be converted into hop count constraint according to formula (2). If the bandwidth to reserve is known, a single iteration of the BF algorithm will produce a feasible path, if there is any. If the bandwidth to reserve is unknown, then the IBF algorithm can be used.

In order to find feasible paths constrained with delay and buffer space, the buffer constraint can be first converted to hop count constraint according to formula (3). Depending on whether the amount of bandwidth to reserve is known or not, the BF or IBF algorithm can then be used.

If a flow can reserve different amount of bandwidth at different links, then the problem of finding optimal paths subject to constraints of delay and jitter is NP-complete [85].

5.2.2 Impact of inaccurate link state information

It is common for enhanced link state protocols to set percentage thresholds for link bandwidth, and flood link bandwidth information only when the bandwidth changes are across thresholds. This means that a router may not have accurate bandwidth information of the other routers because other routers have not announced the up-to-date bandwidth values. It has been proved a path calculated with such inaccurate information will have at most $(1+e)$ times larger latency than the one calculated with accurate information [92]. When $e=1$, i.e., link state advertisements are only flooded when the bandwidth of a link becomes twice as large or half as small, then the delay of the calculated path is at worst 2 times as large.

5.2.3 A constraint-based routing scheme for computing paths with delay constraint

As explained earlier, delay is a useful constraint for path. However, algorithms for computing paths with delay constraint such as the IBF algorithm can be too complex to be practical[1]. The path selection algorithm is complex because queueing delay of each router is not static. However, because propagation delay has become a significant part of the delay, and because the propagation delay of a link is static, path selection based on propagation delay is fairly simple.

One approach to compute paths with propagation delay constraint is to use propagation delay as link metric, and use normal path selection algorithms such as the BF or Dijkstra's algorithm to compute the shortest path.

To account for the queueing delay, the propagation delay requirement can be computed as follows:

Propagation delay = required application delay – queueing delay * hop count

The value of queueing delay at each hop can be set based on experience, e.g., 1 ms. This is an over-estimation of normal queueing delay. The hop count can also be over estimated. This will ensure that the propagation delay supplied to the path selection algorithm is smaller than the actual tolerable propagation delay. If the path selection algorithm can find a path that meets the propagation delay constraint, then the path will meet the application's need[2].

5.3 QoS Routing

QoS routing (QoSR) is a subset of constraint-based routing that focuses on providing QoS for application flows [83][84][85][86][87][88][89][90][91][92][93]. QoS routing

[1] The path selection will be simpler if the amount of bandwidth to reserve for each flow is known. But that requires fine granularity resource reservation, i.e., for each flow.

[2] Some of the ideas in this approach come from A. L. Chiu of AT&T and Y. Rekhter of Cisco.

finds paths for application flows (or flow aggregations) such that the QoS requirements of the flows can be met.

In a network meshed (not necessarily fully meshed) with MPLS LSPs, each LSP can be treated as a link with capacity equal to the bandwidth of the LSP. We then propose that QoS routing can be further used to find the optimal paths in the virtual network for application flows or flow aggregations that are dynamically signaled. If this is the case, LSPs for the application flows can be the built on the top of the LSPs that are used for traffic engineering. A LSP stack is thus created. In a sense, this is similar to putting ATM VCs inside VPs.

To be more specific, constraint-based routing can be used twice in providing QoS. At the first time, constraint-based routing is used for traffic engineering. As a result, the capacity for each (source, destination) pair is set and a virtual network fully or partially meshed with MPLS LSPs is constructed on the top of the physical network. In the traffic engineering process, constraint-based routing is done on a daily or weekly basis, except maybe when there is network topology change. This process is similar to dimensioning [94][95][96][97] in circuit switching networks. At the second time, constraint-based routing is used to find optimal paths for application flows or flow aggregations with QoS requirement. This process is specifically referred to as QoS routing. QoS routing can be done more frequently, triggered by the dynamic requests of application flows. The LSPs created in the traffic engineering process are treated as links in the QoS routing process. The constraints of the paths in the processes of traffic engineering and QoS routing are likely to be different.

Compared to the approach that LSPs created in the traffic engineering process are re-optimized more frequently but there is no QoS routing, the advantage of traffic engineering plus QoS routing is that it is possible to provide QoS at a much finer granularity with approximately the same amount of state information in the core, because the transit LSRs in the middle of the TE-LSPs need not maintain any state information for the QoSR-LSPs. This is achieved by building the QoSR-LSPs for application flows on the top of the TE-LSPs. A LSP stack is therefore required.

If QoS routing is performed on the top of a virtual network topology created by traffic engineering, it becomes very similar the *dynamic routing* process [98][99][100][101][102][103][104][105][106][107][108][109] in circuit switching networks. In circuit switching networks, the transmission capacity from one node to another is pre-set by a process called dimensioning. Therefore, the underlying networks can be viewed as fully connected by logical links with pre-set capacity.

Two well-known dynamic routing schemes in circuit switch networks are briefly reviewed next.

5.3.1 Dynamic routing in circuit-switched networks

Two successful dynamic routing algorithms in telephone networks are the *dynamic alternative routing* (DAR) [98] and the *real-time network routing* (RTNR) [109].

DAR is jointly developed by the University of Cambridge's Statistical Lab led by F. Kelly, and the British Telecom (BT) Lab led by D. Songhurst. It has been shown to be efficient and stable in the BT network.

In DAR, when a call is received, it is routed through the direct logical link to the destination. If the direct link is blocked, a pre-selected alternative route that consists of two logical links is tried. Because a call that is routed through the alternative route consumes more resources than through the direct route, alternative routing is not desirable when the traffic load of the networks is heavy. Therefore, a certain amount of link capacity, e.g. 80%, is reserved for direct calls only. alternative calls are admitted only when the link load is under 80%. This scheme is called *trunk reservation* [98] and is widely used in circuit switching networks. If the call is rejected on the pre-selected alternative route, the call is rejected. Another alternative route is randomly chosen as the new preferred route. If the call is admitted via the preferred alternative route, nothing needs to be changed. In this way, if calls cannot be routed through the direct link, they stick to the preferred alternative route as long as possible. This scheme is therefore called sticky routing. DAR is basically a combination of trunk reservation and sticky routing. Note that in DAR, all the information used are local.

RTNR is similar to DAR, except that:

1. RTNR can handle several classes of services. Trunk reservation is done on a per-class basis;
2. RTNR uses global link state information to select alternate route.

RTNR is used in AT&T's telephone network.

5.3.2 A QoS routing scheme

In Chapter 4, we described a TE system with a core LSP mesh and multiple regional meshes. A QoS routing scheme for such a network is described here.

1. If the source and the destination are in the same region

The QoS routing scheme is the same as DAR. That is, if the direct path between the source and the destination is available, it will be used; otherwise, a tandem node is picked such that the path is shortest. The same tandem node will always be used as long as the path meets the constraints. If the call is rejected along that alternative path, then the call is rejected, and the next 2-hop shortest path is selected.

2. If the source and the destination are in different regions

Pick the shortest path if it meets the constraints; Otherwise, switch to the next shortest path and stick to it until it does not meet the constraints any more.

5.4 An Approach for Reducing the Routing Table Computation Cost for OSPF

Constraint-based routing has higher computation complexity than normal routing. High computation complexity can hurt stability of a network because it makes routing convergence slower in the case of change. Therefore, any approach that can reduce the computation load of the routers can improve the performance and stability of a network.

One approach to reduce the computation load of routers is to reduce the routing table computation frequency, as explained in Section 5.1.2. Another approach is to reduce the computation load of each round of routing table computation. An approach is described in this section. The basic idea is as follows.

In link state protocols, because the complete topology of the network is known, divide-and-conquer schemes can be used to make protocol and other control processing more efficient. We present a simple approach for a router running the *open shortest path first* (OSPF) [71] to automatically detect if its interfaces lead to multiple disjoint regions within an OSPF *area*. If such disjoint regions exist, this approach can make the routing table computation more efficient. This approach requires no change in OSPF, and can be immediately applied to current routers [111][112]. The approach can also be applied to IS-IS.

The organization of the rest of this section is as follows. We give a brief introduction to OSPF in 5.4.1. In Section 0, we present an efficient routing table computation approach for OSPF and illustrate its correctness. The parallelism in this approach is explored in Section 5.4.2. The performance enhancement is shown in Section 5.4.3. Finally, the section is summarized 5.4.4.

5.4.1 Introduction to OSPF

The Internet consists of multiple *autonomous systems* (ASs). An AS is a collection of networks and routers under a single administrative authority. Between ASs, BGP is used to compute the routing table. Within an AS, OSPF or IS-IS is used. We focus on OSPF in this section, but the approach can also be applied to IS-IS.

In OSPF, an AS can be divided into multiple areas. An AS with four areas is shown in Fig. 5-1 (taken from the OSPF RFC), where the cost of each link is 1 unless specified differently [71]. OSPF distributes the information of every link to all routers inside that area. Such topology information is stored in the *link state database* (LSDB) of each router. Each router computes the shortest paths (called routes) to all other routers using Dijkstra's algorithm [117][118]. After the initial computation, if a link in an area changes, every router inside that area must re-compute routes to all the destinations, inside and outside the area. For example in Fig. 5-1, if the link between router RT1 and network n1 goes down, RT1 must re-compute all routes including those to networks n3, n6 and n9. If a link outside the area changes, a router only updates the routes affected by this change. There is no need to re-compute the whole routing table [71].

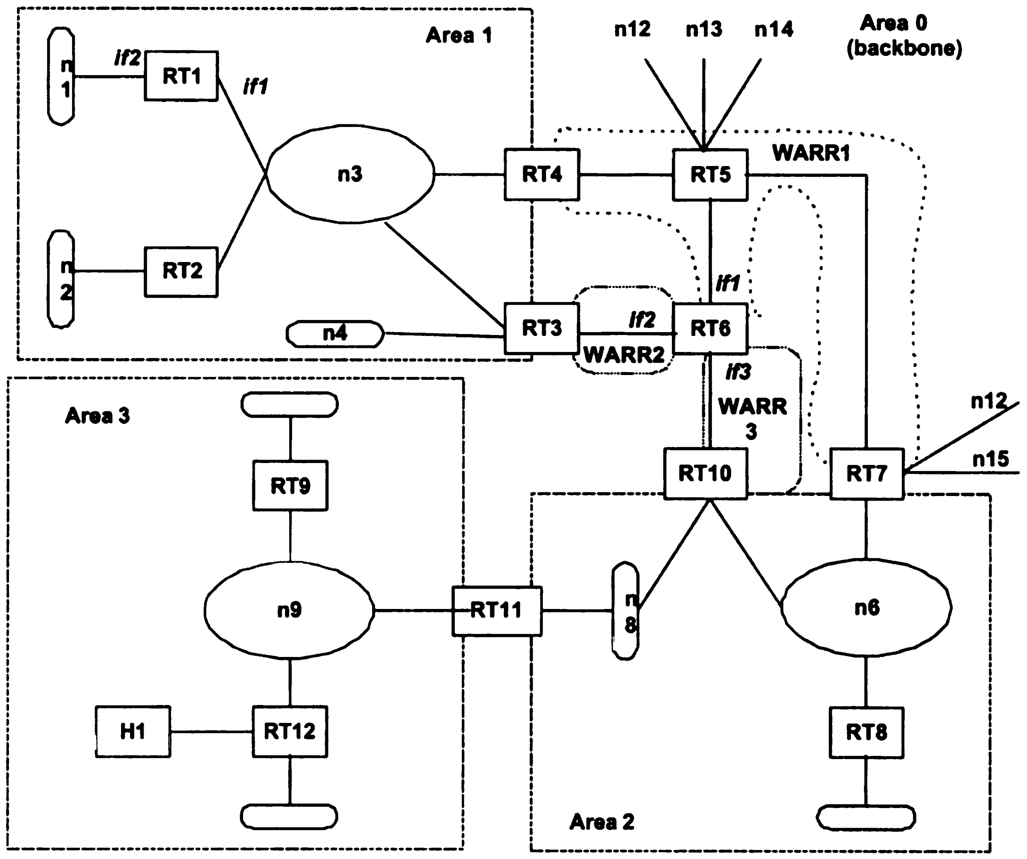


Figure 5-1 A sample autonomous system with 4 areas

Efficient routing table computation for OSPF

The idea is based on the following observation. In OSPF, when there is an intra-area link change, it is necessary for every router inside that area to re-compute all routes in its routing table. However, if the topology of the network is somewhat special, then some of the work may be saved [111]. For example in Fig. 5-1, if link RT1-n1 goes down, RT1 does not have to re-compute routes to networks n3, n6 and n9. We define the *routing region* (RR) of an interface on router R as all the networks and routers reachable from that interface without going through other interfaces of R. For RT1, the RR of interface if2 consists solely of network n1; the RR of if1 consists of all the routers and networks except n1 and RT1 itself. When link RT1-n1 goes down, the reasons why RT1 does not have to re-compute routes to networks n3, n6 and n9 are: 1) link RT1-n1 is in the RR of if2; 2) networks n3, n6 and n9 are in the RR of if1; and 3) the RRs of if1 and if2 are disjoint. The point in this example is, if a router can detect that its interfaces have multiple disjoint RRs (Fig. 5-2a), and can locate the RR where a change occurs, then it can re-compute only routes in the changed RR. We show that this is possible with a link state protocol like OSPF.

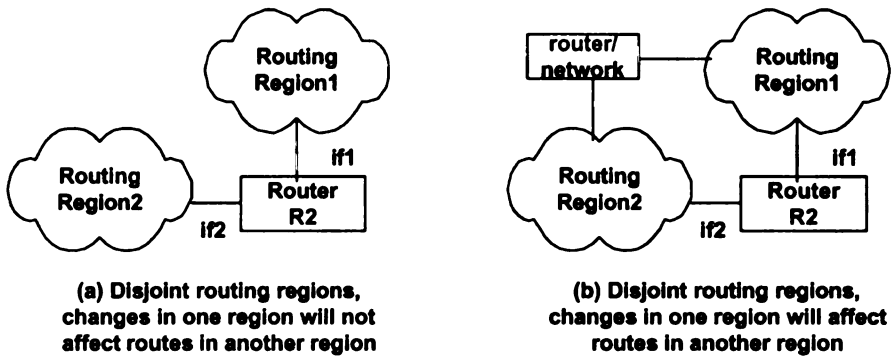

Routing Region1
if1
Routing Region2
Router R2
if2
(a) Disjoint routing regions, changes in one region will not affect routes in another region
router/ network
Routing Region1
if1
Routing Region2
Router R2
if2
(b) Disjoint routing regions, changes in one region will affect routes in another region

Figure 5-2 Routing regions

An intuitive way to do this would be to add some mechanisms to a router so that it can: 1) automatically discover if some or all of its interfaces have disjoint RRs; and 2) determine in which RR a change is. For such an enhanced router, when there is a link change in one RR, if there are some disjoint RRs, computation work can be reduced because routes through other disjoint RRs need not be re-computed (Fig. 5-2a). However, if all the RRs are overlapped, the whole routing table must be re-computed (Fig. 5-2b). But even in the second case, this approach is still as efficient as OSPF, because both approaches re-compute the whole routing table. The more disjoint RRs, the more efficient this approach is. It seems that as long as the enhancing mechanisms are simple enough, this intuitive approach would work well.

But an important question is how often different interfaces of a router will have disjoint RRs. If we look at the whole AS, a destination can usually be reached via different interfaces of a router. In other words, the RRs of these interfaces will overlap somewhere in the AS. Examining the RRs of the routers R3, R4, R5 and R6 in Fig. 5-1 clearly shows this. Therefore, the intuitive approach does not work because it will always be identical to the OSPF approach. To limit the scope of impact of link changes, we must find a new dividing scheme other than dividing by RRs.

5.4.1.1 Description of the algorithm

The key observation in solving the problem is that, although the RRs of two interfaces are likely to overlap somewhere in the AS, they are unlikely to overlap within the area. We call the part of a RR that is within the area as *within-area routing region* (WARR). It is desirable that we divide the computation work according to disjoint WARRs instead of disjoint RRs. We present such an algorithm in this section. As long as the WARRs of two interfaces are disjoint, no matter whether the RRs are overlapped outside the area or not, the algorithm can still limit the scope of impact of a link change to mostly within the changed WARR.

Formally, the WARR of an interface of router R consists of all the routers and networks within the area that are reachable from that interface without going through other interfaces of router R. Networks and routers outside the area are completely ignored in defining WARR. For router RT1 in Fig. 5-1, the WARR of interface 1 (if1) consists of networks n2, n3, n4 and routers RT2, RT3, RT4; the WARR of if2 consists solely of network n1. The WARRs of the three interfaces of RT6 are shown in Fig. 5-1. Note that an area is divided into WARRs differently from different routers' perspective. Under this definition, WARRs of all interfaces on all routers in Fig. 5-1 are disjoint. This is a strong indication that the WARR approach is a good dividing scheme. Maps of real world networks also show that disjoint WARRs are common for routers in corporate networks, and for the edge routers in ISPs [115].

An important property of WARR is that the WARRs of two interfaces are either disjoint or completely overlapped, i.e., identical. It will never happen that two interfaces have different WARRs but these two WARRs are partially overlapped. This is because WARR is defined on *reachability* [116] If two WARRs are overlapped, one WARR is reachable from the other. By definition, the two WARRs would be considered as a single one (Fig. 5-2b). We call this the equivalence property of WARRs. If the WARRs of two interfaces are disjoint, routes through one WARR has nothing to do with the other WARR, therefore each interface can independently compute its part of the routing table through its own WARR. If the two WARRs are identical, just one interface needs to compute the routes. Since all interfaces share a single routing table, the other interface does not have to compute the routes. It simply uses the routes computed by the other interface.

Assuming that a router has mechanisms to automatically detect whether its interfaces have disjoint WARRs, and to locate the WARR where a change occurs, our approach can be summarized conceptually below. The mechanisms themselves are described in Section 5.4.1.3.

Step 0. Outside-area changes are processed in the same way as in OSPF

Step 1. One WARR agent, which can be a process or a thread, is used to compute the routes for each disjoint WARRs. All agents share the same routing table, but they build/maintain different parts of it, one part per agent.

Step 2. Each agent only uses the topology information of its own WARR to compute the routes.

Step 3. After the computation, the changed routes are selectively installed into the routing table of the router.

Compared to OSPF, Step 2 only uses topology information of the changed WARR to compute the routing table while OSPF uses topology information of the whole area. Since RRs can overlap outside the area for disjoint WARRs, special care must be taken to ensure that the computed routes are indeed shortest. This is done in Step 3. Before detailing Step 3, we illustrate the algorithm with two examples.

Case 1: disjoint WARRs without common outside-area destinations.

The two interfaces of router RT1 have disjoint WARRs, and these two WARRs do not lead to any common destinations outside the area. In Step 2, the algorithm computes the route to network n1 through the WARR of if2, and routes to other destinations through the WARR of if1. If link RT1-n1 changes later, the algorithm only re-computes the route to n1 in Step 2. Routes to other destinations are not affected at all. Similarly, if there are any changes in the WARR of if1, the route to n1 does not need to be re-computed. In this case, using only topology information of the changed WARR to re-compute the routing table does not affect the correctness of the algorithm. Compared to the OSPF approach, the computation work is reduced.

Case 2: disjoint WARRs with common outside-area destinations. In Fig. 5-1, if1 and if2 of router RT6 have disjoint WARR1 and WARR2, respectively. But both WARRs lead to network n3 in area 1. In Step 2, the algorithm will find two routes to network n3, through WARR1 and WARR2, with cost 3 and 2, respectively. Only the route through WARR2 with cost 2 should be installed into the routing table. Later, if the cost of link RT6-RT3 increases, e.g. from 1 to 10, the algorithm will again be triggered. After the re-computation in Step 2, the agent of WARR2 will find out that the cost of the route to n3 (only through WARR2) becomes 11. Obviously, this is not the shortest path from RT6 to n3. Although WARR2 previously provided the shortest path to n3, after the change, other WARRs may have shorter paths to n3. Therefore, the algorithm must try to re-compute shorter paths through other WARRs to n3. If there are indeed any shorter routes, as is true in this case (the route through WARR1 with cost 3 is shorter), the shortest one of them should be installed as the route to n3. This is what Step 3 must do.

```
Step 3: After Step 2, if the new route is
3.1        For destinations within the area, it replaces the current route
3.2        Otherwise (for destinations outside the area):
    3.2.1        If the cost of the new route is lower than the current route: replace
                 the current route
    3.2.2        Otherwise, if the cost of the new route is higher:
        3.2.2.1        If the new route and the current route go through the same
                       WARR:
                       Replace the current route with the new route, and trigger the OSPF
                       agents of other WARRs to compute routes for this destination.
```

3.2.2.2 Otherwise (through different WARRs): discard the new route

3.2.3 Otherwise, the cost of the new route is equal to the cost of the current route:

3.2.3.1 If the new route and the current route go through the same WARR:

Replace the current route with the new route

3.2.3.2 Otherwise: append the new route to the current route

The rationale of this algorithm will be illustrated in the next sub-section.

We use RT6 as the computing router to illustrate the idea of Step 3. The current route from RT6 to n3 consists of link RT6-RT3 and RT3-n3 with cost 2. If the cost of link RT6-RT3 increasing from 1 to 10, it will be handled by Step 3.2.2.1. The example in case 2 clearly illustrates why a re-computation is needed. Step 3.2.2.2 handles cases such as the cost of link RT6-RT5 increasing from 1 to 10. The new route of cost 12 through WARR1 should simply be discarded since it is worse than the current route. Similarly, in Step 3.2.3.1, the fact that the new route and the current route are through the same WARR means that the current route has disappeared. Therefore the new route should replace the current route. If the new route is in a different WARR, then the current route still exists. The new route should be appended.

In Step 3.2.2.1, changes in one WARR can trigger computations in other WARRs. But much work is still eliminated compared to OSPF. This is because the algorithm only recomputes routes for those destinations outside the area whose cost has increased. The number of such routes is far less than the total number of routes in the routing table. For example, when the cost of link RT6-RT3 increases from 1 to 10, only routes to networks in area 1 will be affected. The algorithm does not have to re-compute routes to any destinations in area 2 and area 3. OSPF does.

In essence, the introduction of WARRs adds another hierarchy to OSPF, just as the introduction of areas does. The difference is that areas have to be manually configured by the network administrators while WARRs are automatically discovered by the algorithm.

5.4.1.2 Correctness of the algorithm

Correctness of the algorithm can be illustrated below:

1. For an outside-area link change, the algorithm works exactly the same as OSPF and is thus correct.
2. For an intra-area link change, the agent for the changed WARR is guaranteed to receive this link state advertisement (LSA). Compared to OSPF, Step 2 of the algorithm only uses topology information (LSAs) of the changed WARR to compute the routing table, while OSPF uses topology information of all WARRs (i.e., the whole area). But all problems introduced by this difference are fixed in Step 3.

3.1 For destinations within the area, by the *equivalence property* [116] of WARRs, these destinations are unreachable from other disjoint WARRs. Therefore, ignoring LSAs of other WARRs will not affect the correctness of the algorithm.

3.2 For destinations outside the area:

3.2.1 If the cost of the new route is lower than the current route:

Since the change is an intra-area one, routes through other WARRs (if any) remain the same. Their cost is at least the same as the current route (i.e., the current shortest path). Therefore, the new route with lower cost can replace the current route without further computations through other WARRs.

3.2.2 If the cost of the new route is higher:

3.2.2.1 If the new route and the current route go through the same WARR, it implies that the change has caused the current route to become longer. Therefore, the new route must replace the current route. But after this increase in cost, other WARRs may provide shorter routes to the destinations. Other agents must be triggered to compute routes through other WARRs for those destinations. If other WARRs can indeed provide shorter routes, the shortest one of them will be eventually entered into the routing table at Step 3.2.1

3.2.2.2 If the new route and the current route go through different WARRs, since the WARR of the current route is not affected at all, the current route still exists. The new route with higher cost must be discarded.

3.2.3 Otherwise, the costs of the new route and the current route are the same:

3.2.3.1 If the new route and the current route go through the same WARR, it implies that the current route has disappeared. It should be replaced by the new route.

3.2.3.2 If the new route and the current route go through different WARRs, then the current route still exists. The new route should be appended (assume ECMP is enabled).

In OSPF, routing table computations can only be triggered by either outside-area link changes or intra-area link changes. For intra-area link changes, a changed route can only be for a destination inside (handled by Step 3.1) or outside the area (handled by Step 3.2). For destinations outside the area, the route can only become shorter (handled by Step 3.2.1), or become longer (handled by Step 3.2.2), or remain the same (handled by Step 3.2.3). For a new route of higher or equal cost, it can only be in the same WARR of the current route or in a different WARR (handled by Steps 3.2.2 and 3.2.3, respectively). We have shown that the algorithm can handle all possible cases correctly. Its correctness is therefore guaranteed [111].

5.4.1.3 Implementation Issues

In this sub-section, we discuss how to represent WARR in the data structure, how to determine in which WARR a change is, how to detect if the WARRs of two interfaces are overlapped or not, and how to handle WARR merging and splitting. These turn out to be surprisingly easy.

5.4.1.3.1 Detecting the WARR of an interface and locating the WARR of a link state change

In OSPF, topology information of an area is stored in the router's link state database (LSDB), which is composed of link state advertisements (LSAs). By adding the IP address of the receiving interface to each LSA, we can distinguish LSAs of one interface from others. All LSAs of an interface belong to the interface's WARR. If multiple interfaces share the same WARR, LSAs of all these interfaces belong to that WARR. Since changes are reflected by LSAs, we can easily determine in which WARR a change is by looking at the tagged interface IP address of the corresponding LSA. When there is a change in a particular WARR, only LSAs of that WARR are used to re-compute the routing table. Therefore, routes in other WARRs will not be affected.

Note that, although we mention detecting the WARR of an interface and locating the WARR of a link state change many times in this paper, detecting the WARR of an interface and locating the WARR of a change are only logical concept. The mechanisms described in the previous paragraph are the actual implementation mechanisms for those purposes. More specifically, the computing agent of an interface actually does not need to know explicitly which networks/routers belong to its WARR. What the agent needs to do is to only use LSAs of its WARR in re-computing routes. This makes the implementation of the algorithm very simple.

5.4.1.3.2 The handling of overlapped WARRs of multiple interfaces

Two overlapped WARRs will be detected when the WARR agent for one interface tries to enter a route for an intra-area destination into the routing table, but finds out that the WARR agent of the other interface has already entered a route for that destination. The fact that both interfaces can reach an intra-area destination implies that their WARRs are overlapped. By the equivalence property of WARRs, these two WARRs are identical. Therefore we can let one WARR agent (say, the one for the interface with lower IP address) continue to compute the routes, using LSAs of both interfaces. The WARR agent of the other interface is terminated so that redundant work will not be done. After discovering that two interfaces share the same WARR, the continuing agent must immediately re-compute the routes using LSAs of both interfaces. The previous works of both agents are wasted. Therefore, strictly speaking, the algorithm will do a little more work than OSPF if all interfaces share the same WARR. But this can only happen when computing the routing table for the very first time because, after that, overlapped WARRs would have been discovered. Since WARRs of different interfaces are likely to be disjoint, this should be acceptable.

5.4.1.3.3 The handling of WARR merging and splitting

There are still two subtle cases to consider: 1) the WARRs of two interfaces may merge when a link is added to an area; and 2) a merged WARR may split into two disjoint WARRs when a link is removed. For example, if we add a link in area 0 between RT3 and RT4, the WARRs of if1 and if2 of RT6 will merge. After the merging, two interfaces share the same WARR. This will be detected by the mechanism described above. One WARR agent will continue and the other is terminated. Note that in the case of merging, the algorithm does less work than OSPF. OSPF has to re-compute the whole routing table while the algorithm only re-compute routes through the merged WARR.

The handling of WARR splitting is a bit more complex. Imagining that after the merging, link RT3-RT4 is removed again. The merged WARR will split into two disjoint WARRs. When this happens, the WARR agent will find out that the other interface of the WARR is unreachable from outside the router. The algorithm intentionally ignores it for a period of no less than 30 seconds. During this period, it still treats the two WARRs as a single big WARR, as OSPF does. (In fact, OSPF always treats all WARRs as a single big one). The WARR agent will use LSAs of both WARR1 and WARR2 in re-computing the routes. There is nothing wrong with this except that some unnecessary work may be done. If the two WARRs do not merge by the end of this period, another WARR agent will be created for the new WARR. From now on, each WARR agent computes its own routes using only LSAs of its own interface. Unnecessary work is then avoided. The reasons why the algorithm ignores the splitting for a period of no less than 30 seconds are: 1) LSAs of the WARRs are guaranteed to be received by the corresponding interfaces after 30 seconds [81]; 2) the two WARRs may merge again during this period if the link failure is temporary; and 3) the algorithm performs no worse than OSPF when the splitting is ignored. A simpler implementation is to ignore WARR splitting totally. The algorithm performs no worse than OSPF anyway.

5.4.1.3.4 Summary of implementation

The implementation of our approach is summarized below:

1. When a LSA is received from an interface, it is tagged with the IP address of that interface.
2. A WARR agent only uses the LSAs of its own WARR to compute the routes. All agents share the same routing table, but they build/maintain different parts of it, one part per agent.
3. At the beginning, a WARR agent is used to compute the routes for each interface. If overlapped WARRs are detected later for multiple interfaces, only one agent will continue. Agents for other interfaces are terminated. The continuing agent immediately re-computes the routes using LSAs of all the interfaces. When a WARR splits into multiple WARRs, new agents will be created to compute the routes after 30 seconds, one per WARR.

Dividing an area into disjoint WARRs has more advantages than just reducing the computation work. Because routes through different WARRs have nothing to do with each other, they can be computed in parallel, as described in Section 5.4.2.

5.4.2 Parallel Routing Table Computation

Plenty of parallelism exists in this routing table computation algorithm. First, since routes through different WARRs are mutually independent, if multiple link changes happen at about the same time in different WARRs, they can be processed in parallel. Second, in Step 3.2.2.1, after replacing an existing route with a new one with higher cost, agents for other disjoint WARRs will be triggered to search for shorter paths. These computations can be done in parallel. If the router has multiple CPUs on the control card, multiple WARR agents can work simultaneously.

In a multi-threaded OS, a thread can be used for each WARR agent. Overhead to create a thread is low. Different threads share the same address space where the routing table and the LSDB are stored. In this algorithm, threads of different WARRs do not have to communicate with each other. Therefore, a parallel implementation of the algorithm will be simple, have high speedup and very low overhead [112].

5.4.3 Performance Improvement and Related Work

Performance improvement of this approach comes from two sources: the efficiency and the parallelism.

Efficiency comes from avoiding unnecessary work. By dividing an area into multiple disjoint WARRs, we limit the scope of impact of an intra-area link change to only within the WARR and to outside-area destinations that are reachable from that

WARR. If an area is divided into *n* disjoint WARRs, the *size* (defined as the total number of networks and routers) of each WARR will be approximately $1/n$ the size of the area (an ideal case in our algorithm). For intra-area link changes, the *Dijkstra's Algorithm* used in OSPF has a time complexity of $O(s\log s)$, where *s* is the size of the area. The speedup of our approach, gained from reducing the number of networks and routers involved (from an area-size down to a WARR-size), can be more than *n*. Although disjoint WARRs can lead to common destinations outside the area, which makes it necessary to compute routes through other WARRs for those destinations whose cost increased (Step 3.2.2.1), the number of changed routes is usually far less than the total number of routes. Even less is the number of routes with increased cost. Therefore, the performance improvement is still significant in this case.

Plenty of parallelism exists in our approach: 1) multiple intra-area link changes in different WARRs can be processed simultaneously; and 2) when a route with higher cost replaces the current route (Step 3.2.2.1), different agents can simultaneously search for shorter paths through other WARRs for that destination. The number of agents running simultaneously can be as high as *n* in the first case, and $n-1$ in the second case. If the router has *m* CPUs, the speedup of parallel processing can be as high as $min(n-1, m)$.

Combining improvements from both efficiency and parallelism, the best case performance improvement of our approach is $n*min(n-1, m)$.

In real world, the plethora of redundant routes for ISP core routers makes disjoint WARRs less likely. However, disjoint WARRs are common in corporate networks, as shown by their network maps [115]. Therefore, the approach presented in this section is more useful for corporate networks than for ISPs. The approach can also be useful for

ISP edge routers. Given that the implementation of this approach is simple, and that the improvement can be significant, this approach is useful, even if it is only used for corporate networks.

To the best of the authors' knowledge, there is no previous work in using the complete topology information of link state protocols to reduce their computation overhead. Some remotely related works include the *Zebra* project [121] in Japan and the *multi-threaded routing toolkit* (MRT) Project [119][120] of Merit Inc. In the Zebra project, routing daemon *Gated* [122] is implemented with multiple processes, one per protocol. The MRT project is similar except that a thread is used for each protocol. These approaches do not intend to provide much performance improvement over the monolithic Gated. They are mainly for software modularity and scalability.

5.4.4 Summary

This section presents a novel point, i.e., in a link state protocol, because the topology of the network is known, divide-and-conquer schemes can be applied to make the processing more efficient. As an example, we present a simple approach for a router running OSPF to automatically detect if its interfaces have multiple disjoint WARRs. If such disjoint WARRs exist, this approach can limit the scope of impact of an intra-area link state change to mostly within the changed WARR. More specifically, only routes through that WARR, not all the routes in the routing table, need to be re-computed. Compared to OSPF which re-computes the whole routing table, this approach can reduce the computation work by more than n times in ideal cases, where n is the number of disjoint WARRs. Furthermore, route computations in different WARRs are independent and can be done in parallel to give an additional speedup of $min(n\text{-}1, m)$, where m is the number of CPUs in the router. The correctness of the algorithm is illustrated.

The implementation of our approach is very simple, as summarized at the end of Section 5.4.1.3.4. It requires no change to the OSPF Protocol itself. Routers using this approach can immediately improve their performance. The improvement does not have to wait until other routers also use this approach. This approach is especially suitable for routers in corporate networks.

Divide-and-conquer approaches can be applied to other link state protocols such as IS-IS and OSPF to reduce the computation overhead. With the coming of QoS routing, link state protocols will become widely used to provide the much needed topology information. Approaches that can reduce protocol-processing overhead, such as the one presented in this section, will be highly demanded.

5.5 Inter-domain constraint-based routing

Most large ISPs have direct peering with each other, either public or private. Therefore, for customers that subscribe to large ISPs, an application flow traverses at most two ISPs most of the time – the source ISP and the destination ISP. Therefore, mechanisms for selecting a transit domain among multiple candidates for an application flow are not really useful. In other words, there is little need for inter-domain constraint-based routing. For ISPs to provide QoS, in addition to support Diffserv and intra-domain traffic engineering, the only thing that the source ISP needs to do is to select the right egress point for the application flows among multiple candidates, and make sure that the egress point is not congested. From this perspective, inter-domain constraint-based routing is taken cared of by inter-domain traffic engineering.

5.6 Relationship between constraint-based routing and MPLS

Constraint-based routing and MPLS need not necessarily be used together. Each router can used constraint-based routing to compute the next hop for a packet, and forward it to the next hop. The next hop will do the same thing. This process repeats at each router along the path until the packet reaches the destination. MPLS can be used just as a forwarding scheme, where the LSP from a particular source to a particular destination just follow the same path as IGP route [5].

However, when constraint-based routing and MPLS are used together, they make each other more useful. With constraint-based routing, routers can have inconsistent information, either regarding topology or resource. If paths of packets are determined by many routers hop by hop as described above, routing loops can be created because of different routers may have an inconsistent view of the network. Source routing is therefore desirable in constraint-based routing. MPLS makes source routing more efficient, because the complete path of a packet is determined by the label inserted into the packet at the ingress LSR. Furthermore, constraint-based routing enables an MPLS-capable router to determine the paths for LSPs based on some constraints. Forwarding traffic along such paths can reduce the probability of network congestion and increase network efficiency. Therefore, when constraint-based routing and MPLS are used together, they become useful tools for traffic engineering [5].

5.7 Conclusion

Constraint-based routing is an important tool for traffic engineering and for providing QoS. In this chapter, we first discuss the issues related to constraint-based routing in general. Algorithms and issues related to constraint-based routing with delay and jitter constraints are then presented. A simple approach for finding paths with delay constraint is proposed. We then describe QoS routing, the aspect of constraint-based routing that focuses on finding feasible paths for application flows or flow aggregations with QoS requirement, and propose a simple QoS routing scheme that can be used on the top of a virtual network constructed in the traffic engineering process. We also describe an approach for reducing the routing table computation cost for OSPF. The inter-domain issues of constraint-based routing, and the relationship between constraint-based routing and MPLS, are then discussed.

The issues and approaches discussed in this chapter apply to online constraint-based routing as well as offline constraint-based routing.

Chapter 6 QoS Schemes at the Application Layer

In this chapter, we discuss the application-layer approaches for improving the performance of Internet traffic, especially Web traffic. We focus on traffic directing that is used to influence traffic distribution in the wide area, and load balancing that is used to reduce the load on the servers and improve service availability. These approaches are very useful for providing QoS in the Internet. We also discuss the relationship between traffic directing, load balancing and other schemes for providing QoS in the Internet.

6.1 Traffic Directing in the Wide Area

Currently, the majority of the Internet traffic is Web traffic. Therefore, improving the performance of Web traffic is important and useful. A number of approaches for improving performance of Web traffic have appeared [47][48][49]. The most conspicuous one is the FreeFlow approach [47]. It is briefly reviewed below.

6.1.1 Traffic directing

In order to understand how FreeFlow works, one needs to understand first how Web browsers and the *hypertext transfer protocol* (HTTP) [126] work. When a user accesses a *universal resource locator* (URL), e.g., http://www.yahoo.com/index.html, what the browser does is initiating a TCP connection to the specified Web server, www.yahoo.com in this case, to request for page index.html. This page will contain text as well as objects such as banners, pictures, or JAVA applets. When server www.yahoo.com replies to the browser's request, it does not send everything back to the browser at the same time. Instead, it sends only the text in the page, and URLs to the objects embedded in the page. The Web browser then requests for the embedded objects. Since the URLs for the embedded objects point to the same Web server, i.e., www.yahoo.com, the browser will again get the objects from Yahoo's Web server. Because the amount of data of the objects is usually far larger than that of the text, it takes much more time to transport the objects than to transport the text. This process is illustrated in Fig. 6-1.

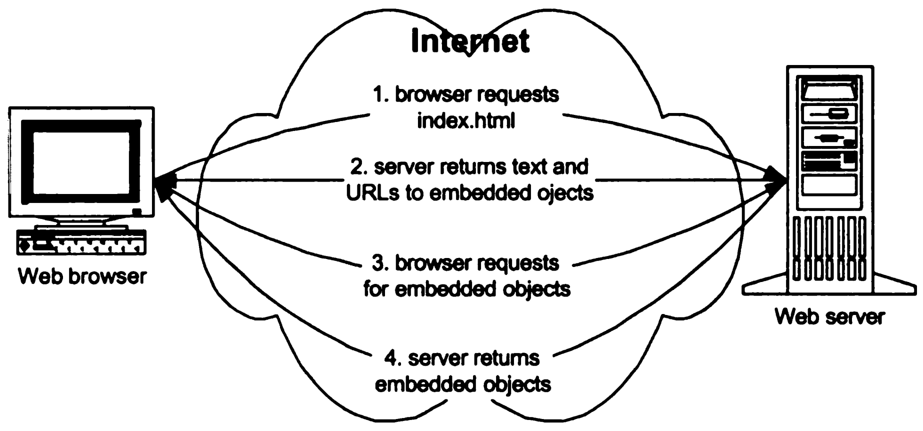

Internet
1. browser requests index.html
2. server returns text and URLs to embedded ojects
3. browser requests for embedded objects
4. server returns embedded objects
Web browser
Web server

Figure 6-1 Content delivery without traffic directing

The innovation of the FreeFlow approach is changing the URLs of the embedded objects to pointing to other servers, not the original Web server that provides the text. In order to do this, the original Web server, i.e., www.yahoo.com, will have to run some program to have their *hypertext markup language* (HTML) [127] files changed. This program changes the URLs of the embedded objects from pointing at www.yahoo.com to pointing at another server. This server can be dynamically picked based on some statistics of network performance. Such servers with synchronized content are deployed in many different places. Therefore the server picked is usually far closer geographically to the browser than www.yahoo.com, and the response time from such a server to the browser is much shorter. The performance of Web traffic is therefore improved. We call this process traffic directing (TD). It is illustrated in Fig. 6-2.

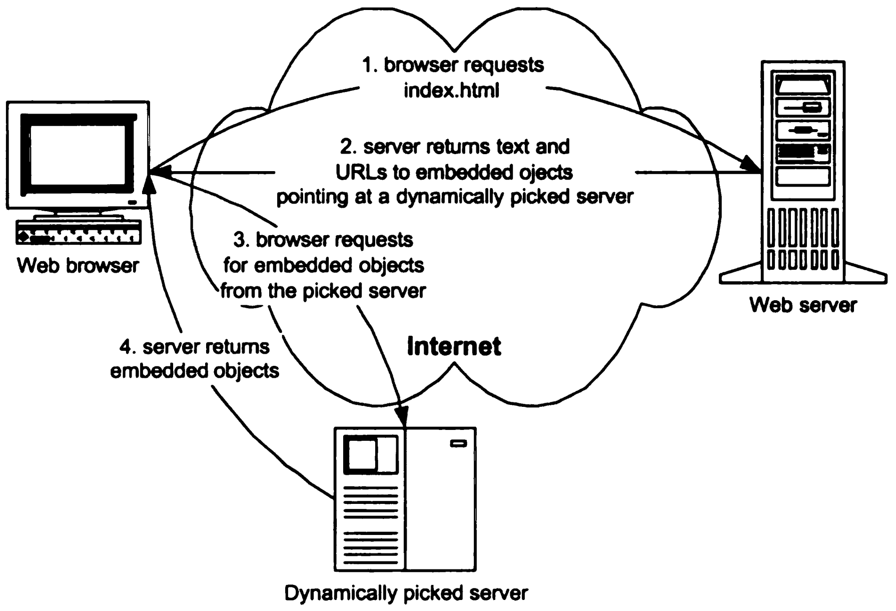

1. browser requests index.html
2. server returns text and URLs to embedded ojects pointing at a dynamically picked server
3. browser requests for embedded objects from the picked server
4. server returns embedded objects
Internet
Web browser
Web server
Dynamically picked server

Figure 6-2 Content delivery with traffic directing

Other Web caching approach is similar in nature.

Because traffic directing essentially moves content closer to the users, it makes content delivery more efficient in the Internet. Since ISPs charge the content providers such as Yahoo based on how much traffic they send and receive, ISPs that have more subscribers will benefit more from traffic directing than ISPs that have fewer subscribers (e.g., Web hosting companies).

6.1.2 The role of traffic directing in providing QoS in the Internet

Traffic directing can be considered as an application-layer approach for providing QoS in the Internet. The interactions between traffic directing and Diffserv, and traffic engineering, are investigated in this section.

6.1.2.1 Interaction between traffic directing and Diffserv

Traffic directing is an application-layer approach for providing QoS. It reduces traffic in the highly loaded networks or the highly loaded part of a network by picking the source of traffic dynamically. This may reduce the probability of network congestion, and therefore improve the perceived performance of traffic.

Diffserv is a transport/network layer approach for providing QoS. If there is any congestion in part of a network, the Diffserv mechanisms provide graceful performance degradation or no degradation for high priority traffic at the expense of low priority traffic. Therefore, traffic directing and Diffserv are mutually complementary.

6.1.2.2 Interaction between traffic directing and traffic engineering

Since traffic engineering is a network-layer approach for providing QoS, traffic directing and traffic engineering is also complementary. However, the interaction between traffic directing and traffic engineering is more complex. Traffic directing is in essence very similar to state-based constraint-based routing [8] in traffic engineering. Both traffic directing and traffic engineering try to control the distribution of traffic to avoid congestion and thus improve network performance. The main difference is that traffic directing controls traffic distribution at the application layer while traffic engineering controls traffic distribution at the network layer. Because similar controls are deployed at different layer, care must be taken that they do not interfere with each other.

Scheduling systems in traffic directing that allocate servers to in replicated, geographically dispersed information distribution systems may require performance parameters of the network to make effective decisions. It is desirable that the traffic engineering system provides such information. When there is congestion in the network, the traffic directing and traffic engineering systems should act in a coordinated manner.

Because traffic directing can introduce more traffic dynamics into a network, network planning should take this into consideration. It can be desirable to reserve a certain amount of extra capacity for the links to accommodate this additional traffic fluctuation.

6.2 Load Balancing among Servers

No matter what sophisticated mechanisms are provided in the backbone, if the data servers fail to handle the requests, QoS cannot be delivered. For example, FreeFlow directs requests for embedded objects away from the original server to the dynamically picked servers. This speeds up the delivery of content-rich Web pages. However, because the initial requests are still sent to the original server, if the original server fails, the FreeFlow approach cannot helps. Besides, although the FreeFlow approach significantly reduces the load on the original servers, for busy Web sites, the number of requests on the original server is still every high. Therefore, other approaches are needed to improve the availability of the original servers and reduce their load. In this section, we present the such an approach.

6.2.1 A load balancing approach

Traditionally, a single Web server is connected to a router that is connected to the Internet, as showed in Fig. 6-3. In order to reduce the load of the server, and improve availability of the Web site, a load balancer and multiple servers can be used. In this scenario, the servers are connected to the load balancer, which is in turn connected to the router, as showed in Fig. 6-4. The load balancer intercepts each user request, and assigns

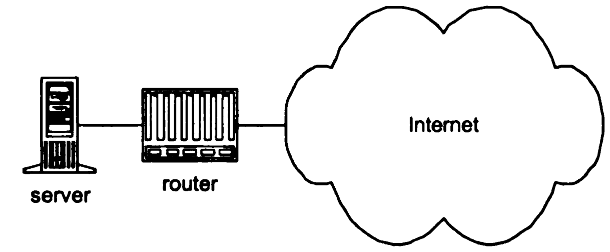


Figure 6-3 Server connection without a load balancer

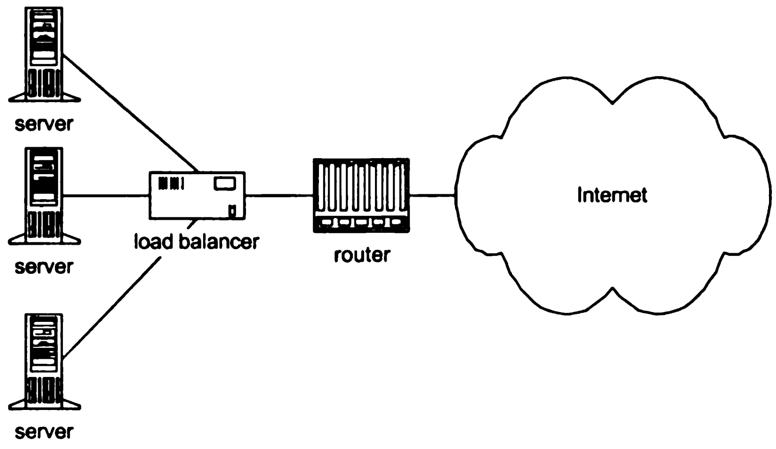


Figure 6-4 Server connection with a load balancer

it to a particular server based on some kind of criteria. The load balancing scheme can be round robin or weighted round robin based on some statistics. The load balancer also performs application-layer health check for each server to make sure that it is not only reachable but also serving user requests properly (instead of returning "404 Objects Not Found"). The load balancer can also do packet filtering, source tracing and idle connection reaping. These are useful for protecting the servers from *denial of service* (DoS) attacks.

Obviously, the load balancer can also be used to provide load balancing and protection for other application servers such as email servers.

6.2.2 Relationship between load balancing and traffic directing, Diffserv and traffic engineering

Since load balancing is for reducing the load on each particular server, and increasing the availability of the servers, it is complementary to traffic directing, Diffserv and traffic engineering. Load balancing techniques are concerned with the original servers while traffic directing, Diffserv and traffic engineering are concerned with the delivery of traffic in the backbone. No QoS solution is complete without load balancing among the servers.

Chapter 7 CONCLUSION AND FUTURE WORKS

In this dissertation we describe a framework for providing QoS in the Internet. This framework consists of multiple schemes at different layers of the network. The scheme at the transport/network layer is Diffserv. The schemes at the network layer are traffic engineering and fast reroute. We also discuss application-layer schemes such as traffic directing and load balancing.

7.1 Diffserv

Diffserv is to divide traffic into multiple classes and treat them differently, especially during time when there is a shortage of resources such as link bandwidth and buffer space. Essentially, high priority traffic is given preferable treatment during network congestion at the expense of the low priority traffic.

We propose an approach for providing differentiated services in IP-based networks. Three classes of services are provided, i.e., premium service, assured service and best effort service. In this approach,

1. Customers negotiate SLAs with ISPs. The SLAs specify what services the customers will receive. SLAs can be static or dynamic. For static SLAs,

customers can transmit data at any time. For dynamic SLAs, customers must request for the desired service before transmitting data.

2. With service allocation, a customer domain decides how its hosts and applications share the services specified in the SLAs. With resource provisioning, an ISP decides how to configure its routers so that the contracted services can be delivered.
3. Packets are classified and have their DSCPs set at the edge of a network. Subsequent forwarding treatment is solely based on the DSCPs of the packets. In order to deliver premium service and assured service, the following mechanisms are needed in ISP networks:
   - At the ingress routers: classification, policing, marking and shaping;
   - At all routers: BA classifications, RED/RIO and WFQ scheduling on PQ, AQ and DQ.

Diffserv has a conflict with TCP on the use of the precedence field in the IP header – because Diffserv-capable routers may change the precedence fields of packets en route, TCP connections may be reset undesirably for TCP stacks that strictly implement RFC793. The proposed solution to this problem is to make TCP ignore checking the precedence field.

7.2 Traffic Engineering and Fast Reroute

Traffic engineering is an iterative process of network planning and network optimization. The purpose of traffic engineering is to optimize resource efficiency and network performance. Traffic engineering is important for providing QoS in the Internet because, when there is no congestion, network performance can be very good even without Diffserv. The task of traffic engineering is to avoid/relieve congestion, and to prevent concentration of high priority traffic.

Constraint-based routing, MPLS, an enhanced link state IGP, and a path signaling protocol are useful tools for traffic engineering. Constraint-based routing computes routes that are subject to constraints such as bandwidth and administrative policy. Because constraint-based routing considers more than network topology in deciding routes, it may find a longer but lightly loaded path better than a heavily loaded shortest path. Network traffic is hence distributed more evenly. An enhanced link state IGP provides the topology information of the network and the reservable bandwidth and affinity of each link. Such information is needed by constraint-based routing to compute the paths for the MPLS LSPs. A path signaling protocol can then be used the set up the LSPs. Essentially, MPLS provides the connections in the traditionally connectionless IP networks.

Deploying an MPLS system for traffic engineering in a large ISP network is feasible. MPLS, constraint-based routing, an enhanced link state IGP, and a path signaling protocol have to be deployed in every router that participates in the MPLS system. Because everything is done by software upgrade, such a wide-range deployment turns out not to be as difficult as many people might think.

An MPLS system is very useful for traffic engineering. It automatically achieves desired traffic behavior and significantly relieves network administrators from the tedious work of manipulating routing metrics. It is especially useful in regions of irregular network topology or tight capacity.

Network planning is very important. It is through network planning that networks evolve. When there is a network problem, network optimization must provide an immediate fix. Because of the time constraint, the fix may not be the optimal solution. Network planning may subsequently be used to improve the situation. Network planning also makes a network more robust and makes network operation simpler.

MPLmS is a very important emerging technology. The essence of MPLmS is to use a similar control plane for both optical cross-connects (OXCs) and label switching routers (LSRs). More specifically, the control plane for OXCs will consist of an enhanced link state IGP, constraint-based routing and a path signaling protocol. Optical channel trails (OCTs) can then be computed and set up in a similar way as MPLS LSPs. MPLmS shifts some of the traffic engineering work from the network layer to the physical layer.

Inter-domain traffic engineering can be done quantitatively if an end-to-end traffic matrix is available. MPLS provides a new approach for inter-domain traffic engineering. Because an end-to-end traffic matrix is critical for both intra-domain and inter-domain traffic engineering, it is imperative that router vendors provide some mechanisms for constructing traffic matrices. Providing counters to record the amount of traffic from each router to all IP address prefixes is a simple and effective approach.

Fast reroute is to temporarily repair an affected path so that it can continue to carry traffic before a more optimal path can be computed by the source and used to carry traffic. Fast reroute is important for providing QoS because, when a router or a link fails, it takes IGPs from seconds to minutes to propagate information and re-compute paths. During this time interval, traffic sent along an affected path will be lost. Therefore, it is important that the affected path is temporarily repaired so that it can continue to carry traffic. This is especially important for providing QoS to real-time or other high priority traffic. Because fast reroute introduces significant complexity into a network, it may not be necessary for best effort traffic.

7.3 Traffic Directing and Load Balancing

Traffic directing (TD) is to utilize the high-performance parts of the Internet as much as possible and avoid using the low-performance parts. Internet users can be considered as either clients or servers. Traffic directing can be done by putting multiple servers with synchronized content at different places, and picking a specific server for each client dynamically based on geographic distance and/or some network performance statistics such as latency. Traffic directing and Diffserv are mutually complementary.

The interaction between traffic directing and traffic engineering can be complex. Traffic directing is essentially very similar to the constraint-based routing in traffic engineering. Both traffic directing and traffic engineering try to control the distribution of traffic to avoid congestion and thus improve network performance. The main difference is that traffic directing controls traffic distribution at the application layer while traffic engineering controls traffic distribution at the network layer. Because similar controls are deployed at different layers, care must be taken to make sure that they do not interfere with each other.

Scheduling systems in traffic directing that allocate servers to in replicated, geographically dispersed information distribution systems may require performance parameters of the network to make effective decisions. It is desirable that the TE system provides such information. When there is congestion in the network, the TD and the TE systems should act in a coordinated manner.

Load balancing is to use multiple servers (geographically close to each other) and distribute requests evenly among them. Load balancing increases service availability and reduces the load of each server. Load balancing is complementary to traffic directing, Diffserv and traffic engineering.

7.4 Final Comments and Future Works

This dissertation proposes a framework consisting of multiple schemes for providing QoS in the Internet. These schemes are only effectively when the routers are functioning properly. If a router crashes, all the mechanisms in the router are useless. Worse, traffic is dropped, and other routers have to re-compute their IGP and BGP routing tables. Therefore, increasing the reliability and availability of routers is more important for providing QoS than anything else.

The software and hardware system of a router is very complex. It is difficult, if not impossible, to avoid errors in such a complex system. Therefore, instead of trying to avoid errors, it is more practical and effective for routers to have mechanisms to minimize the impact of errors and to recover from errors gracefully. Using a modular software architecture with separated memory space for different tasks is an effective approach.

More mechanisms increase software and hardware complexity, and thus the likelihood of router failure. With bandwidth becoming cheaper and more abundant, it is advisable for ISPs to provide a certain amount of extra capacity so that complex mechanisms are not needed for providing QoS. A certain amount of extra capacity also makes network operation easier. The reduction in network operation cost can well offset the cost of extra bandwidth. However, extra capacity will not eliminate the need for Diffserv, traffic engineering, fast reroute, traffic directing and load balancing, because capacity is not always available in the place where it is needed most. Therefore, a balance needs to be made between over-provisioning and QoS mechanisms.

SLA monitoring is necessary for providing QoS in the Internet. Since ISPs charge users more for QoS, SLA monitoring is needed for verifying that the contracted service quality is actually delivered. Furthermore, detailed metering of traffic is also needed so that accounting and billing can be done. Customers can be billed based on their bandwidth usage (95th percentile) or total data (in bytes) they send and receive.

Providing QoS in the Internet is an important but difficult task. In this dissertation, we propose a systematic approach. Some schemes in this approach such as traffic engineering have been proved to be very effective. Important future works include quantitative evaluation of the effectiveness of Diffserv, fast reroute and traffic directing, and exploring effective SLA monitoring approaches.

REFERENCE:

[1] R. Comerford, "State of the Internet: Roundtable 4.0", IEEE Spectrum, Oct. 1998

[2] A. Tanenbaum, "Computer Networks", third edition, Prentice Hall PTR, 1996.

[3] B. Halabi, "Internet Routing Architectures", Cisco Press, 1994.

[4] Goralski, "SONET", McGraw Hill, 1997.

[5] X. Xiao and L. Ni, "Internet QoS: A Big Picture", IEEE Network Magazine, March/April, pp. 8-18, 1999.

[6] P. Ferguson and G. Huston, "Quality of Service", John Wiley & Sons, 1998.

[7] Juniper Whitepaper, "Optimizing Routing Software for Reliable Internet Growth", http://www.juniper.net/leadingedge/whitepapers/optimizing-routing-sw.fm.html

[8] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao, "A Framework for Internet Traffic Engineering", Internet draft, Jan. 2000.

[9] X. Xiao, A. Hannan, B. Bailey, and L. Ni, "Traffic Engineering with MPLS in the Internet", IEEE Network magazine, March 2000.

[10] X. Xiao, T. Irpan, A. Hannan, R. C. Tsay, L. M. Ni, "Traffic Engineering with MPLS", America's Network magazine, pp. 32-37, Nov. 1999.

[11] E. Rosen, A. Viswanathan and R.Callon, "Multiprotocol Label Switching Architecture", Internet draft <draft-ietf-mpls-arch-06.txt>, Aug. 1999.

[12] http://www.ietf.org/html.charters/wg-dir.html#Security_Area.

[13] B. Mukherjee, "Optical Communications Networks," McGraw-Hill, 1997.

[14] G. Newsome and P. Bonenfant, "The Automatic Switched Optical Network," Contribution to T1 STANDARDS PROJECT - T1X1.5, 1999.

[15] P. Bonenfant and and X. Liu, "A Proposal for Automatically Switched Optical Channel Networks (ASON)," Contribution to T1 STANDARDS PROJECT - T1X1.5, 1999.

[16] D. Awduche, Y. Rekhter, J. Drake, and R. Coltun, "Multi-Protocol Lambda Switching: Combining MPLS Traffic Engineering Control With Optical Crossconnects", Internet draft <draft-awduche-mpls-te-optical-01.txt>, Nov. 1999.

[17] Cisco's 12000 Series, http://www.cisco.com/warp/public/733/12000/

[18] Ascend's GRF routers, http://www.ascend.com/300.html

[19] Bay Networks' Accelar Routing Switches, http://business5.baynetworks.com/MainBody.asp

[20] 3Com's switches, http://www.3com.com/products/switches.html

[21] Juniper's M40 Series, http://www.juniper.net/products/default.htm

[22] Lucent's PacketStar 6400 Series, http://www.lucent.com:80/dns/products/ps6400.html

[23] D. Ferrari and L. Delgrossi, "Charging For QoS", IEEE/IFIP IWQOS '98 keynote paper, Napa, California, May 1998

[24] Braden, R., Clark, D. and Shenker, S., "Integrated Services in the Internet Architecture: an Overview", Internet RFC 1633, Jun. 1994

[25] R. Jain, "Myths about Congestion Management in High Speed Networks," Internetworking: Research and Experience, Volume 3, 1992, pp. 101-113.

[26] S. Shenker, C. Partridge and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, Sept. 1997

[27] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", RFC 2211, Sept. 1997

[28] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, Sept. 1997

[29] D. Ferrari, D. Verma, "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", IEEE JSAC, vol. 8, no. 3, April 1990, pp. 368-379.

[30] A. Banerjea and B. Mah, "The Real-Time Channel Administration Protocol", Proceedings of the 2nd International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'91), Springer-Verlag, Heidelberg, Germany, pp. 160-170.

[31] R. Guerin, S. Blake and S. Herzog, "Aggregating RSVP-based QoS Requests", Internet draft <draft-guerin-aggreg-RSVP-00.txt>, Nov. 1997

[32] D. Clark, "The Design Philosophy of the DARPA Internet Protocol", ACM SIGCOMM '88, Aug. 1988

[33] J. Postel, "Service Mappings," RFC 795, Sept. 1981.

[34] Almquist, P., "Type of Service in the Internet Protocol Suite", RFC 1349, July 1992.

[35] A. Parekh, "A Generialized Processor Sharing Approach to Flow Control in Integrated Services Networks", Ph.D. dissertation, MIT, Feb. 1992

[36] C. Partridge, "Gigabit Networking", Addison-Wesley, 1994

[37] H. Zhang, "Service Disciplines For Guaranteed Performance Service in Packet-Switching Networks", Proceedings of the IEEE, 83(10), Oct. 1995

[38] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, Dec. 1998.

[39] Y.Bernet et al., "A Framework for Differentiated Services", Internet draft <draft-ietf-Diffserv-framework-00.txt>, May 1998

[40] K. Nichols, V. Jacobson and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", Internet Draft, <draft-nichols-diff-svc-arch-00.txt>, Nov. 1997.

[41] K. Nichols et al., "Differentiated Services Operational Model and Definitions", Internet draft <draft-nichols-dsopdef-00.txt>, Feb. 1998

[42] D. Clark and J. Wroclawski, "An Approach to Service Allocation in the Internet", Internet draft <draft-clark-different-svc-alloc-00.txt>, Jul. 1997

[43] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, Dec. 1998.

[44] Heinanen, J., Baker, F., Weiss, W. and J. Wroclawski, "Assured Forwarding PHB Group", RFC 2597, June 1999.

[45] Jacobson, V., Nichols, K. and K. Poduri, "An Expedited Forwarding PHB", RFC 2598, June 1999.

[46] Y.Bernet et al., "A Framework for use of RSVP with Diff-serv Networks", Internet draft <draft-ietf-Diffserv-rsvp-00.txt>, Jun. 1998

[47] http://www.akamai.com

[48] http://www.inktomi.com

[49] http://www.digislc.net

[50] http://www.f5.com

[51] R. Gougen, G. Swallow, "RSVP Label Allocation for Backup Tunnels", Internet draft, <draft-swallow-rsvp-bypass-label-00.txt>, Oct. 1999.

[52] D. Haskin, R. Krishnan, "A Method for Setting an Alternative Label Switched Paths to Handle Fast Reroute", Internet draft, <draft-haskin-mpls-fast-reroute-03.txt>, Mar. 2000.

[53] L. Andersson, B. Cain, B. Jamoussi, "Requirement Framework for Fast Re-route with MPLS", Internet draft, <draft-andersson-reroute-frmwrk-00.txt>, Nov. 1999.

[54] B. Braden et al., "Recommendation on Queue Management and Congestion Avoidance in the Internet", RFC 2309, Apr. 1998

[55] W. Yeong, T. Howes, and S. Kille, "Lightweight Directory Access Protocol", RFC 1777, Mar. 1995.

[56] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, Sept. 1981.

[57] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[58] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[59] X. Xiao, E. Crabbe, A. Hannan and Vern Paxson, "TCP Processing of the IP Precedence Field", Internet Draft <draft-xiao-tcp-prec-01.txt>, Jan. 2000.

[60] R. Stevens, "TCP/IP Illustrated", Volume 1, Addison Wesley, 1994.

[61] L. Andersson, P. Doolan, N. Feldman, A. Fredette and R. Thomas, "Label Distribution Protocol", Internet draft <draft-ietf-mpls-ldp-02.txt>, Nov. 1998

[62] P. Vaananen and R. Ravikanth, "Framework for Traffic Management in MPLS Networks", Internet draft <draft-vaananen-mpls-tm-framework-00.txt>, Mar. 1998

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

[64] C. Villamizar, "OSPF Optimized Multipath (OSPF-OMP)", Internet draft, <draft-ietf-ospf-omp-02.txt>, Feb. 2000.

[65] D. Awduche, L. Berger, D. Gan, T. Li, G. Swallow and V. Srinivasan, "Extension to RSVP for LSP Tunnels", Internet draft <draft-ietf-mpls-rsvp-lsp-tunnel-05.txt>, Feb. 2000.

[66] D. Awduche, A. Hannan and X. Xiao, "Applicability Statement for Extensions to RSVP for LSP-LSPs," Internet Draft <draft-ietf-mpls-rsvp-tunnel-applicability-01.txt>, Sept. 1999.

[67] L. Andersson, P. Doolan, N. Feldman, A. Fredette, B. Thomas, "LDP Specification", Internet draft, <draft-ietf-mpls-ldp-06.txt>, Oct. 1999.

[68] B. Jamoussi et. al., "Constraint-Based LSP Setup using LDP", Internet draft, <draft-ietf-mpls-cr-ldp-03.txt>, Sept. 1999.

[69] T. Li, G. Swallow and D. Awduche, "IGP Requirements for Traffic Engineering with MPLS", Internet draft <draft-li-mpls-igp-te-00.txt>, Feb. 1999.

[70] H. Smit and T. Li, "IS-IS extensions for Traffic Engineering", Internet draft, <draft-ietf-isis-traffic-01.txt>, May 1999.

[71] R. Coltun, "The OSPF Opaque LSA Option", RFC 2370, Jul. 1998.

[72] N. Shen and H. Smit, "Calculating IGP routes over Traffic Engineering LSPs", Internet draft <draft-hsmit-mpls-igp-spf-00.txt>, June 1999.

[73] Y. Rekhter, T. Li, “A Border Gateway Protocol 4 (BGP-4)”, RFC 1771, Mar. 1995.

[74] E. Rosen and Y. Rekhter, “BGP/MPLS VPN”, RFC 2547, March 1999.

[75] Wide Area Network Design Laboratory (WANDL), http://www.wandl.com.

[76] M. Garey and D. Johnson, “Computer and Intractability”, W. H. Freeman and Company, 1979.

[77] T. Li and Y. Rekhter, "Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)", RFC 2430, Oct. 1998

[78] T. Li, "CPE based VPNs using MPLS", Internet draft <draft-li-MPLS-vpn-00.txt>, Oct. 1998

[79] M. Waldvoge, G. Varghese, J. Turner and B. Plattner, "Scalable High Speed IP Routing Lookups", ACM SIGCOMM ’97, Cannes, France, Sept. 1997.

[80] S. Nilsson and G. Karlsson, "Fast Address Lookup for Internet Routers", ACM SIGCOMM ’97, Cannes, France, Sept. 1997.

[81] J. Moy, “OSPF Version 2”, RFC 2178, Apr. 1998.

[82] J. Moy, "OSPF", Addison and Wesley, 1998.

[83] E. Crawley, R. Nair, B. Jajagopalan and H. Sandick, "A Framework for QoS-based Routing in the Internet", RFC 2386, Aug. 1998.

[84] G. Apostolopoulos, R. Guerin, S. Kamat and S. Tripathi, "Quality of Service Based Routing: A Performance Perspective", ACM SIGCOMM '98, pp. 17-28, Vancouver, Canada, Aug. 1998.

[85] Q. Ma, "QoS Routing in the Integrated Services networks", Ph.D. dissertation, CMU-CS-98-138, Jan. 1998.

[86] Z. Wang, "Routing and Congestion Control in Datagram Networks", Ph.D. dissertation, Dept. of Computer Sci., University College London, Jan. 1992.

[87] R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams, "QoS Routing Mechanisms and OSPF extensions", Internet draft <draft-guerin-QoS-routing-ospf-03.txt>, Jan. 1998.

[88] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley, "QoS Extensions to OSPF", Internet draft <draft-zhang-qps-ospf-01>, Sept. 1997.

[89] Z. Wang and J. Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications", IEEE JSAC, Sept. 1996.

[90] S. Chen and K. Nahrstedt, "An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions", IEEE Network Magazine, Special Issue on Transmission and Distribution of Digital Video, 1998.

[91] S. Chen and K. Nahrstedt, "Distributed Quality-of-Service Routing in Ad-Hoc Networks", IEEE Journal on Special Areas in Communications, Vol. 17, No. 8, Aug. 1999.

[92] A. Orda, "Routing with End-to-End QoS Guarantees in Broadband Networks", Technical Report, Technion, I.I.T., Israel.

[93] Y. Goto, M. Ohta and K. Araki, "Path QoS Collection for Stable Hop-by-hop QoS Routing", Proceedings of INET '97, Kuala Lumpur, Malaysia, Jun. 1997.

[94] A. Girard, "Routing and Dimensioning in Circuit-switched Networks", Addison-Wesley, 1990.

[95] A. Girard and B. Liau, "Dimensioning of Adaptively Routed Networks", IEEE/ACM Transactions on Networking, vol. 1, pp. 460-468, August 1993.

[96] A. Girard, "Revenue Optimization of Telecommunication Networks", IEEE Tans. Comm., vol. 41, pp. 583-591, April 1993.

[97] O. Dioume, A. Girard and F. Soumis, "Dimensioning Telephone Networks with Budget Constraints", Telecommunication Systems, vol. 1, pp. 149-178, December 1992.

[98] R.J. Gibbens, F.P. Kelly, and P.B. Key, "Dynamic Alternative Routing", Routing in Communication Networks (Editor Martha Steenstrup), pp. 13-47, Prentice Hall, Englewood Cliffs, 1995.

[99] F.P. Kelly, "Dynamic Routing in Stochastic Networks", Stochastic Networks (Editors F.P. Kelly and R.J. Williams), The IMA Volumes in Mathematics and its Applications, vol. 71, pp. 169-186, Springer-Verlag, New York, 1995.

[100] F.P. Kelly, "Mathematical Models of Multiservice Networks", Complex Stochastic Systems and Engineering (Editor D.M. Titterington), pp. 221-234, Oxford University Press 1995.

[101] G. Louth, M. Mitzenmacher and F. Kelly, "Computational Complexity of Loss Networks", Theoretical Computer Science vol. 125, pp. 45-59, 1994.

[102] F. Kelly, "Bounds on the Performance of Dynamic Routing for Highly Connected Networks", Mathematics of Operations Research vol. 19, pp. 1-20, 1994.

[103] F. Kelly, "On Tariffs, Policing and Admission Control for Multiservice Networks", Operations Research Letters, vol. 15, pp. 1-9, 1994.

[104] R.J. Gibbens, F.P. Kelly, and S.R.E. Turner, "Dynamic Routing in Multiparented Networks", IEEE/ACM Transactions on Networking, vol. 1, pp. 261-270, 1993.

[105] F.P. Kelly and C.N. Laws, “Dynamic Routing in Open Queueing Networks: Brownian Models, Cut Constraints and Resource Pooling”, Queueing Systems, vol. 13, pp. 47-86, 1993.

[106] F.P. Kelly, “Network Routing”, Philosophical Transactions of the Royal Society, vol. A337, pp. 343-367, 1991.

[107] F. Kelly, “Notes on Effective Bandwidths”, in "Stochastic Networks: Theory and Applications" (Editors: F. Kelly, S. Zachary and I.B. Ziedins), pp. 141-168, Oxford University Press, 1996.

[108] F. Kelly, "Modelling Communication Networks, Present and Future", Philosophical Transactions of the Royal Society A354, pp. 437-463, 1996.

[109] G. R. Ash, J. S. Chen, A. E. Frey and B. D. Huang, "RealTime Network Routing in a Dynamic Class-of-Service Network", Proceedings of ITC 13, Copenhagen, Jun. 1991.

[110] ATM Forum PNNI subworking group, "Private Network-Network Interface Spec. v1.0 (PNNI 1.0)", afpnni-0055.00, Mar. 1996.

[111] X. Xiao and Lionel Ni, "Reducing Routing Table Computation Cost in OSPF", Proceedings of INET '99, San Jose, June 22-25, 1999. A short version of this paper also appears in the Proceedings of the Internet Workshop '99, Osaka, Japan, Feb. 1999.

[112] X. Xiao and Lionel Ni, "Parallel Routing Table Computation for Scalable IP Routers", Proceedings of the IEEE International Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing (Published

by Springer as Lecture Notes in Computer Science 1362), pp. 144-158, Las Vegas, Feb. 1998

[113] X. Xiao, L. Ni and V. Vuppala, "A Comprehensive Comparison of IP Switching and Tag Switching", Proceedings of IEEE Intl. Conf. on Parallel and Distributed Systems (ICPADS '97), pp. 669-675, Seoul, Dec. 1997

[114] X. Xiao, W. Tang, "Benchmarking the TCP/UDP performance of Myrinet over Linux", Technical report, MSU-CPS-ACS-1196.

[115] Network maps, http://web.mit.edu/afs/net.mit.edu/contrib/maps/network

[116] K. Rosen, "Discrete Mathematics and its Applications (4th Edition)", McGraw Hill, 1998.

[117] D. Knuth, "The Art of Computer Programming, Volume 1: Fundamental Algorithms", 3rd Edition, Addison-Wesley, 1997.

[118] T. Cormen, "Introduction to Algorithms", MIT Press, 1990.

[119] Multi-threaded Routing Toolkit, http://www.merit.edu/research.and.development/mrt/html

[120] Internet Performance Measurement and Analysis Project (IPMA), http://www.merit.edu/ipma

[121] K. Ishiguro, Zebra Project, ftp://ftp.zebra.org/pub/zebra

[122] http://www.gated.org/

[123] G. Chartrand, O. Oellermann, "Applied and Algorithmic Graph Theory", McGraw Hill, 1993.

[124] S. Ross, "Introduction to Probability Models Sixth Edition", Academic Press, 1997.

[125] R. Jain, "The Art of Computer Systems Performance Analysis", John Wiley & Sons, 1991.

[126] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", RFC2616, Jun. 1999.

[127] T. Berners-Lee, D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, Nov. 1995.