Dynamic Service Provisioning in IP Networks

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Ibrahim Khalil

aus Bangladesh

Leiter der Arbeit:

Prof. Dr. T. Braun

Institut für Informatik und angewandte Mathematik

Dynamic Service Provisioning in IP Networks

Inauguraldissertation
der Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

vorgelegt von

Ibrahim Khalil

aus Bangladesh

Leiter der Arbeit:

Prof. Dr. T. Braun

Institut für Informatik und angewandte Mathematik

Von der Philosophisch-naturwissenschaftlichen Fakultät angenommen.

Der Dekan:

Bern,7. Februar 2003 Prof. Dr. Gerhard Jäger

Preface

This dissertation is the result of nearly three years of work during August 1998 - March 2001 at Computer Networks and Distributed Systems group of Institute of Computer Science and Applied Mathematics at University of Berne. It is time to thank all my colleagues that provided me with their invaluable feedback and support. This thesis would not have been possible without the help of many people.

I would like to thank my sole advisor Prof. Dr. Torsten Braun whose vision and enthusiasm has had a great impact on my scientific development and the research work presented in this thesis. He formulated the initial idea of QoS and VPN management in the CATI project and hired me as a junior researcher to work on this excellent topic. The prototypes presented in this thesis work would not have been possible without his generous expenditure on network lab equipment to help me implement them and understand the research issues better. Professor Braun was instrumental in shaping the contents of this thesis by encouraging me to publish in journals and conferences, and making numerous corrections to submitted and accepted publications. Without his constant encouragement during my difficult times this thesis would not have been possible. In many ways he not only helped shape my published research work, but also future research direction and my professional development. He is truly a great advisor and I am truly grateful to him.

I would also like to thank Prof. Dr. Burkhard Stiller of ETH Zurich who was the leader of the CATI project and often gave helpful suggestions and insights during the project meetings. Many thanks to Prof. Dr. Hanspeter Bieri for accepting to be a co-examiner and evaluate the dissertation work.

I also greatly appreciate the large academic freedom, the working environment, and the very positive atmosphere at RVS,IAM. I am particularly indebted to my RVS colleagues , Manuel Günter, Roland Balmer, Florian Bramgartner, Linquing Liu who worked diligently and efficiently providing laboratory support in my implementations. Manuel Günter formulated the initial Qos VPN architecture in the CATI project based on which I primarily implemented the Service Broker. He provided and shared ideas and is the co-author of few of our papers that are part of this thesis.

My three months stay at TBD Networks, California was immensely helpful in

defining configuration modeling techniques and use them in intelligent VPN service provisioning and configuration audit agents. Many thanks to the Dr. Anindo Banerjea, CTO of the company for giving me the opportunity to work there and practice what I learned from research at RVS. With his help, and help from Norbert Kiesel, Director of Software Engineering, I was able to write intelligent device driver for provisioning Nortel Contivity Switches. That was a good proof of concept of the work presented in chapter 7.

Dr. David Billard and Noria Foukia of University of Geneva were generous to help in the experimental QoS-VPN setup and allowed me to use the group's lab facilities. I very much appreciate their help.

Ruth Bestgen, secretary of Prof. Braun, provided the best administrative support that one could ask for. While I was away from Switzerland for almost two years and working at PONTE communications,CA she kept doing all my official Phd registrations and arranged numerous immigration documents for me to come to Switzerland. Words are not enough to express my gratitude for her personal support.

The works presented in this thesis were mostly done in the projects *Charging and Accounting Technologies for the Internet* (CATI) and *Advanced Network and Agent Infrastructure for the Support Of Federations of Workflow Trading Systems* (ANAISOFT) funded by the Swiss National Science Foundation (SNF).

Finally, I would like to thank my wife Sharmeen for her endless patience and support throughout all these years; and also my parents that motivated me to pursue a doctoral degree. My only son Ismail, who was born during my research work, was a source of joy and great motivating factor when I was tired and lost.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet's global presence, flat fees, distance independent rates makes it attractive as a universal communication infrastructure for businesses. However, the astonishing growth of Internet in recent years has been characterized by the struggle of Internet Service Providers (ISPs) and network managers to keep pace with the demand of network resources. As the Internet evolves into a global commercial infrastructure, there is a growing need to support more enhanced services than the traditional best-effort service. It is deemed that the Internet will be used much more often for mission critical communications with enhanced security, and multimedia services requiring high bandwidth. There are two key issues that businesses consider when evaluating the use of the Internet for any application:

- *Quality of Service (QoS)*: Originally developed for academic and public use, the Internet was designed to be simple, flexible, scalable and efficient. It was modeled after the design philosophy where the source and destination end-hosts perform most of the reliability mechanisms as well as congestion handling, and the network itself performs minimal operations maintaining little state. While these attributes contributed to IP's explosive growth, as businesses increasingly turn to the Internet for their communication needs, a number of limitations have been exposed in the best effort service offered by IP, QoS being one of them. New applications such as voice over IP (VoIP), video streaming, conference calling, and other multimedia applications heavily depend on QoS and service differentiation mechanisms provided by the underlying network.

- *Security*: The Internet lacks of built-in security support. IP traffic is easy to eavesdrop and IP packets can easily be manipulated e.g. to appear coming from an arbitrary source address. Further, the best-effort nature of the Internet invites denial-of-service attacks based on excessive production of traffic. Such security threats fuel the trend to protect corporate network traffic on the Internet.

Fortunately, solutions exist to address both issues. A widely used approach to tackle the security problem is a Virtual Private Network (VPN). An IP VPN is commonly defined as a routed link between two or more points across a heterogeneous network topology with various degrees of security that ensure privacy for all parties. It is a private network on top of a public network infrastructure like the Internet and uses tunneling technology to make private IP networks public IP-compatible. By using encryption data passed through the tunnel can be encrypted. When tunneling occurs the source end encapsulates packets in IP packets for transit across the Internet. The encapsulation process involves adding a standard IP header to the original packet, which is then referred to as the payload. A corresponding process at the destination end decapsulates the IP packet by removing the IP header, leaving the payload - the original packet - intact.

As a solution to the QoS problem the Internet Engineering Task Force (IETF) has recently developed Differentiated Services (DiffServ) technology. With DiffServ, traffic entering a network is classified, possibly conditioned at the boundaries of the network, and assigned to different behavior aggregates. Each behavior aggregate is identified by a single DS codepoint (DSCP). In the interior of the network the aggregated traffic can be allocated a certain amount of node resources.

A marriage between QoS and VPN gives us QoS enabled IP VPN (QoS-VPN), and is considered as a fruitful solution for corporate communication. The abilities of QoS enabled VPNs to emulate a private wide area network using IP facilities and guarantee bandwidth and latency have recently generated tremendous interest in its wide spread deployment to replace the expensive dedicated private leased lines.

However, the complexities introduced by VPNs and the requirement to provide QoS have made the job of the ISPs and systems administrators extremely difficult, and as today's network infrastructure continues to grow, the ability to manage increasing complexity is a crucial factor for VPN solutions. But, at the same time, this also opens the possibility for ISPs to sell VPN services to mostly corporate end users.

Because of the complexities enterprise customers having several sites and large number of devices are willing to outsource their VPNs and other network service management to ISPs. There are economic reasons too. Qualified IT staff are at a premium; most customers want the provider to assume any management headaches, and turn their in-house people on more projects. Besides, many businesses do not have IT staff already trained in high-end-security/VPN technology. Also, many of the Customer Premises Equipments (CPE) are not VPN capable. Service providers have a new way to provide IP VPNs to the end user: network-based IP VPNs delivered from the provider's network edge to the customer premises over ISP networks. What makes this an excellent idea is that it enables telecommunication companies to house all the intelligence and VPN functionality in their network without setting up and managing expensive CPE on-site. Such network-based VPNs are supposed to eliminate truck rolls for maintenance and the hassle of deploying new equipment every time a new user is added to that network. For QoS of VPNs customers are forced to turn to ISPs because they (i.e. customers) have no control over ISP networks. There are several scenarios where customers would like to outsource services.

As providers take responsibilities of VPNs and other network resource management and gain economies of scale, they are facing new challenges. This thesis identifies and addresses these challenges and proposes novel solutions to ease network management for ISPs or large enterprises. In section 1.1 we give an overview of the possible scenarios where outsourcing of VPN and QoS management is feasible. Section 1.2 identifies and summarizes the key VPN and QoS management challenges ISPs or large enterprises face today. In section 1.3 we present the thesis organization and section 1.4 briefly discusses the main contributions of this thesis.

1.1 QoS enabled VPN Scenario

Many service providers have deployed advanced logical and physical networking to address three emerging VPN applications:

- Remote access VPNs for mobile workers and telecommuters. With this kind of VPN a single remote computer connects to the corporate intranet.
- Intranet VPNs for intra-company transactions. Several branch offices of the same company at remote locations are connected using such VPNs.

This type of VPN can employ several service platforms - secure internet, Differentiated Services, Frame Relay/ATM/MPLS backbone - on a site-by-site basis depending on the needs of each office.

- Extranet VPNs for supply-chain management. This type of VPN allows one corporation to open their network resources to other companies, mainly for business transactions.

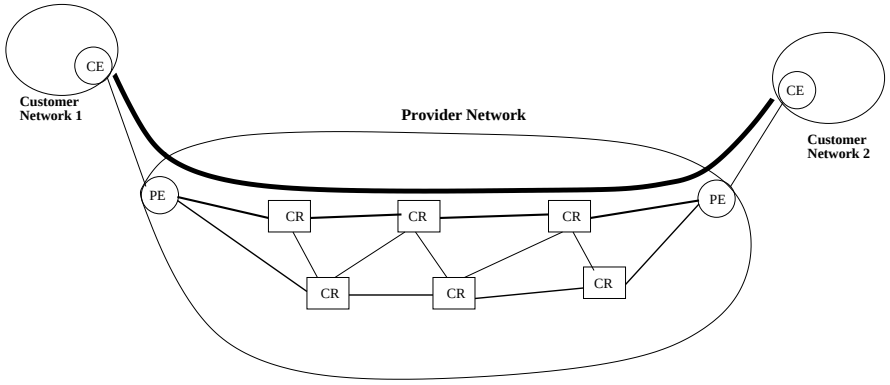


Figure 1.1: CE-based VPN

In this thesis, we mainly focus on Site-to-Site Intranet VPNs which stand as the only candidate to replace Frame Relay or ATM Permanent Virtual Circuit (PVC) based dedicated leased lines connecting two branch offices of the same company at two different locations. However, many of the ideas and solutions presented in this work apply to Extranet VPNs as well although there are differences between the two types. While Extranet VPNs span over multiple trust domains, all the site-to-site VPNs of a corporation are either managed by the company or usually outsourced to a single ISP. Intranet VPNs can either originate from the customer premise, in which case we call it a Customer Edge (CE) based VPN, or, from the provider's network boundary, falling in the category of Provider Edge (PE) based VPNs. In this dissertation we do not make much distinction between these two types and assume that in both cases services are outsourced to the ISPs.

1.2 Challenges of QoS VPN Management

Although IP VPNs have promised unlimited opportunities for service providers for some time, until now it has been almost impossible to deploy an IP VPN

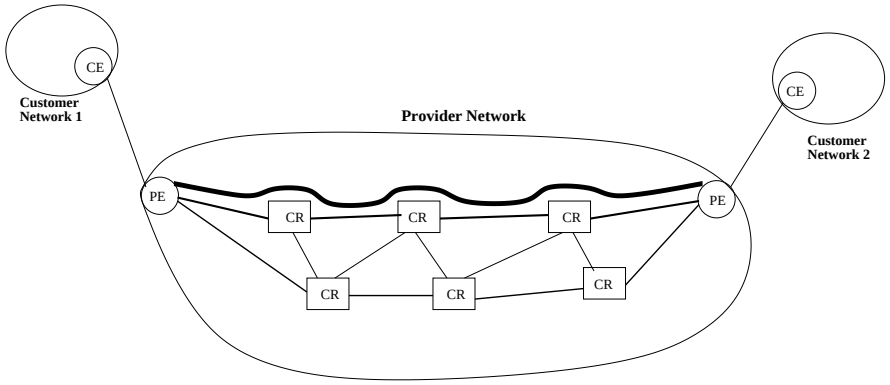


Figure 1.2: PE-based VPN

service that meets customers' needs in a timely manner. Solutions have been complex, requiring manual configuration of each network device via data entry into the command line interface (CLI). As service providers face numerous management challenges in implementing and managing IP VPNs for a diverse and rapidly growing customer base, network control and provisioning are considered to be the most daunting part in a growing communication infrastructure. Policy-based network control systems are a new way to tackle this subject. Today, policies related to network security and QoS have the largest impact on simplifying network control. Following is a list challenging features that a QoS-VPN management system is expected to offer to make it easily deployable, highly usable and economical:

- *Automated Order and SLA Management*: Management of end-to-end SLA and quality of service (QoS) in a shared and layered infrastructure environment and also resolving the challenges in delivering and assuring competitive, timely IP VPN-based service offerings.
- *Automated Service Activation*: Automatic activation of IP VPN devices and provisioning services from a centralized data store. This also includes customer-based management of their own VPNs.
- *Simplified Management in Mutivendor Environment*: Simplified management of complex VPNs from multiple vendors that use diverse network access technologies and configuration methods.
- *QoS Integration*: Integration with Quality of Service (QoS) management to deliver differentiated service offerings with VPNs.

- *Service Assurance*: Providing high-level, service-oriented views of the network, allowing service providers to quickly monitor, respond to network health issues that affect customer services. Predictable and flexible service delivery in large scale networks is extremely challenging to ISPs.
- *Billing Mediation*: Providing unified billing, immediately realizing profit potential of deployed differentiated services.
- *Providing Services in Multi-ISP Scenario*: Ability to collaborate with other service providers to offer end-to-end services.

1.3 Overview of the Thesis

The current chapter of the thesis introduces the challenges of QoS-VPN and in rest of thesis we propose new solution techniques to meet those challenges. The thesis is organized as follows:

- **Chapter 2**: Chapter 2 gives an overview of QoS and VPN enabling technologies like IPSec, DiffServ, traffic engineering methods etc.
- **Chapter 3**: A policy-based Service Broker (SB) architecture [BGK01], [GBK99] is presented in this chapter. We propose QoS and security policy templates to generate device specific QoS-VPN configurations and automate service activation process by delivering generated configurations using agents called configuration daemons.

  In chapter 3, we also describe the implementation of a Service Broker [KBG00], [KB00a], [BGK01] managing QoS-VPNs for customers that have Service Level Agreements (SLAs) with their ISPs and allowing one such user to specify demands through a WWW interface to establish a VPN with certain QoS levels between two endpoints. In this chapter we not only show the implementation details of various components of the Service Broker and connection admission and termination process, but also the pricing mechanism of such outsourced VPNs.
- **Chapter 4**: In this chapter we propose a new range-based SLA [KB02b], [KB00b] mechanism that allows customers to specify their QoS requirements as a range of quantitative parameters in the service requests since they will be unable or unwilling to predict the load between VPN endpoints. To

realize this advanced SLA we present edge provisioning algorithms that can logically partition the capacity at the edges to various classes (or groups) of VPN connections and manage them efficiently to allow resource sharing among the groups in a dynamic and fair manner.

- **Chapter 5**: The work presented in this chapter is on the application of range-based SLAs in the core network. Dynamic and frequent configurations of an interior device driven by edge bandwidth modifications is not desired as this will lead to scalability problems and also defeats the purpose of the DiffServ architecture which suggests to drive all the complexities towards edges. Therefore, in chapter 5, we propose virtual core provisioning [KB01a] that exploits range-based SLA to have higher multiplexing gain, eliminates the need to provision the physical core devices in real-time and requires only a capacity inventory of interior devices to be updated based on VPN connection acceptance, termination or modification at the edges.

- **Chapter 6**: Service Broker implementations presented in chapter 3,4 and 5 consider offering automated service activation in a single provider domain. In this chapter, we describe the implementation [KB01b] of a BB that uses simple signaling mechanism to communicate with other cooperative Brokers to enable customers to dynamically create Virtual Leased Lines (VLLs) over multiple Diffserv domains.

- **Chapter 7**: A simple push-based model to configure network services may lead to configuration inconsistencies and inefficiencies. In this chapter, we have taken a new approach [KB02a]to configuration modeling that is device neutral and based on which any existing or emerging IP services can be presented by encapsulating service semantics, including service-specific data. We have developed new mobile intelligent provisioning and audit agent architectures that use the knowledge built upon configuration dependency modeling. Examples of intelligent agents and a prototype system are presented to complement the proposed management architecture.

- **Chapter 8**: This chapter concludes the thesis by summarizing our works and identifying future research directions.

1.4 Contribution of the Thesis

Policy-based management of QoS-VPN in large networks are considered to be the most challenging aspects of modern IP networks. Throughout the thesis we have explored these challenges and injected new ideas in that area. We summarize our contributions in the following areas:

- *Policy Based Service Broker Architecture*: The idea of Policy Based Networking (PBN) or the concept of Bandwidth Broker (or Policy Server) architecture is nothing new. Originally Bandwidth Broker architecture to provision DiffServ like QoS was proposed in [NJZ99] and [Sch98]. The IETF proposed a PBN architecture [Gro], [YDP00] to configure and manage network wide policies. We have adapted and enhanced the existing ideas to make it make it work in the present context - i.e. to dynamically manage and provision QoS-VPNs in large networks.

  We introduced the idea of QoS and IPSec Policy templates to translate user service requirements and high level policies into low level device specific configurations. One useful feature of the architecture is that it allows to add new policy templates in the system as new device types are added to the network. This is extremely useful in a growing network where the ability to operate in multi-vendor environments is considered crucial.

- *Implementation of Service Brokers*: We have implemented a Service Broker (SB) system that allows not only network administrators but also corporate customers to customize and activate QoS-VPNs dynamically on the fly via a web-based front end. We have also extended the implementation to create VLLs over multiple DiffServ domains.

- *Differential Tunnel Pricing*: We have introduced a new differential tunnel pricing mechanism by which pricing of a QoS-VPN tunnel is computed based on its network resource reservation and the load during the time tunnel is active.

- *Range-Based SLA*: The novel range-based SLA mechanism proposed in this thesis allows customers to specify bandwidth demands as a range of quantitative values rather than a single one. This contribution offers several advantages:

  - users are usually reluctant to predict load between VPN endpoints. Range-based SLAs give the flexibility to specify bandwidth requirement as a range.
  - using range-based SLA customers may enjoy the upper-bound rate (say, 1 Mbps when a range 0.5-1 Mbps is chosen) during low load period without paying anything extra. This kind of pricing might be attractive to the users and the ISPs can take advantage of this to attract more customers without breaking the commitment.
  - with range-based SLAs providers have the flexibility to allocate bandwidth that falls between a lower and upper bound of the range only, and therefore, exploit this to make multiplexing gain in the core that is usually not possible with a deterministic approach.

- *Virtual Core Provisioning*: With virtual core provisioning proposed in this thesis network devices at the core do not need to be frequently configured for end-to-end QoS guarantees. A new VPN connection is subject to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration of edge devices, but only resource state updates of core routers interfaces in the Service Broker database.

- *Scalable Signaling Between Service Brokers*: Rather than using RSVP in inter-domain signaling to reserve capacity across domains, we use a novel method to identify DiffServ domains, and hence the Service Brokers that are responsible for maintaining them. Identified brokers in a chain are signaled and positive responses from all the brokers leads to end-to-end VLL setup. This greatly simplifies bandwidth reservation as it does not require sender and receiver to exist during reservation process.

- *Configuration Modeling and Intelligent Agents*: A new configuration modeling technique was proposed in this thesis that solves many of the problems of traditional configuration mechanisms which ignore dependencies among configuration elements.

- *Automated Network Device Auditing*: We developed an automated device configuration audit technique to be applied in a large scale network environment that is considered to be an extremely useful feature by network managers, but rarely addressed by researchers. We have shown how intelligent audit agents can be periodically sent to the network devices to check

the consistency of network device configuration, and also automatically fix the problem when one is found. The proposed audit management ensures that the configuration state in the policy repository is equivalent to the actual device state.

Chapter 2

Overview of QoS-VPN Technologies

The ability of VPNs to emulate a private wide area network using IP facilities has recently generated tremendous interest in its wide spread deployment and replace the expensive dedicated private leased lines. As extensions of the enterprise network a VPN solution must guarantee reliability and Quality of Service by enabling users to define enterprise-wide traffic management policies that actively allocate bandwidth for in-bound and out-bound traffic based on relative merit or importance to all other managed traffic. In this chapter, we give an overview of QoS and VPN enabling technologies like Differentiated Services, IPSec, and numerous traffic engineering methods such as ATM, MPLS, constraint-based routing etc. We also briefly describe policy-based management systems and various management architectures and protocols that are commonly used to manage the QoS-VPN enabling technologies.

2.1 IPSec-Based Data Transport Security

The most important feature of a Virtual Private Network service is its privacy ensuring mechanism, that protects VPN traffic from the underlying public network. For Internet VPNs the most powerful mechanisms are tunneling and encryption. Tunneling (also called encapsulation) is a method of wrapping a packet in a new one thus providing it with a new header. The whole original packet becomes the payload of the new one as shown in Figure 2.1 and 2.2. When encryption is

applied to a packet, it conceals the content of that packet. Tunneling requires intermediate processing of the original packet on its route. The destination specified in the outer header retrieves the original packet and sends it to the ultimate destination. Although the encapsulation and encryption may degrade the performance to some extent, the processing overhead is compensated by extra security. In general, the following features are provided by a VPN tunnel.

- *Data confidentiality* - The sender can conceal clear-text by encrypting them before transmitting across a network.
- *Data integrity* - The receiver can verify that the data has not been altered during transmission, either deliberately or due to random errors.
- *Data Origin Authentication* - The receiver can authenticate that the data was originated from the sender. This service, however, is dependent upon the data integrity service.

Security requirements of one user may vary from others under various circumstances depending on his needs. To facilitate flexibility various IPSec [KA98] tunneling options exist to allow a customer to choose the one that suits him best. They are described below:

- **IPSec AH in Tunnel Mode**: The Authentication Header (AH) is used to provide integrity and authentication to IP datagrams. When packets go through an AH type tunnel an AH is embedded (Figure 2.1(b)) in the data to be protected (a full IP datagram). This mode of operation greatly reduces the chances of successful denial of service attacks, which aim to block the communication of a host or gateway by flooding it with bogus packets. The AH protocol allows for the use of various authentication algorithms, most notably, MD5 (a hash message authentication code - HMAC variant) and SHA (HMAC variant). Both MD5 and SHA are widely used. HMAC is a keyed hash variant used to authenticate data.
- **IPSec ESP in Tunnel Mode**: The Encapsulation Security Payload (ESP) is used to provide integrity check, authentication and encryption to IP datagrams (Figure 2.2(a)). Although both authentication and encryption are optional, at least one of them is always selected. If both of them are selected, then the receiver first authenticates the packet and only if this step was successful proceeds with decryption. This mode of operation reduces the vulnerability to denial of service attacks.

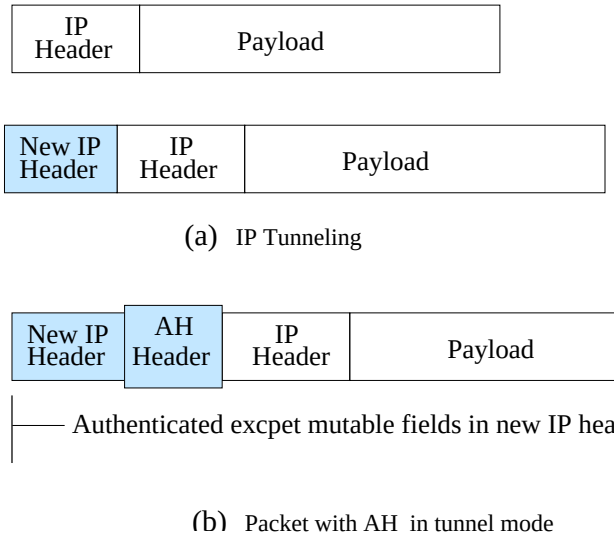


Figure 2.1: IP Tunneling: (a)Basic IP Tunneling, (b) AH Tunneled Packet

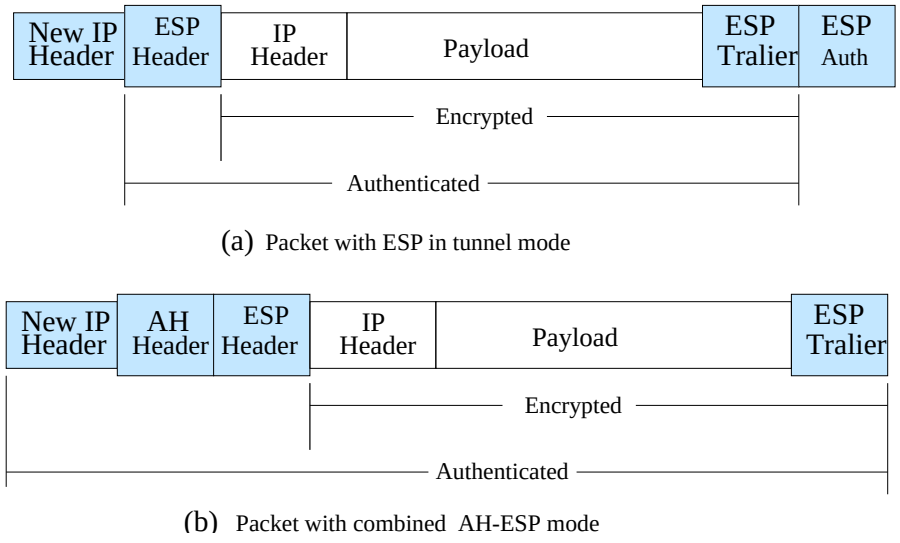


Figure 2.2: IP Tunneling: (a) ESP Tunneled Packet, (b) Combined Tunnel Mode

- **Combined Tunnel Mode**: Even though most tunnel gateways are required to support only an AH tunnel or ESP tunnel, sometimes it is desirable to have tunnels between gateways that combine both IPSec protocols. The result is that we have an outer IP header followed by the IPSec headers in the order set by the tunnel policy, then the original IP packet, as it is shown in Figure 2.2(b).

2.2 Quality of Service Support for VPNs

2.2.1 Differentiated Services (DiffServ)

Differentiated Services [BBC+98] can provide quality-of-service to VPN traffic by traffic aggregation based on Differentiated Services Code Points (DSCP). In DiffServ, the first six bits of the the 8-bit Type of Service (ToS, or DS) field in the IPv4 header comprise the DSCP. Service Differentiation is achieved by a simple model [BBC+98], [BBCF01] where VPN traffic entering a network (DiffServ domain) is classified, possibly conditioned at the boundaries of the network to meet policy requirements in accordance with a specific classification, and assigned to different behavior aggregates. Each behavior aggregate is identified by a single DSCP. By using DSCP in network devices within the network packets can be classified and subjected to specific per-hop behaviour (PHB) - i.e. queuing or scheduling behavior. Thus, in the interior of the network, with the help of DS codepoint- PHB mapping [NBBB98], [BBCF01], the VPN traffic can be allocated a certain amount of node resources. Figure 2.3 shows Diffserv functionalities required at different points to provision the network. Each router is viewed with two interfaces and each interface having two queues. Traffic entering $Q1$ leaves through $Q4$ while traffic entering $Q3$ leaves through $Q2$.

Diffserv proposes two forwarding types to support different types of services: Expedited Forwarding (EF) and Assured Forwarding (AF). EF provides minimal delay, jitter, and packet loss, and assured bandwidth. In EF, the arrival rate of packets at a node must be less than the output rate at that node. Packets that violate the traffic profile are dropped or delivered out of sequence. It is generally believed [GLH+99] that most VPN users would demand Virtual Leased Line (VLL) type low loss point-to-point connection and the EF per-hop behavior (PHB) seems to be the appropriate choice. Scheduling mechanisms like Class Based Queuing (CBQ) [FJ95] or its variants [McK90], [SZN97], [SV97] are all

strong candidates to facilitate EF PHB. A typical SLS for such a service might include the ingress and egress points of the DiffServ domain that shall provide the service and a peak rate which can be guaranteed to the traffic stream.

AF PHB offers four classes, each containing three drop precedences. In a DiffServ domain each AF class receives a certain amount of bandwidth and buffer space in each node. The drop precedence indicates relative importance of the packet within an AF class. During congestion, packets with higher drop precedence values are discarded first to protect packets with lower drop precedence values. By having multiple classes and multiple drop precedences for each class, various levels of forwarding assurances can be offered. A typical SLS includes rates for low and medium drop priority packets and might also specify ingress and egress points.

Service Level Specifications (SLSs) must be established between a VPN customer and service provider and also among providers when a VPN tunnel spans over several DiffServ domains. These SLSs form the basis for traffic classification and conditioning actions. Using a BA classifier VPN traffic can be classified based on DSCP values in the packet header. Traffic can also be classified based on source and destination address, source and destination port, and protocol ID etc (called MF classifier).

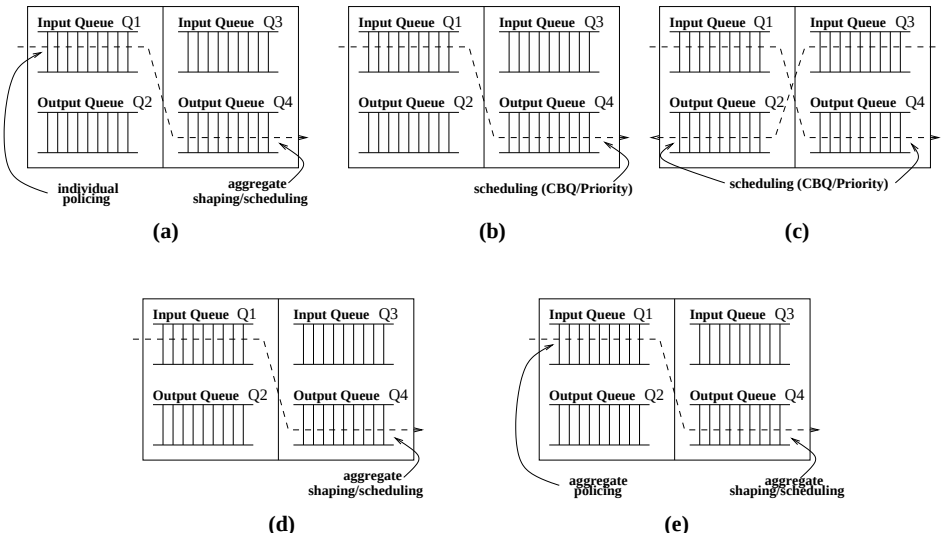


Figure 2.3: Diffserv Functionalities (a) Ingress Router, (b) Egress Router,(c) Interior Router, (d) Egress Router Connecting Neighbour Domain and (e) Ingress Router Connecting a Neighbour Domain

Conditioning involves metering, marking, shaping, and dropping (policing). A meter monitors traffic and determines whether classified traffic is meeting the specified traffic profile. A marker sets the DS field of a packet to a particular codepoint, adding the marked packet to a particular DS behavior aggregate. A shaper delays some packets in a traffic stream using a token bucket in order to ensure that traffic is in accordance with the traffic specification, often storing large bursts of packets until they can be safely released into the network. A policer drops excess packets from a flow if that flow violates the traffic specification.

When providing DiffServ to VPN tunnels, special attention must be given to DSCP mapping at the tunnel starting point. In outsourced VPNs, the ingress router of an ISP might perform DiffServ classification and traffic conditioning as well as tunnel encapsulation. If encapsulation is performed first, the ingress router can select the appropriate DSCP for the corresponding tunnel, while if DiffServ processing is done first, the DSCP of the inner IP header must be copied to the DSCP field of the outer IP header.

Management of a DiffServ domain can be done using so-called bandwidth brokers. A bandwidth broker maps SLSs to concrete configurations of DiffServ routers, in particular to edge routers of a DiffServ domain.

2.2.2 Traffic Engineering

With traditional IP routing schemes, VPN traffic follows the shortest path calculated by the ISP's Interior Gateway Protocol (IGP). The basic problem with this is that some links on the shortest paths between certain VPN tunnel ingres-egress pairs may get congested while links on the other possible alternate paths remain free. Traffic engineering provides the ability to move traffic flows away from the shortest path selected by the IGP and onto a potentially less congested physical path across the service provider's network.

Traffic Engineering has been done in the past based on overlay models, i.e. running IP over an underlying connection-oriented network technology such as ATM or Frame Relay. Such overlay networks are still in place. The IETF has introduced MPLS, constraint-based routing to do traffic engineering effectively and exploit the economies of the bandwidth that has been provisioned across the entire network. We will briefly review the techniques, benefits and limitations of these traffic engineering approaches.

IP-ATM Overlay Networks

In an IP over ATM overlay model (Figure 2.5) IP routers surround the edge of ATM cloud and communicate with each other by a set of Permanent Virtual Circuits (PVCs) that are configured across ATM physical networks (Figure 2.4). To interconnect IP routers of particular VPNs, ISPs can establish meshes of PVC connections that function as logical circuits. The overlay model not only provides a tunneling infrastructure and precise QoS, but also supports traffic engineering because of its ability to explicitly route PVCs to precisely distribute traffic across all links so that they are evenly utilized. However, the problem of IP-ATM overlay network is that it requires the management of two separate networks with different technologies resulting in increased operational complexity.

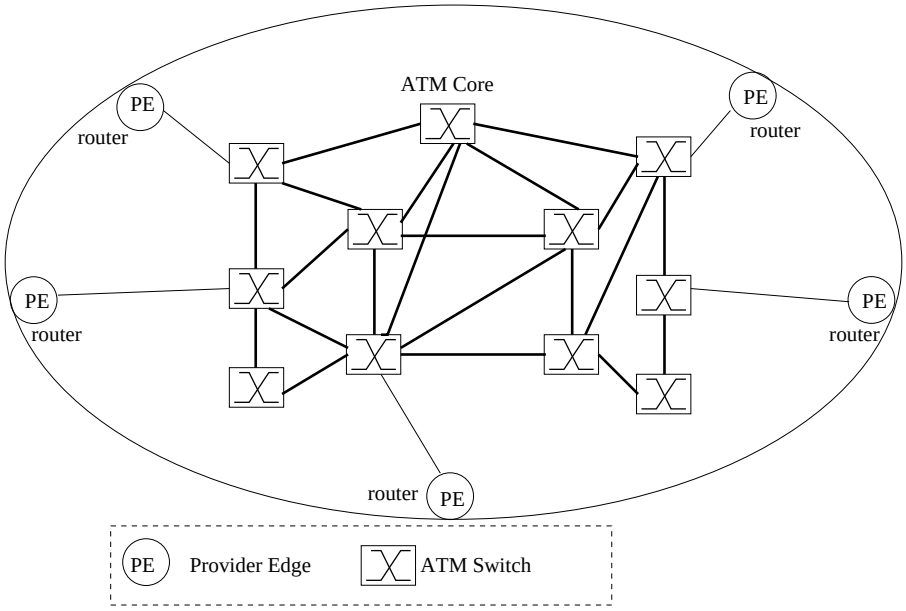


Figure 2.4: Physical Topology of IP-ATM Core

Multiprotocol Label Switching (MPLS)

Multiprotocol Label Switching (MPLS) offers an efficient and robust solution to the problems of over-utilized paths in transit networks by allowing to establish an explicitly-routed label switched path (LSP) to handle a large volume of traffic that takes a particular route (Figure 2.6). Using constraint-based routing with an utility tool path can be selected that offers the highest capacity with lowest

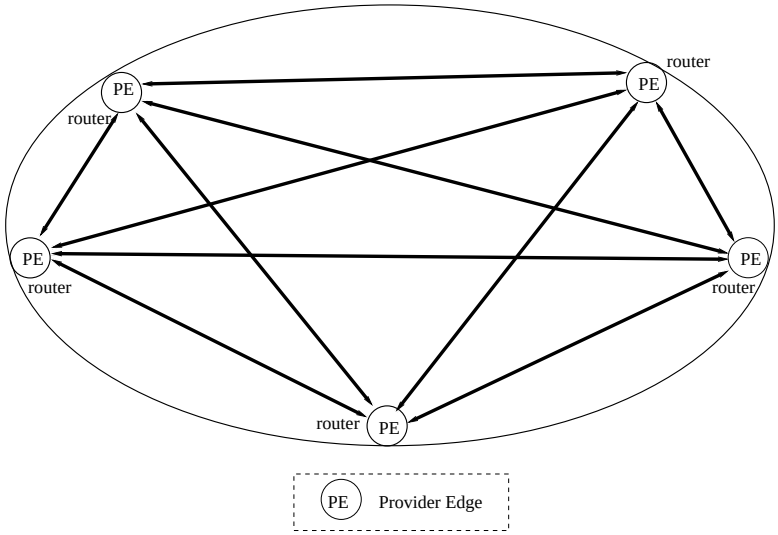


Figure 2.5: Logical IP-ATM Core

congestion and meets a VPN tunnel's performance requirements. Once a path has been determined MPLS can be used to setup LSPs to carry VPN traffic.

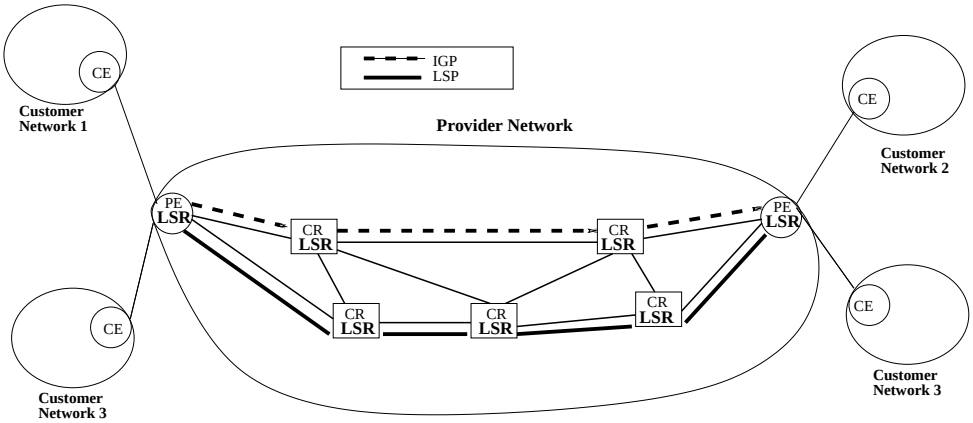


Figure 2.6: MPLS Traffic Engineering

An LSP is similar to an ATM PVC and is created by concatenation of one or more hops, forwarding a packet from one label-switched router (LSR) to another LSR across an MPLS domain. When IP packets arrive at ingress to an MPLS domain, label switching routers (LSRs) classify the packets into forwarding equivalence classes (FECs) based on a variety of factors, including, e.g., a combination of the information carried in the IP header of the packets and the local routing

information maintained by the LSRs. An MPLS label of 4 bytes and containing a 20-bit label field, 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 is then prepended to each packet according to their forwarding equivalence classes and forwarded to its next hop in the LSP. At the next hop the old label is replaced with a new label, and the packet is forwarded to the next hop along the LSP. This process is repeated at each LSR in the LSP until the packet reaches the egress LSR when the label is removed and the packet is forwarded based on destination IP address in the original IP header.

Another issue with MPLS is QoS support. Again, DiffServ fits nicely to MPLS. Both DiffServ and additional MPLS intelligence (packet classification, MPLS labelling, DSCP marking etc.) are required in edge routers, while interior routers just perform packet processing based on MPLS labels and DiffServ DSCPs.

MPLS overcomes several limitations of the overlay model concerning scalability and efficiency. Although it is a scalable technique and provides traffic engineering capabilities by its connection-oriented nature, and supports DiffServ QoS, it does not have built-in security mechanisms. However, higher security level can be achieved by using LSP as a QoS pipe and switching IPSec based tunnels over that pipe. This is shown in Figure 2.7.

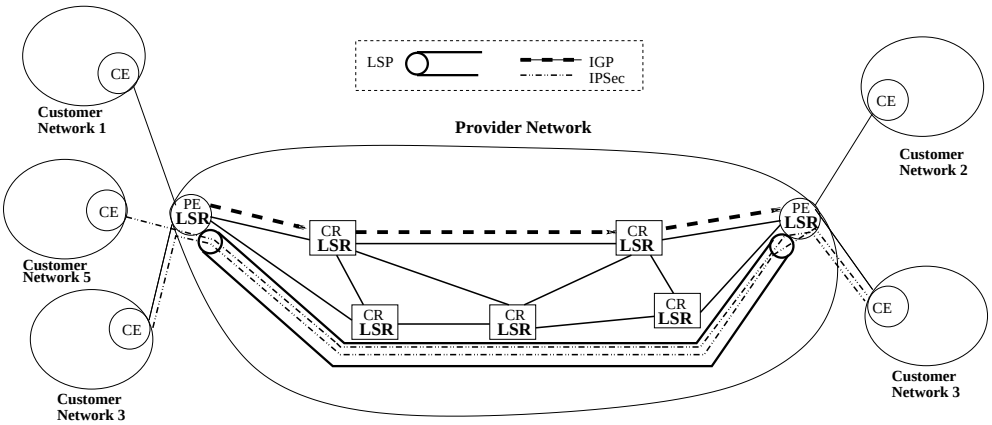


Figure 2.7: IPSec Tunnels over LSP Pipe

Constraint-Based Routing

With constraint-based routing routes for VPN traffic can be computed through a network that are subject to satisfaction of a set of constraints such as bandwidth,

hop count, delay etc. The constraints may also be imposed by the network itself or by administrative policies. If bandwidth is used as a constraint, and shortest path does not meet that constraint, the CBR may select a path that has more hop counts but is lightly loaded. Thus traffic is distributed more evenly. Path oriented technologies such as MPLS have made constraint-based routing feasible and attractive in public IP networks.

2.3 Policy-Based Provisioning

The Policy-based network (PBN) management paradigm [Gro] is nothing new and has been investigated extensively by both research and the commercial communities including the Internet Engineering Task Force (IETF). Policies are seen as a way to control network behavior by correlating business decisions with the overall network actions. In essence, a policy consists of a condition and an action to be applied when the condition is met. In general, it has the following format:

$if < condition(s) > then < action(s) >$

A condition is a set of expressions or objects used to determine whether a given action should be performed which may include parameters such as user id, IP addresses, traffic and application types. An action defines what must be done in order to enable a policy rule. It may result in the execution of one or more network operations to configure network devices. A network manager needs only to specify high-level business rules to achieve service differentiation. In a PBM system these business rules are captured by policy rules, which are stored in a policy repository. In actual network operations, policies are then deployed on individual devices where high-level policy rules are translated into device specific configurations.

2.3.1 IETF Policy-Based Architecture

The policy-based management architecture defined within the IETF/DMTF framework and used as the primary policy architectures by many commercial vendors, is shown in Figure 2.8. A detailed description of the concepts and their implementation within the IETF framework including policy validation and translation algorithms, policy distribution mechanisms and policy enforcement point algorithms can be found in [Gro], [YDP00]. The tools in this architecture used to automate policy enforcement are:

- *Policy Management Tool*: It is the interface between the network administrator and the system that allows administrators to specify policies to be enforced in network devices. It also translates an administrator's policy input into a format compatible with the policy repository.

- *Policy Repository*: The policy repository is a directory service or database used to store policies generated by the policy management tool. Lightweight Directory Access Protocol (LDAP) directory service is preferred by most to store policy objects.

- *Policy Decision and Enforcement Points*: A Policy Decision Point (PDP) retrieves policies from the policy repository, makes decisions based on retrieved information, and translates them into device specific configurations to send them to network devices also known as Policy Enforcement Points (PEPs). The PDP and PEP can be combined into a single device or may actually be different physical devices. COPS [BCD+00] is the IETF recommended protocol for communication between PDP and PEP.

2.3.2 Bandwidth Brokers

A Bandwidth Broker is a software agent that allows a customer of an ISP to buy bandwidth as a commodity, enforces the bilateral agreement between them by turning high level QoS policies into low level device configurations and pushing them to network devices to create the bandwidth service. If the bandwidth service needs to be extended to the neighbour domain, the agent can purchase the bandwidth from its peer by establishing another bilateral agreement called inter-ISP Service Level Agreement (SLA). The former agreement, i.e. contract between a corporate customer and ISP is often termed as customer-ISP SLA.

Bandwidth Brokers have originally been proposed in [NJZ99], [Sch98] to automate the activation of Virtual Leased Like (VLL) services by installing DiffServ PHBs and traffic conditioning functions in network devices. However, the creation of VLLs not only require appropriate PHB installations, but also advance bandwidth reservations and resource admission control. This is required to ensure that bandwidth resources are not over-sold. A Bandwidth Broker, therefore, is considered as a centralized server managing and controlling network resources of DiffServ Domain. As Bandwidth Broker essentially sells network resources to customers or peer ISPs and enables service differentiation, it must have a pricing

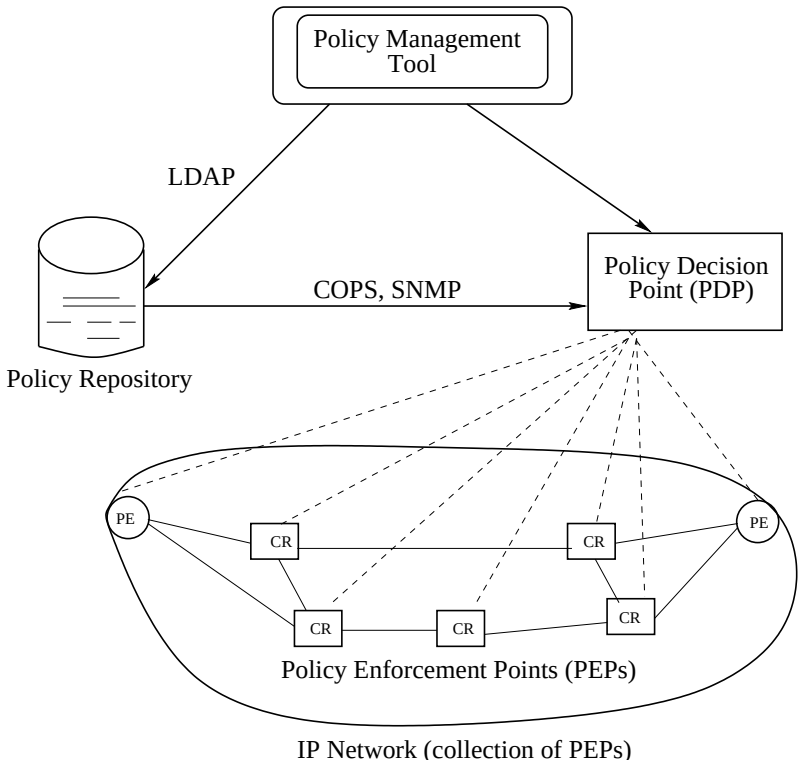


Figure 2.8: IETF/DMTF Policy-Based Management Architecture

and accounting mechanism to make profit out of the sales. In the context of the IETF policy based architecture, a bandwidth broker is seen as a PDP which creates services by translating user requests and business policies into low level device configurations and by sending them to the network devices. Although COPS and RSVP can be used to communicate with network devices to configure services in them, proprietary protocols are often used in bandwidth brokers to push network configurations to routers or switches.

2.4 Prevalent Network Management Architectures

Current network management applications based on prevalent network management architectures [MFZH99] typically comprise manager and agent components. The manager component traditionally resides on a central system and is accessed by an operator; the agent component resides remotely and actively monitors the health of a network device such as a router, hub, or switch. Existing network

management architectures found in practice today are:

- *Centralized management*: A single centralized Top Level Manager (TLM) looks after the management of the network. The manager, accompanied by a central management repository, monitors and configures the networks elements by communicating with agents that reside in those network elements.

- *Hierarchical management*: In hierarchical or weakly distributed management a TLM is supported by a numbers of Intermediate Level Managers (ILM) that communicate with the agents and perform requested operations on behalf of TLM. The centralized repository stays with the centralized TLM.

- *Cooperative management*: In a peer or cooperative network management system a set of TLMs or MLMs, each with a distributed repository, manage the different network domains and cooperate amongst them.

- *Fully distributed network management*: In such an architecture a large number of distributed managers share the responsibility of management and the management repository is partitioned and replicated.

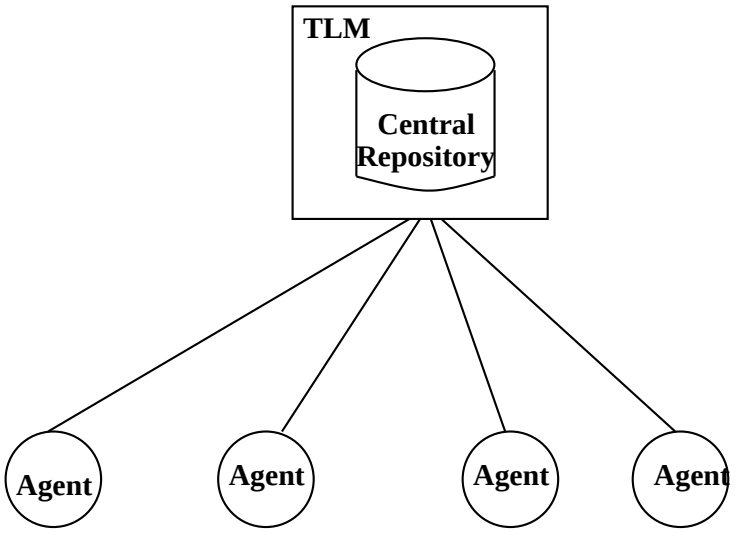


Figure 2.9: Centralized Management Architecture

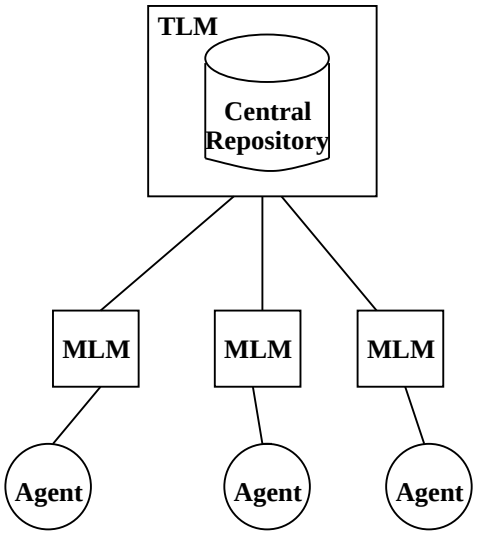


Figure 2.10: Hierarchical Management Architecture

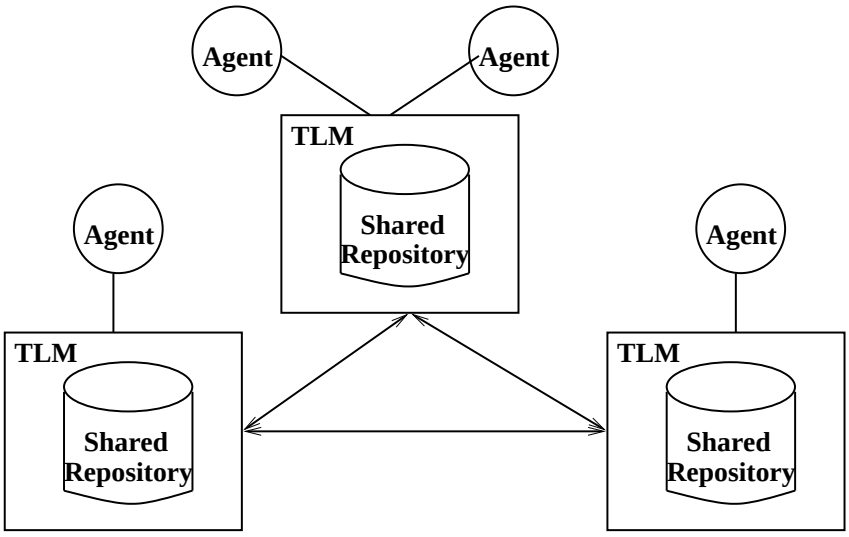


Figure 2.11: Cooperative Management Architecture

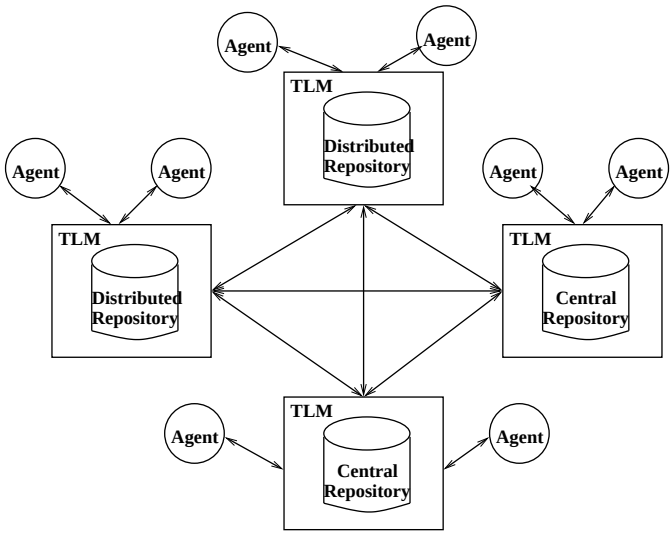


Figure 2.12: Distributed Management Architecture

2.5 Common Network Management Protocols

2.5.1 Simple Network Management Protocol (SNMP)

The agents in the traditional management architectures are usually the widely deployed SNMP [Sta98] of the TCP/IP protocol suite or CMIP [WBLH90]of the OSI world. Both of these protocols are designed according to a client-server based centralized model where the manager acting as a client collects and processes data by interacting with one or more agents acting as servers. Physical resources are represented by managed objects within these protocols. All managed objects in the SNMP environment are arranged in a hierarchical or tree structure. The leaf objects of the tree are the actual managed objects, each of which represent some resource, activity, or related information that is to be managed. The tree structure itself defines a grouping of objects into logically related sets called a management information base (MIB). Associated with each type of object in a MIB is an identifier of the ASN.1 type OBJECT IDENTIFIER. The object identifier is a unique identifier for a particular object type and serves to name the object. Some of the very well known problems of the SNMP based approach are:

- *Scalability*: SNMP supports lexicographic ordering, whereby the management station can supply an object or object instance identifier and then

ask for the object instance that occurs next in the ordering. However the absence of scoping and filtering reduces searching capabilities in SNMP. Searching in SNMP can thus lead to performance and scalability problems, as the addition of more and more objects to a MIB will slow down the retrieval rates of objects in the MIB. This could be a problem especially when one considers the large and growing number of standards-based MIB modules as defined in the periodically updated list of standard protocols.

- *Lack of ability to configure*: The vast majority of SNMP objects are *read-only*. SNMP does well in its ability to obtain network status information and determine network health. It is less powerful when it comes to making modifications to the network, although the SNMP `set` command (often achieved fully by a MIB Browser) allows an administrator to take some forms of corrective action via SNMP. The MIB of an SNMP device is usually fixed; it is constructed by the network equipment vendor (i.e. router manufacturer, computer hardware vendor, etc.) and cannot be added to or modified. The MIB tree is extensible by virtue of experimental and private branches, but only vendors can define their own private branches to include instances of their own products. MIB implementation for new emerging services like IPSec, MPLS, QoS do not even exist in most vendors equipments.

- *Security*: SNMP has no encryption standard. This is particular egregious because the read and write community names (which serve to limit access to a managed agent) are embedded in each SNMP message, and can be easily sniffed from of the network. However, newer version of SNMP [BW02] has made some effort to embed security mechanisms in it. The lack of security has served to reduce the usefulness of SNMP in the management of remote networks.

In short, SNMP, and hence the management architectures deploying it not only suffer from scalability and security problems, but also lack the ability to dynamically provision many emerging IP services.

2.5.2 Common Open Policy Services (COPS)

We earlier mentioned that COPS [BCD+00] is the recommended protocol for communication between a PDP and PEP in the IETF proposed policy-based

architecture. It is a simple query and response protocol that allows policy servers (PDPs) to communicate policy decisions to network devices (PEPs). In order to be flexible, the COPS protocol has been designed to support multiple types of policy clients.

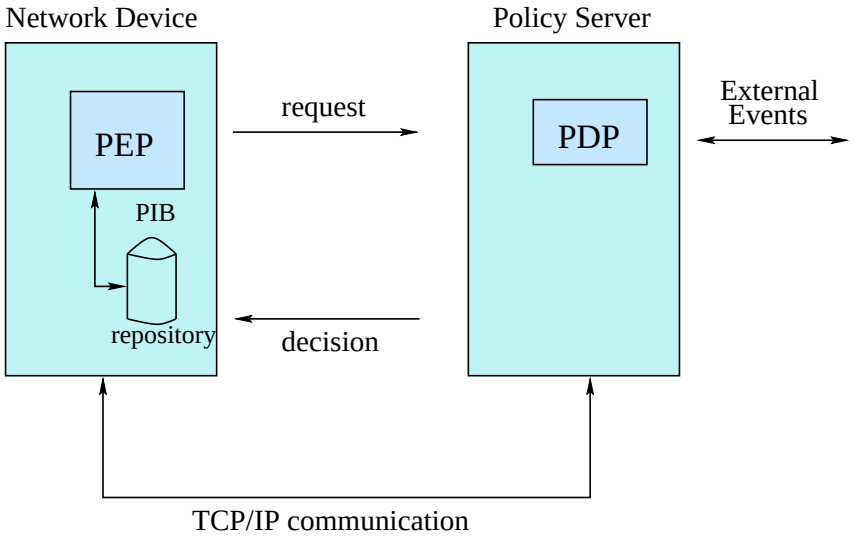


Figure 2.13: The COPS Protocol

The IETF architecture [Gro], [YDP00] supports the distribution of policies using both outsourcing (COPS-RSVP) and provisioning (COPS-PR) models:

- *Outsourcing model*: In the outsourcing model, on receiving a request, the PEP sends a query to the PDP and waits for its decision before responding to the request. This is useful when the PEP requires an immediate policy decision. In COPS-RSVP, when a RSVP reservation message arrives, the PEP must decide whether to admit or reject the request; it outsources this decision by sending a specific query to the PDP, and waits for a decision before admitting the pending reservation.
- *Provisioning model*: In the provisioning model, the PDP proactively configures the PEP based on user input and does not require PEP-PDP communication for each PEP event. Using COPS-PR, provisioning may be performed for a complete QoS enabled VPN configuration or only for the QoS or VPN portion of the configuration.

COPS data is a collection of policy rules. It uses a Policy Information Base

(PIB), analogous to MIBs in SNMP, as its global name space of provisioning policy. The PIB is common to both PDP and PEP and can be described as a tree, where branches of the tree represent policy rules classes (PRC) and the leaves represent policy rules instances (PRI). There maybe multiple instances of policy rules (PRIs) for any given rule class (PRC).

Many claim that although SNMP and MIBs are well suited to the requirements of device local configurations, COPS and PIBs are necessary to meet the higher-level requirements of policy-based management. However, COPS and PIBs are supported a in handful of network device types and network services.

2.6 Conclusion

In this chapter, we briefly described various QoS and VPN related technologies. The complexity of creating and managing QoS-VPN services in large networks make us realize that traditional network management systems and protocols are not well suited to cope with the rapidly evolving Internet security and QoS technologies. There is a need of automated QoS-VPN management system to be deployed by ISPs that would simplify service creation and management in large networks.

Chapter 3

Management of QoS Enabled VPNs

Although with the advent of Differentiated Services IP backbones can now provide various levels of quality of service (QoS) to VPN traffic, as today's network infrastructure continues to grow, traditional network management systems do not seem to cope well with the rapidly evolving Internet security and QoS technologies. Device-by-device network configuration is not only time consuming, but also error-prone and introduces policy lapses. Earlier in chapter 1 we introduced these challenges and argued that successful policy enforcement requires elevating and automating management process. In chapter 2, we briefly described the QoS enabled VPN (QoS-VPN) building blocks. In this chapter, we describe the policy-based Service Broker architecture and its implementation to automate and manage QoS-VPNs for customers that have Service Level Agreements (SLAs) with their ISPs and allows one such user to specify demands through a WWW interface to establish a VPN with certain QoS between two endpoints. This chapter not only shows the implementation details of various components of the Service Broker and connection admission and termination process, but also the pricing mechanism of such outsourced VPNs.

3.1 Introduction

Virtual Private Networks (VPNs) [GLH+99], [MKM+01], [FG99] enable secured private communications of distinct closed networks, for example, corporate networks, over a common shared network infrastructure. Rather than using expensive dedicated leased lines, VPNs use worldwide IP network services, including the Internet cloud and service provider's IP backbones to connect multiple geographically dispersed sites to each other into a private network. However, there is a growing demand that since private networks built on using dedicated lines offer bandwidth and latency guarantees, similar guarantees be provided in IP based VPNs. While the Internet has not been designed to deliver performance guarantees, with the advent of differentiated services [BBC+98], [BBC+99], IP backbones can now provide various levels of quality of service. The Expedited Forwarding (EF)[JNP99] Per Hop Behaviour (PHB) is the recommended method to build such a Virtual Leased Line Line (VLL) type point-to-point connection for VPN. This is absolutely critical to ensure that the VPN can deliver the myriad number of benefits of this rapidly growing technology.

However, the complexities introduced by VPNs and the requirement to provide QoS have made the job of the ISPs and systems administrators extremely difficult, and as today's network infrastructure continues to grow, the ability to manage increasing complexity is a crucial factor for VPN solutions. But, at the same time, this also opens the possibility for ISPs to sell VPN services to mostly corporate end users. For example, in a typical VPN-DiffServ deployment scenario (Figure 3.1), an ISP might offer to establish QoS enabled VPN between stub network A and B for a corporate end user who owns those networks and has regional offices there. Outsourcing VPN services not only would ease the job of the corporate customers, but also seems to be the only solution to dynamically construct end-to-end dedicated services over public IP infrastructure since customers have no control over resources in the transit network.

To offer such services, the ISPs will, however, need a management system not only to enable the users to construct services dynamically on demand, but also to provide new charging and billing facilities for the VPN services. In this chapter, we describe the architecture and implementation of such a management system called Service Broker (SB). As the World Wide Web (WWW) is widely available we provide web based interfaces as front ends where registered users can login, verify themselves and initiate a VPN based on their predefined SLA. This would obviate the need of invoking help from system administrator or ISP and

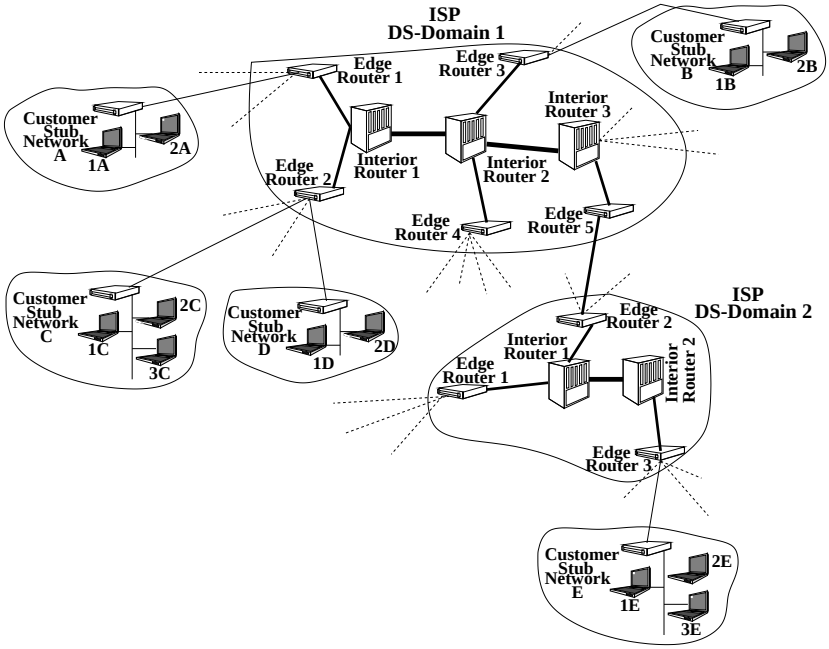


Figure 3.1: VPN-Diffserv Deployment Scenario

at anytime they can disconnect the VPN service or check their current bills.

The Rest of the chapter is organized as follows: section 3.2 gives a brief architectural overview of the SB needed to construct end-to-end service and section 3.3 describes the details of functional components that constitute the SB. Then in section 3.4, we explain the system flow during VPN connection establishment or termination process. Section 3.5 describes differential tunnel pricing mechanism and section 3.6 shows the experimental network setup with some examples and performance results. Finally, in section 3.7 we conclude this chapter.

3.2 Policy-Based Provisioning Architecture

Differentiated Services and VPN security can be seen as value-added network services. Such services allow the service providers to produce new revenues beyond pure connectivity services. However, the new services increase the complexity of the network management significantly. To address the challenges, in this section, we propose policy-based management for QoS-VPN in large networks

whose network control and provisioning are the most challenging aspects of modern computer networks. The policy-based architecture is compliant with the IETF/DMTF policy architecture [YDP00] and is an enhancement of our earlier work on Service Broker hierarchy [GBK99]. However, the service broker architecture itself is an extension of the so-called bandwidth brokers [NJZ99], [TWOZ99] to address the dynamic configuration and management of DiffServ.

3.2.1 A Service Broker Hierarchy for Configurable Services

The idea of Bandwidth Brokers can be extended to network services in general, including VPN services and service bundles such as QoS-VPNs. This leads to a framework called the broker hierarchy.

The hierarchy is structured in functional components. We made the distinction between intra-domain and inter-domain service tasks. Furthermore we distinguish between stand-alone (orthogonal) services and composed services. Finally we added a layer to hide the diversity of network equipments. This results in a four-layered broker hierarchy as depicted in Figure 3.2. The configuration daemons (CD) hide the heterogeneity of the underlying network equipment. The internal service broker (ISB) manages the provider controlled network resources for a service, and coordinates them. For example, referring to Figure 3.1, if a user who has a SLA with the ISP maintaining DS-Domain 1 wants to establish a tunnel between host 1A in stub network A and host 1B in stub network B, the ISB of that domain can handle that request to establish the desired connection. The external service broker (ESB), however, handles the negotiation between brokers of peer ISPs across the trust border. The necessity of the ESB is obvious when the customer in the previous example wants to setup a tunnel between host 1A and host 1E in stub network E. Since stub network E resides by the perimeter of another domain (DS-Domain 2), the ESB of domain 1 needs to negotiate with the ESB of domain 2. The result of a negotiation is an inter-ISP service level agreement (SLA), which describes the collaboration between the two provider networks. A broker signaling protocol is necessary to set up an unbroken chain of SLAs between all involved ISPs in order to set up an Internet wide service.

The broker hierarchy and its rationale is described in more detail in [GBK99]. The work presented in this chapter is an instance of this broker hierarchy framework. We focus on the implementation of a DiffServ VPN service using the IPSec [KA98] protocol. The synergy between DiffServ and IPSec [SBGP99] and a single

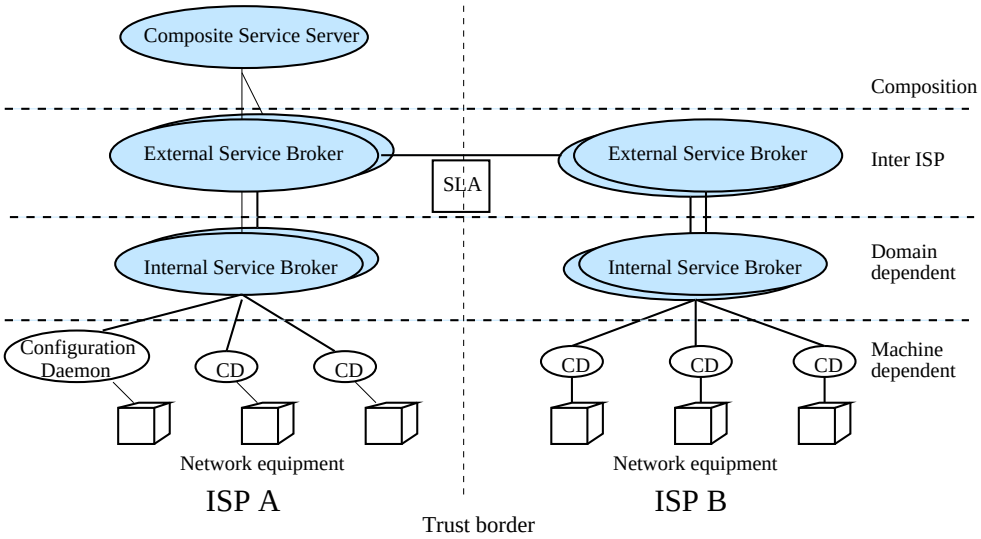


Figure 3.2: The Service Broker Hierarchy.

administrative domain scenario lead to a stand-alone implementation using only the ISB and the CD layer. In this chapter whenever we refer to Service Broker (SB), we essentially mean ISB.

3.2.2 Service Broker Architecture

A Policy Based Network management solution is a way to automate and ease QoS-VPN and other advanced IP services activation and ongoing management. Instead of manually configuring each network device via Command Line Interface (CLI) the policy-based QoS-VPN Management solution can feature a streamlined end-to-end activation process through interfaces that enables both network administrators and end users to apply settings to multiple devices with ease. Programmability of network components to dynamically create services tremendously saves time and makes configuration management error free. In this section we describe the policy-based Qos-VPN service provisioning architecture that forms the basis for further works in terms of implementation and advanced policy-based dynamic service provisioning in subsequent chapters.

The Service Broker architecture is shown in Figure 3.3. The main objective of this is to capture high level business policies and convert those into device specific configurations. In order to achieve that we have a QoS-VPN policy builder that takes policy input from network administrators or corporate end users. The SB

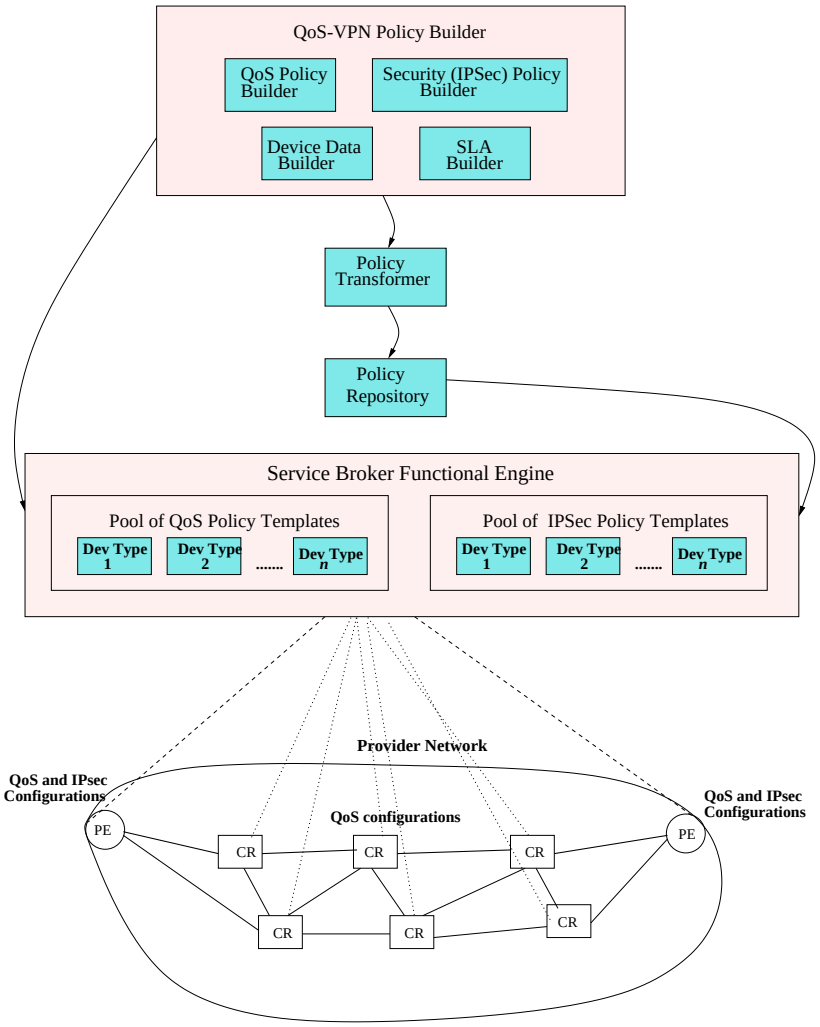


Figure 3.3: Service Broker Architecture for Management of QoS-VPNs

functional engine that converts them to low level configurations with a pool of device specific templates. Since QoS-VPNs require defining both user QoS and security (IPSec) policies for the corporate network, the architecture facilitates two separate builders - QoS policy builder and IPSec policy builder. Additionally the policy builder bundles device data builder and a SLA builder. The Device data builder stores device related data for automated device access and translation of high level policy by templates in the functional engine. The SLA builder facilitates contracts between the customer and service provider. The Policy transformer converts policy data it receives from the policy builder into a proprietary format

before sending them to the policy repository. The Service Broker functional engine contains pools of QoS and IPSec policy templates that are device specific. As new device types are added, administrators can simply add required QoS and IPSec policy templates to the pool. This is an extremely useful feature in a growing network where the ability to operate in a multi-vendor environment is considered crucial.

QoS Provisioning in DiffServ Networks does not only mean determination and allocation of resources necessary at various points in the network, but also includes modification of the existing resources. Both quantitative, as is the case with VPNs, and qualitative traffic (some assured service) are required to be provisioned at the network boundaries and in the network interior. Determination of resources required at each node for quantitative traffic needs the estimation of the traffic volume that will traverse each network node. While an ISP naturally knows from the SLA the amount of quantitative VPN traffic that will enter the transit network through a specific edge node, this volume cannot be estimated with exact accuracy at various interior nodes being traversed by VPN connections, if we do not know the path of such connections [Ash99]. However, if the routing topology is known, this figure can almost be accurately estimated. If the default path does not meet the requirements of an incoming connection, alternate and various QoS routing [CNRS98], [CN98] methods can be used to find a suitable path and enforced by MPLS techniques [FWD+01].

Based on the basic needs of provisioning a DiffServ enabled VPN network to support quantitative services we consider the provisioning as a two-layered model: the top layer responsible for edge provisioning and driving the bottom layer, which is in charge of interior provisioning (Figure 3.4). Service Brokers in our policy-based architecture support this layering and are not only capable of performing dynamic end-to-end admission control to set up a leased line like VPN by maintaining the topology as well as policies and states of all nodes in the network, but are also capable of managing and provisioning network resources of a separately administered DiffServ domain and cooperating with other similar domains.

3.3 Service Broker Implementation

The SB, which is the heart of our QoS-VPN management system and designed based on the earlier proposed architecture, takes the role of a QoS manager to

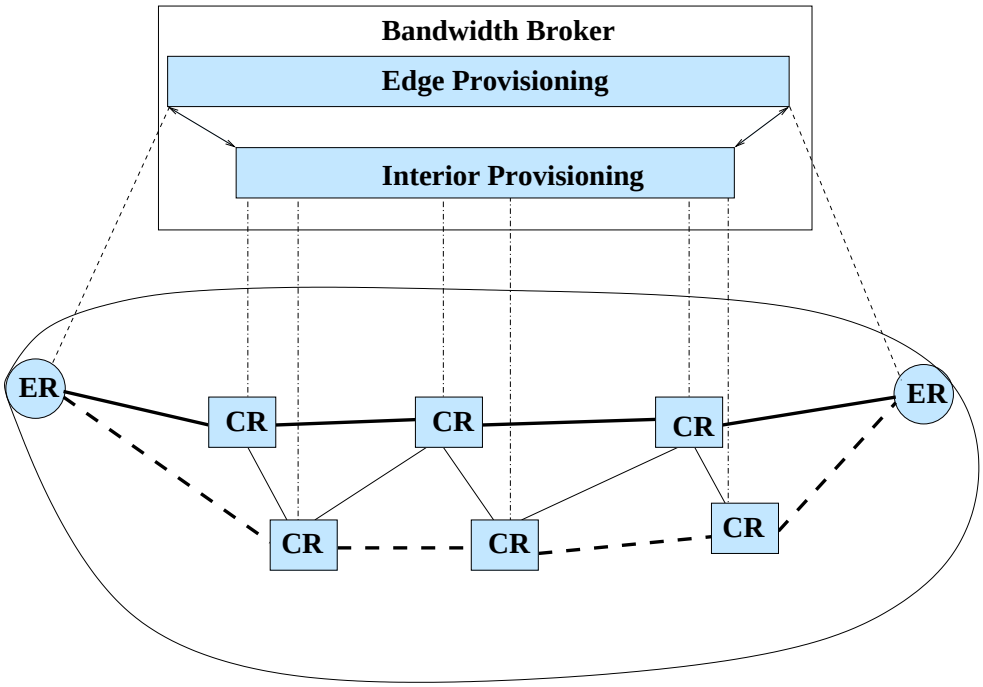


Figure 3.4: Layered Provisioning of VPN-Diffserv Networks

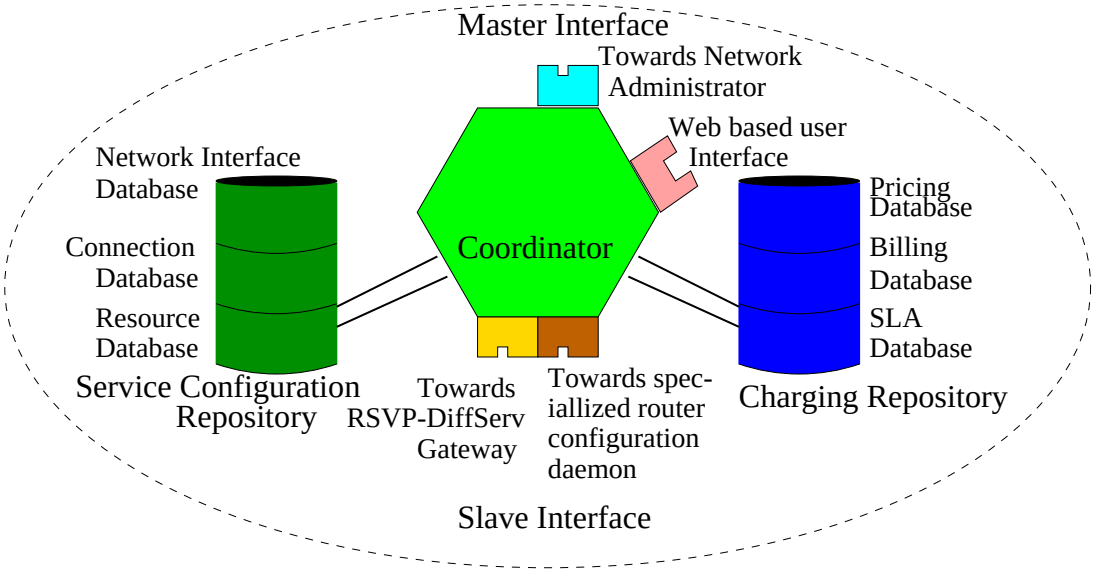


Figure 3.5: Service Broker Functional Elements

optimally configure network resources and adaptively decide based on user preferences and resource availability. These decisions could take place with minimum user intervention with respect to specifying the user's requirements. As the underlying network may provide different classes of services to satisfy various VPN customers, by identifying the generic functionality provided by any resource, we present our SB (Figure 3.5) with a standard interface (Figure 3.6) to the network resource.

Figure 3.6: QoS-VPN Web Interface

In this section we will identify the prerequisites for dynamic VPN service activation on demand by examining some simple configuration examples of widely used Cisco [Sit] routers and based on those requirements define the essential components of the Service Broker that we have implemented.

3.3.1 Identifying Design Rationale

In this section, referring to the experimental setup of Figure 3.7 we will present a configuration example in routers when an administrator needs to setup a 1 Mbps tunnel between the routers 130.92.70.101 and 130.92.66.141 for source 172.17.0.103 and destination 172.20.0.100 to meet the demand of a certain user. We need to mention that traffic flowing from source 172.17.0.103 traverses interior interfaces 130.92.70.1 and 130.92.66.1 before reaching the destination. Assume that both of these interfaces have been configured to support 10 Mbps and 15 Mbps of premium service (EF traffic)

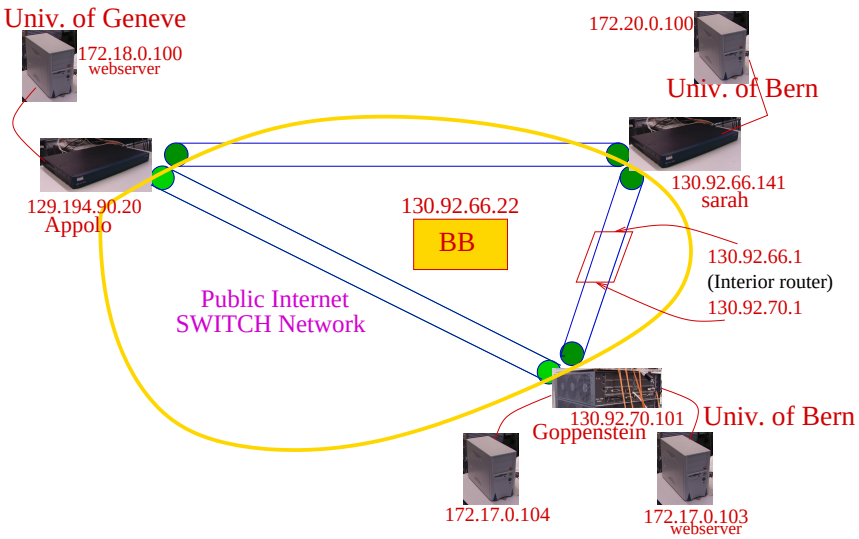


Figure 3.7: Experimental Setup of VPN

To setup such a tunnel the administrator needs to configure appropriate commands for classification, IKE (Internet Key Exchange) policy, various IPSec tunneling/encryption methods that might be used with the tunnel, commands that specify peer router's necessary identification to establish the tunnel. The classified flows also need to be possibly policed and marked at the inbound and shaped at the outbound on aggregated basis. Basically, there are mainly two functionalities that need to be provided by the edge routers: VPN tunneling, QoS (policing, shaping). For the requested tunnel to be established, the administrator, therefore, configures Router 130.92.70.101 (Cisco 7206) and 130.92.66.141 (Cisco 2611) as shown in Figure 3.8 and 3.9 respectively.

```
/* IKE Policy commands specifying hash method, key sharing method, key of peer */
1    crypto isakmp policy 1
2    hash md5
3    authentication pre-share
4    lifetime 500
5    crypto isakmp key lab-tunnel address 130.92.66.141

/* Various tunneling/encryption methods that can be used with the requested tunnel */
6    crypto ipsec transform-set ah-md5-hmacANDesp-des ah-md5-hmac esp-des
7    crypto ipsec transform-set ah-md5-hmac ah-md5-hmac
8    crypto ipsec transform-set ah-rfc1828 ah-rfc1828
9    crypto ipsec transform-set ah-sha-hmac ah-sha-hmac
10   crypto ipsec transform-set esp-encryption esp-des

/* Commands to create tunnel between  routers 130.92.70.101 and
130.92.66.141 for source 172.17.0.103 and destination 172.20.0.100 */
11   crypto map cati-tunnel 143 ipsec-isakmp
12   set peer 130.92.66.141
13   set transform-set ah-md5-hmacANDesp-des
14   match address 143

/* Aggregate traffic shaping rate at the outbound */
15   interface FastEthernet0/0
16   traffic-shape group 150 10000000 1000000 1000000 1000
17   crypto map cati-tunnel

/* policing indvidual VPN connection at the inbound */
18   interface FastEthernet1/0
19   rate-limit input access-group 143 1000000 2000000
     8000000 conform-action set-prec-transmit 1
     exceed-action drop

/* Classifying all VPN packets that originate from 130.92.70.101. Used for aggregate sharing */
20   access-list 150 permit ip host 130.92.70.101 host any

/* Classifying the requested traffic */
21   access-list 143 permit ip host 172.17.0.103 host 172.20.0.100
```

Figure 3.8: Configuration of Router 130.92.70.101

The comments in the configuration example of router 130.92.70.101 already explain briefly which part of the scripts are responsible for the various tunneling and QoS functions. Note that in line 5 the peer router's ip address and the *shared secret key* are the same. ISPs might actually enter the same for other peers in a domain even before establishing tunnels for between any two peers. We can also see in lines 6-10 numerous tunneling/encryption methods have been defined and only one of these is used (line 13) for the desired tunnel that is specified by the commands in lines 11-14. Also note that in line 21 traffic is classified with a specific number (access-list number) and the same number is used in lines 11,

```
1    crypto isakmp policy 1
2    hash md5
3    authentication pre-share
4    lifetime 500
5    crypto isakmp key lab-tunnel address 130.92.70.101
6    crypto ipsec transform-set ah-md5-hmacANDesp-des ah-md5-hmac esp-des
7    crypto ipsec transform-set ah-md5-hmac ah-md5-hmac
8    crypto ipsec transform-set ah-rfc1828 ah-rfc1828
9    crypto ipsec transform-set ah-sha-hmac ah-sha-hmac
10   crypto ipsec transform-set esp-encryption esp-des
11   crypto map cati-tunnel 145 ipsec-isakmp
12   set peer 130.92.70.101
13   set transform-set ah-md5-hmacANDesp-des
14   match address 145
15   interface FastEthernet0/0
16   traffic-shape group 150 15000000 1500000  1500000 1000
17   crypto map cati-tunnel
18   interface FastEthernet1/0
19   rate-limit input access-group 145 1000000 2000000
     8000000 conform-action set-prec-transmit 1
     exceed-action drop
20   access-list 150 permit ip host 130.92.66.141 host any
21   access-list 145 permit ip host 172.20.0.100  host 172.17.0.103
```

Figure 3.9: Configuration of Router 130.92.66.141

14 and 19. Although it is not necessary to maintain the same number in lines 11 and 14 at least, this is a must in line 19 where classified packets are policed and marked. However, by maintaining the same number we can maintain a unique tunnel ID.

With all these explanations now let us examine what we need to do in order to establish another 2 Mbps tunnel between the same routers for source 172.17.0.104 and destination 172.20.0.100. If we consider only uni-directional QoS then we need policing/marking command only in router 130.92.70.101. Here the interface of 130.92.70.101 supports C_{TOTAL} =10 Mbps for quantitative VPN traffic. Earlier we accepted a 1Mbps connection i.e. $C_{allocated}$ = 1 Mbps and now $C_{request}$=2 Mbps. By performing the admission check we see that $C_{allocated} + C_{request} < C_{TOTAL}$.

Since only a 1 Mbps connection was installed previously, end-to-end admission test also signals positively, i.e. an interior router interface 130.92.66.1 also passes the admission check for this uni-directional QoS support. We can, therefore, accept the connection. All we need to do is add the following routing commands in router 130.92.70.101 and also a similar set of commands in router 130.92.66.141.

```
crypto map cati-tunnel 144 ipsec-isakmp
set peer 130.92.66.141
set transform-set ah-md5-hmacANDesp-des
match address 144
interface FastEthernet1/0
rate-limit input access-group 144 2000000 2000000
 8000000 conform-action transmit exceed-action drop
access-list 144 permit ip host 172.17.0.103
 host 172.20.0.100
```

3.3.2 Prerequisites for Dynamic Configuration

We have seen from the above example how routers are actually configured (manually) and what needs to be done in the routers if additional tunnels need to be established between two peer routers. Based on this experience of section 3.3.1 and admission control requirement, we will now try to identify the prerequisites of an automated system like a Service Broker capable of dynamic service configuration and mimic a system administrator. These are:

- *User and Request Validity*: The system should be able to identify the user and the validity of a request.
- *VPN Readiness*: Edge routers should be *VPN Ready*. This means that IKE policy commands from line 1 to 5 and command for specifying tunneling/encryption methods (lines 6 -10) should be pre-configured in the VPN edge routers so that when a new tunnel is configured it can choose one of previously configured tunneling/encryption methods (for example, in line 13 `ah-md5-hmac esp-des` is chosen).
- *Resource management of edge and interior routers*: Each edge and interior router need to be pre-configured to support a certain amount of quantitative traffic as specified by ISP. The system should also be able to track the resource usage so that admission control can be performed to have a well provisioned system.
- *Topology maintenance*: The system should store the topology of the network. When a connection request arrives the system should know which transit path (i.e. which interior routers the VPN connection will traverse) will be followed.

- *Management of existing connections*: The same connection that is already established cannot be established with a different tunnel ID.

- *Management of interfaces*: The system should be intelligent enough to identify which routers and interfaces should be configured.

- *Remote configuration of routers*: The system should be able to configure the appropriate routers to activate the service immediately without the invocation of a human operator.

3.3.3 Service Broker Databases

Based on the above prerequisites the SB has been developed to establish VPN tunnels dynamically on customer demand and also to allocate QoS to them. The SB interacts with specialized configuration daemons (CD) when a certain user request arrives to setup a QoS-VPN tunnel. The basic operation of our system is as follows: based on request parameters provided by the user, the SB first contacts a SLA database to check the validity of the user and its request parameters. It then checks with the connection database whether a similar requested connection already exists or not. If this is not the case, the SB looks at its resource database to decide if the tunnel can be established. A positive answer would then lead to a tunnel establishment by the CD. When a user disconnects the VPN tunnel, the SB releases resources and invokes the pricing database to calculate the pricing for that tunnel. In the following we will briefly describe these components and their role before we move to the detailed description of system flows in section 3.4.

- **SLA Database**: The SLA database does not only contain the user's identification but also specifies the maximum amount and type of traffic he/she can send and/or receive for a tunnel. As we are concerned about closed user groups, a SLA also contains the boundary of a valid VPN area. Referring to Figure 3.1 where stub networks A, B and C might belong to the same organization located at different remote locations, one can easily see that they form a mesh environment and any site may want to establish with the other under the same contract. Therefore, this set of locations defines the perimeter of the VPN area and are entered in this database as source and remote stub addressess. The User authentication process prohibits malicious users to setup unauthorized tunnel and access network resources illegally.

The SLA, however, allows users to add new VPN areas to his old contracted list of valid VPN areas. It contains the following tuple:

<User ID, Password, Maximum BW in Mbps, Source Stub Address, Remote Stub Address>

- **Resource Database**: The resource database contains resources available between any two edge routers. This means that this database has resource information of all the routers in a certain domain. In our implementation we keep records of pre-computed tunnels with Tunnel IDs. For each tunnel, however, we also need to know its ingress router's ip address, tunnel source address (which might be the same as ingress router's ip address), egress router's ip address, tunnel destination address (this might as well be same as egress router's ip address) and the capacity of the tunnel in Mbps. Also, we need to keep track of the status of the tunnel in terms of availability. Therefore, the tuples are:

  <tunnel id, ingress router, tunnel source addr, egress router, tunnel destination addr, bandwidth, status>

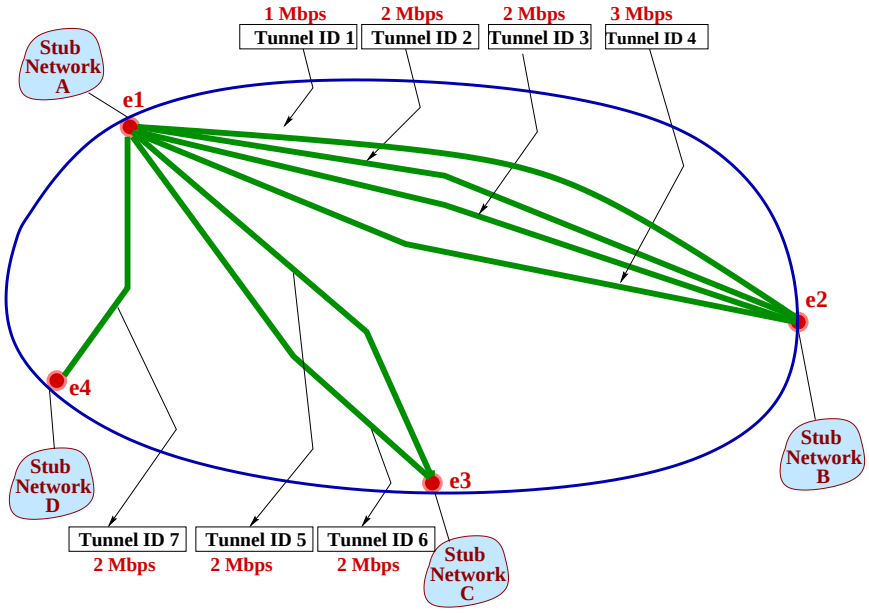


Figure 3.10: Mapping of Resources to Various Tunnels

However, it should be clarified that a tunnel might originate from an ingress router to several possible egress routers. Referring to Figure 3.10, users

residing in the stub network A might want to establish tunnels between router e1 and e2, or between e1 and e3, or between e1 and e4 to communicate with other stub networks. Assume that the ISP has decided to allocate a maximum 10 Mbps capacity to traffic stemming from e1 and destined towards other edge points. The ISP might, however, allow one 1 Mbps, two 2 Mbps tunnels, one 3 Mbps tunnel to be created between e1 and e2 and also allow two 2 Mbps tunnels from e1 to e3 and only one 2 Mbps tunnel from e1 to e4. This would result in a map as shown in table 3.1. In the table while ingress and egress router addresses are necessary for identifying the edge routers, the tunnel source and destination addresses are needed to create tunnels. Ingress (or egress) router and its corresponding tunnel source address (or dest. addr for egress) might be the same if the same address is used in both cases.

Tunnel ID	Ingress Router	Tunnel Source Address	Egress Router	Tunnel Dest. Address	Bandwidth in Mbps	Status
1	e1	e1	e2	e2	1	1
2	e1	e1	e2	e2	2	1
3	e1	e1	e2	e2	2	1
4	e1	e1	e2	e2	3	1
5	e1	e1	e3	e3	2	1
6	e1	e1	e3	e3	2	1
7	e1	e1	e4	e4	2	1

Table 3.1: Resource Table for the Network of Figure 3.10

It is clear from this mapping that if all the tunnels as mapped in the table 3.1 are active simultaneously, then router e1 would need to support 14 Mbps. Since we have only 10 Mbps for this router we need to keep track of each router's capacity and perform an admission control. For any ingress interface if C_{TOTAL} is the total capacity reserved for VPN traffic, $C_{allocated}$ is the bandwidth that is already allocated to existing tunnels, and $C_{request}$ is the requested capacity of the new incoming connection, then C_{TOTAL} should be more than or equal to $(C_{allocated} + C_{request})$.

- **Connection Database**: The connection database contains a list of currently active VPN connections. Here a connection always mean a VPN tunnel. It has various functions: (i) when a new request arrives for connection or termination, the SB can check if that connection already exists or not and then make its decision, (ii) it indicates how much resources have been consumed by VPN users, (iii) and provides a record to pricing mechanism. Its tuples are:

<user id, source address, destination address, bandwidth, tunnel id, activation time>

- **Interface Database**: The interface database contains necessary records of edge routers that are used as tunnel end-points for the outsourced VPN model. In such a model since some customer stub networks are connected to the ISP edge router we need to specify which stub networks are connected to a particular edge router. Also, an edge router might have one or more inbound and outbound interfaces which also need to be specified for each stub network that is actually connected to a particular inbound interface of a router. This is important because normally at the inbound interface tunnels are policed on an individual basis and at the outbound they are shaped on an aggregated basis. At the same time, outbound interfaces are also used as the tunnel endpoints. Finally, a tunnel map to which all defined tunnels are attached is also part of the record in this database that is activated at the outbound interface of the router. The tuples are :

  < stub network, edge router, generic router name, inbound interface, outbound interface, tunnel map name>

- **Pricing and Billing Database**: The pricing database contains pricing information of various tunnels. Its only interaction with the SB is at the time when a connection (tunnel) is terminated and the SB needs to know the price of that by making a query to it. The billing database contains details of terminated connections and their computed price. For details see section 3.5.

3.4 Description of System Flows

In this section we will describe how a connection is established or torn down, how various components interact with the SB, and under which circumstances a new connection request or tear down request are refused.

3.4.1 Successful Connection Establishment

Figure 3.11 shows all the communications involved in setting up a VPN connection between two stub networks or simply between an originating host and the remote

host. We will describe the operational details by referring to the communication marked on Figure 3.11. Considering each communication in turn :

- 1) A user sends a connection request message to the SB for a new connection request via web or via other signaling mechanisms such as RSVP. The SB is in charge for determining whether the connection should be allowed or refused. It achieves this by communicating with each of the components in turn. The request contains user id, user password, source and remote stub addresses, amount of bandwidth and encryption/encapsulation method.

- 2,3) The SB contacts the SLA database that is responsible for validating the user and his request. If the user is identified correctly, his source and remote address conforms the contract, and also the bandwidth requested is less than or equal to the agreed traffic contract, it sends a positive response.

- 4,5) The SB contacts the configuration daemon to check its status. The status can be busy, available, or down. Only in the case of availability the user request can be processed further.

- 6,7) The SB contacts the connection database to check the existence of an exactly similar tunnel. This is because for a source and destination pair only one tunnel can remain active.

- 8,9) The SB asks the resource database to allocate a tunnel of a certain bandwidth. The resource database responds to the SB and either allocates the resource or denies based on resource availability.

- 10) The SB allocates the requested resource and tells the configuration daemon to create appropriate configuration scripts. In the meantime the resource and the connection database update their records. The new connection request data is appended to the connection database and the tunnel that has just been allocated from the resource database is marked as used.

- 11,12) The CD puts a busy signal on itself and creates the configuration scripts by using policy templates. It then sends configuration scripts to the routers. The routers send signals to the CD.

- 13,14) The CD removes the busy signal from itself and sends an acknowledgment to the SB which sends a notification to the user.

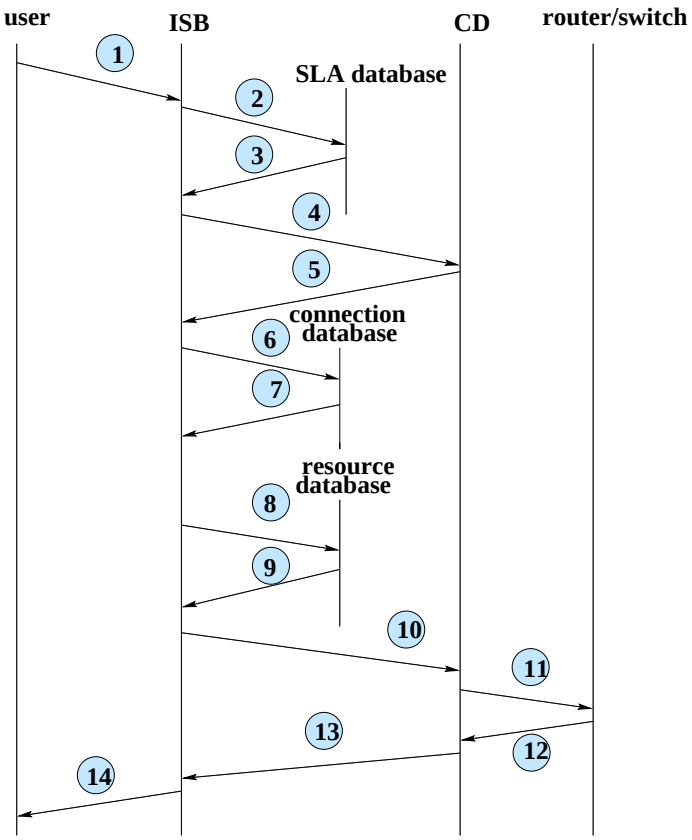


Figure 3.11: Successful QoS-VPN Connection Setup

3.4.2 Successful Connection Termination

Referring to Figure 3.12, the system walks through the following steps for successful connection termination:

- 1,2,3,4,5,6,7) These steps are similar to the steps mentioned for connection setup. However, in step 2 only user id and password are checked. We also need to see if the daemon is busy or not, and also that the requested connection exists in the connection database. In summary, once a termination request arrives the system needs to make sure that the tunnel exists and was created by the same user who has sent the termination request.

- 8) If the connection is found in the connection database the SB talks to

the CD to create an appropriate configuration script. In the meantime the connection record is deleted from the connection database and the resource database updates its records by making the same tunnel available which has just been deleted.

- 9,10,11) The CD creates and sends the configuration script to the router. The router sends a signal to the CD which then confirms the SB about the configuration.

- 12,13,14) Before the SB finally forwards this positive acknowledgment to the user it invokes the pricing database and calculates the appropriate pricing using the method described in section 3.5 and stores the necessary information of the terminated connection and computed price in the billing database. Once that is done the user receives the acknowledgment via the web interface.

3.4.3 Connection Rejection and Failure in Termination

A connection request is rejected if (i) the SLA profile does not match (case D-1) , (ii) the daemon is found busy (case D-2), (iii) the connection already exists (case D-3) or (iv) not enough resources are available (case D-4). The various stages where a connection creation process might get refused are shown in Figure 3.13. We will briefly describe them in the following:

- Case (D-1): User id, password might be wrong, VPN areas for which user wants to establish tunnel might be invalid, or the bandwidth requested might be higher than the agreed one, and in such a case the user will not be granted a connection.

- Case (D-2): Even if the request parameters are valid the CD might be found busy and hence, the connection will not be possible.The system can, however, be tailored for automatic retry as desired by an user.

- Case (D-3): If the request passes the above two stages successfully, it might be found that a connection already exists in the connection database. In such a case the request will be refused.

- Case (D-4): If the above cases do not happen during a tunnel creation process then the SB asks the resource database to grant the request user

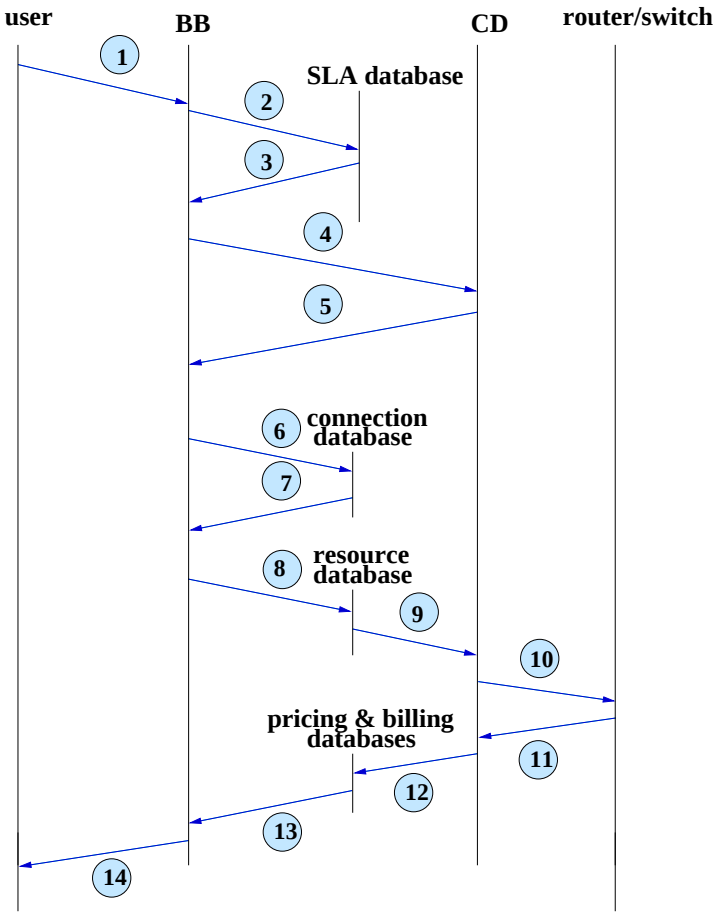


Figure 3.12: Successful Connection Termination

has asked for. If the resource database cannot meet that demand it sends a signal to the SB about resource unavailability, and the SB forwards that message to the user.

Other than case (D-1) and (D-2) whose actions are quite obvious, the connection termination request (Figure3.14) might be rejected (case D-3) if no connection entry is found in the connection database for deletion of a request or if the tunnel belongs to the someone else other than the requester.

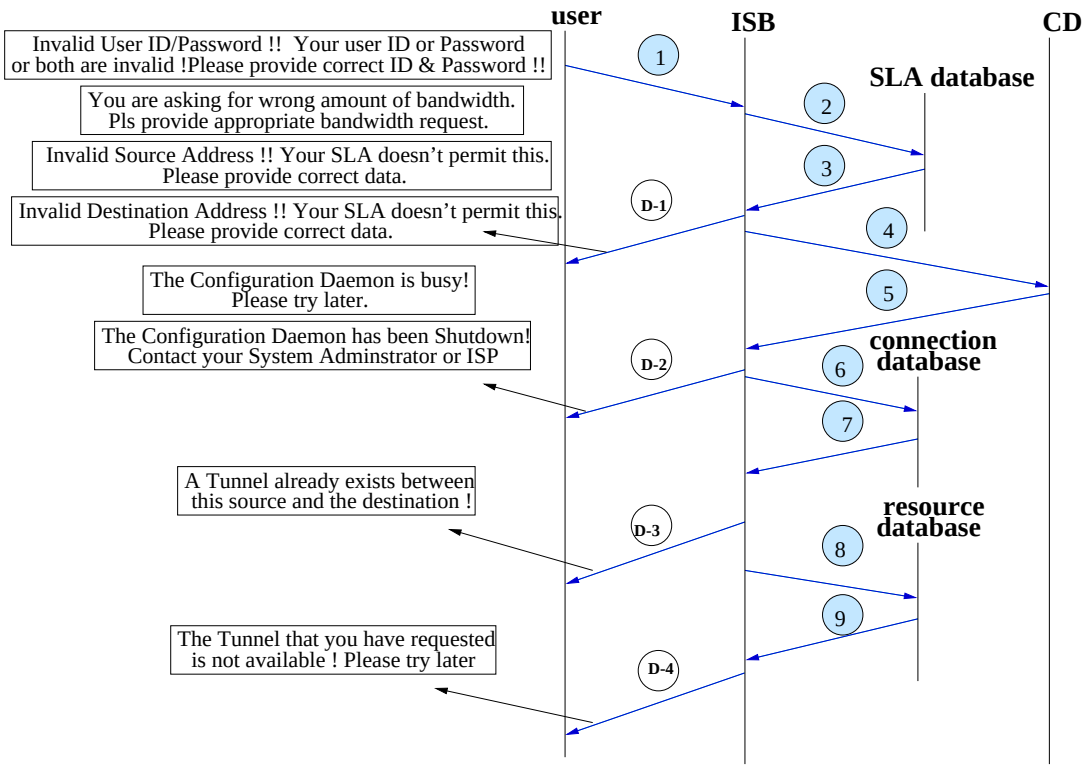


Figure 3.13: Denial of Connection Setup

3.5 VPN Pricing

Although the current flat rate pricing with uniform best effort data transport service is simple and attractive, it has many defects. It provides a single level of service quality, and does not allow users to select what is best for their needs. To many, this leads to misallocation of resources. To deal with this, there are proposals to regulate the usage by imposing fees based on the amount of data actually sent. This, however, is fundamentally flawed as usage based fees would impose usage costs on the user whether the network is congested or not and might even collapse the whole revenue model [Cla99].

3.5.1 VPN Pricing Model

With our VPN Service model defined in earlier sections, Internet Service Providers are going to provide a variety of services through multiple service classes (e.g. 1

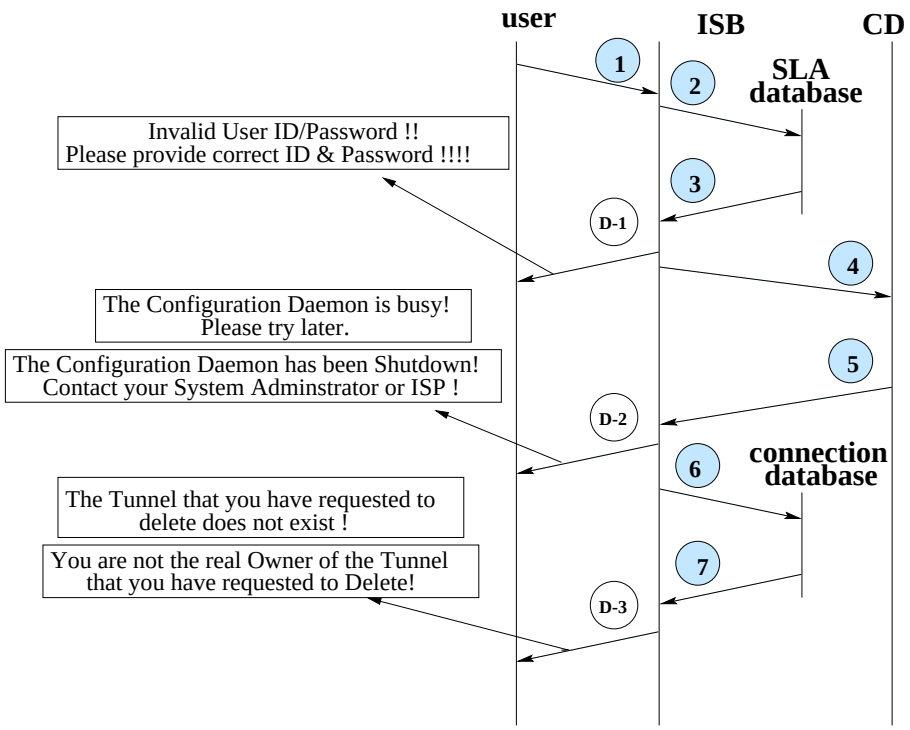


Figure 3.14: Failure in Connection Termination

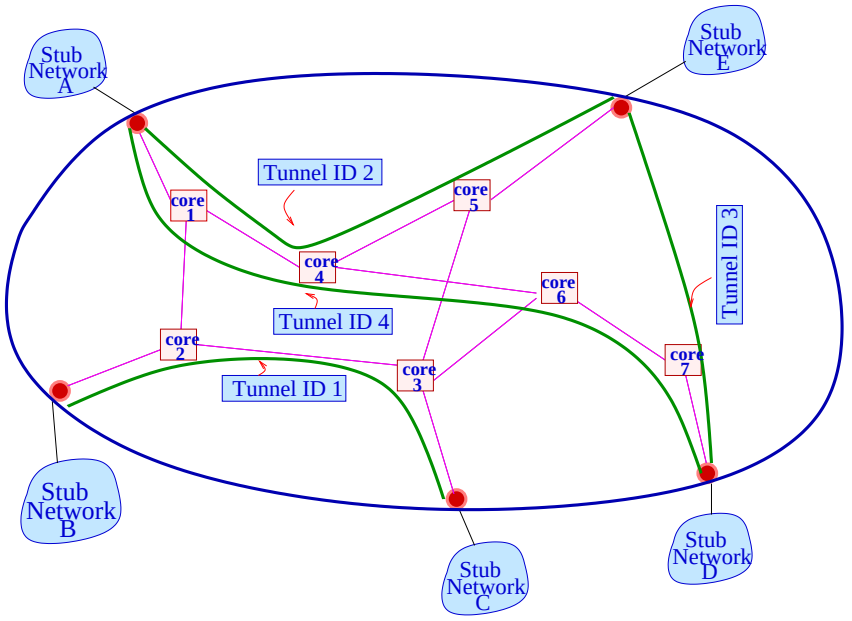
Mbps, 2 Mbps or 3 Mbps dedicated bandwidth) where each service class will provide a different performance. The objective is to allow users to select among a choice of services, so that users who wish to use more resources can pay accordingly.

In our implementation we have provided a method to compute the price of a VPN which considers the reserved bandwidth of the tunnel and the duration it was used. However, that price might change over the duration of an active period, e.g. if we have special tariffs for the day and the night. This actually reflects that price changes as the load changes, i.e. we consider price to be a function of resource and load. A 2 Mbps tunnel that is charged 4 cents per minute during peak period would not be charged the same during off load period. Also a 2 Mbps tunnel that is spanned over several core routers might be priced higher than a tunnel of same capacity but spanning over few routers. In reality, most of the costs will depend on the transmission lines and some of the expensive lines might be consisting of fewer routers than the less expensive lines of same capacity. In such cases tunnel price can be set accordingly. Based on this idea we propose that

pricing for QoS enabled VPN tunnel should be calculated based on its network resource reservation and the load during the time tunnel is active. Therefore,

$$Price \quad = \quad f(Resource, Load)$$

This model works much like the telephone system when one pays more for long distance call than a local call and price changes at different times. As understanding of pricing by the users is crucial for the success of managed VPN services, we believe this model would be attractive not only to the users but also will simplify the billing process for ISPs.



Price Per Minute in cents								
Tunnel ID	00:00 - 01:00	01:00 - 02:00	02:00 - 03:00	06:00- 07:00	07:00- 08:00	08:00- 09:00	09:00 - 10:00	23:00 - 24:00
1	1	1	1	1	1	2	4	1
2	1	1	1	1	2	4	6	1
3	1	1	1	1	1	2	4	1
4	1	1	1	1	5	7	9	1

Figure 3.15: Differential Tunnel Pricing of an Example Network

3.5.2 Differential Tunnel Pricing

As in our system resource is a single quantitative item usually expressed as 1 Mbps, 2 Mbps etc. it well understood how one can price a pipe of varying hop

distances. For example, in Figure 3.15 both tunnel 1 and 2 are of 1 Mbps but tunnel 2 is priced higher at certain times since it spans over 3 core routers, and therefore, consumes more resources than tunnel 1. For the same reason tunnel 4 might be priced higher than tunnel 3 although both are of 2 Mbps. If tunnel 3 is highly congested line (or highly demanded) or its installation cost was higher than normal cases (e.g. for intercontinental links), the ISP might well set the price 3 higher than tunnel 4 even thought it spans over less routers. Therefore, setting up the pricing matrix depends entirely on ISP's experience and business practice. However, having said that, we still need to incorporate load in the pricing formula. Normally, load changes with time and we can, therefore, define a differential tunnel pricing matrix as shown in Figure 3.15 where price changes with time. If $P_i(t)$ denotes the price of tunnel i at time t and T_{in} and T_{out} denotes the tunnel creation time and termination time respectively, then P_{TOT}, the total computed price of the tunnel for the duration (T_{out} - T_{in}), can be expressed as:

$$P_{TOT} = \sum_{T_{in}}^{T_{out}} P_i(t)$$

In our implementation we have defined 24 time zones, i.e. the price for a tunnel changes every hour. It is, however, possible to have more or less than 24 zones. All that is needed to be done in such a case is to define a similar pricing matrix based on the number of desired time zones. Again, how ISP should decide about the number of time zones depends ISP's experience with load fluctuations from past and also on its policy. For example, if the load is frequently changing then a higher number of time zones might be appropriate to reflect the high fluctuations of load on the price.

Based on our pricing model we will now show how one can calculate the price of a tunnel for a certain duration of time. Let us assume that T_{in} and T_{out} have formats like $H1 : m1$ and $H2 : m2$ respectively where $H1$ and $H2$ represent hour portions of time while $m1$ and $m2$ represent the minute portions. If $p_i(h)$ denotes the price of tunnel $i = 1, 2, 3, 4...N$ (if we have N tunnels) for time zone $h = 0, 1, 2, 3, ...23$ etc, then the total price of a VPN tunnel can be defined as:

$$P_{TOT} = \begin{cases} p_i(h).[m2 - m1] & \text{if } M = 0 \\ p_i(h)[60 - m1] + p_i(h+1).m2 & \text{if } M = 1 \\ p_i(h)[60 - m1] + \sum_{h=h+1}^{h+M-1} p_i(h).60 & \\ +p_i(h+M).m2 & \text{if } M \geq 2 \end{cases}$$

where $M = H2 - H1$. We will now provide a few examples using the formulas presented above and differential pricing matrix shown in Figure 3.15. If a certain user has used a tunnel of 1 Mbps from 6:10 a.m. to 6:20 a.m. between stub network B and C and that tunnel happens to have the tunnel ID 1, then the price can be calculated as: $p_1(6)$ *(20-10)=1*10= 10 cents. If the same tunnel is active from 6:10 to 7:20 then the price would be $p_1(6)$ *(60-10) + $p_1(7)$* 20 =1*50 + 1*20 =70 cents. Again, if we setup a tunnel having ID 4 between stub network A and D from 6:30 to 9:20, then we can calculate the price as $p_4(6)$ *(60-10) + $p_4(7)$* 60 + $p_4(8)$* 60 + $p_4(9)$* 20 = 1* 50 + 5 *60 + 7* 60 + 9*20 =950 cents.

3.5.3 Billing

In section 3.4.2, referring to Figure 3.12, we explained that when a VPN connection is terminated, the SB invokes the pricing database to compute the price and send the record to the customer's billing database. In brief, the complete charging system works as follows: if a new VPN request is accepted then that connection along with the login time is recorded in the connection database. After a certain period when the user disconnects his VPN tunnel that entry is deleted from the connection database. The SB then looks up the rate for the deleted tunnel from the pricing database, computes its price using the formulas in above section 3.5.2 for the duration of activation and then adds deleted connection record along with logout time stamp and the computed price to the billing database. The user can also, at any time, ask the system to query the most recent billing.

In section 3.6.2 we will see that user *catispp* establishes a 1 Mbps connection between source 172.18.0.100 and destination 172.17.0.103 at time 6:17:20. Now, if the connection is terminated at 9:23:14 then the SB deletes the connection entry from the connection database as shown in table 3.6, invokes the pricing database of table 3.4, and calculates the price using the method that we have earlier described. Once that is done, the computed total price, which is 606.5 cents, termination time 6:17:20, and deleted connection entry are added to customer's billing database as

shown in table 3.7.

3.6 QoS-VPN Testbed and Performance Results

To test our implementation of the broker system and its capabilities to setup VPN tunnels and allocate QoS to the established tunnels we ran some experiments in our campus network between two private subnets and also over the public SWITCH [SWI] network between Bern and Geneva. The topology we used is shown in Figure 3.7 which we also used for explaining the QoS-VPN configuration example in section 3.3

We have three private networks with closed user group properties. The machine 172.18.0.100 is a web server containing some pre-recorded MPEG-I streams and is located in a Private network 172.18.0.0 at the University of Geneva. Both 172.20.0.100 and 172.17.0.103 (also a web server) are in two different networks at the University of Berne. All the machines are connected to routers having public IP addresses. Since machines with private addresses are not able to talk to each other over public network unless they communicate over tunnels, this setup is useful to demonstrate that tunnels are really created when a registered user sends a requests to the Broker via the web interface. The Broker runs on a machine 130.92.66.22 and communicates with the routers interfaces that have public IP addresses.

The routers 129.194.90.20 and 130.92.66.141 are both Cisco 26xx while 130.92.70.101 is a 7206 router. All the routers are IPsec and QoS capable. The hop distances between 129.194.90.20 and 130.92.70.101 is 10, between 129.194.90.20 and 130.92.66.141 is 9 and between 130.92.66.141 and 130.92.70.101 is 2. Ideally, to create Virtual Leased Line (VLL) type service the interior routers between the edges should also be DiffServ capable and be able to protect the traffic that are marked as EF at the edges by using CBQ or WFQ. However, since we did not have control over all these routers, we restricted the experiments to tunnel creation and policing/shaping at the edge routers over which we have complete control.

Although we had around 1.5 Mbps between the router in Geneva and routers in Bern during the daytime, we had ample of capacity between the routers in Bern.

Tunnel ID	Ingress Router	Tunnel Source Address	Egress Router	Tunnel Destination Address	Bandwidth in Mbps	Status
140	130.92.70.101	130.92.70.101	129.194.90.20	129.194.90.20	1	1
141	130.92.70.101	130.92.70.101	129.194.90.20	129.194.90.20	2	1
142	130.92.70.101	130.92.70.101	130.92.66.141	130.92.66.141	1	1
143	130.92.70.101	130.92.70.101	130.92.66.141	130.92.66.141	2	2
144	130.92.66.141	130.92.66.141	129.194.90.20	129.194.90.20	1	2
145	130.92.66.141	130.92.66.141	129.194.90.20	129.194.90.20	2	2

Table 3.2: Resource Database for the Test Network

Stub Network	Edge Router	Generic Name of Router	Inbound Interface	Outbound Interface	Tunnel Map Name
172.17.0.0	130.92.70.101	Goppenstein	FastEthernet1/0	FastEthernet0/0	cati-tunnel
172.20.0.0	130.92.66.141	sarah	FastEthernet0/1	FastEthernet0/0	cati-tunnel
172.18.0.0	129.194.90.20	Appolo	FastEthernet0/1	FastEthernet0/0	genbern

Table 3.3: Interface Database for the Test Network

3.6.1 Setting up SB Databases for the Test Network

To give a clear idea how the Service Broker works we will show how we setup the various databases that we discussed in section 3.3. Initially, we only need to setup the interface, resource, SLA and pricing databases which are invoked during establishment or termination of a QoS-VPN tunnel. Tables 3.2, 3.3, 3.5, 3.4 show the partial database entries for the test network.

3.6.2 Examples of Connection Setup

Consider an example when user catispp wants to establish a 1Mbps a tunnel between hosts 172.18.0.100 and 172.17.0.103 which are in two different private networks. Once he submits his request via the WWW interface the SB checks SLA validity, daemon status and connection database. While checking the SLA database (Table 3.5) it finds that catispp is a valid user (and password is correct), source and remote stub addresses are valid, and requested rate (1 Mbps) is less than the maximum contracted rate (4 Mbps). Assume that the configuration daemon is on (i.e status is 1). As no such connection (i.e. between

Tunnel ID	00:00 - 1:00	1:00- 2:00	...	6:00 - 7:00	7:00 - 8:00	8:00- 9:00	9:00 - 10:00	...	23:00 - 24:00
140	1	1	...	1.5	3.5	4	4	...	1
141	1	1	...	2	4.5	5.5	5.5	...	1
142	1	1	...	1.5	2	2.5	2.5	...	1
143	1	1	...	2	3	3.5	3.5	...	1
144	1	1	...	1.5	3.5	4	4	...	1
145	1	1	...	2	4.5	5.5	5.5	...	1

Table 3.4: Tunnel Pricing Matrix for the Test Network

User ID	Password	Maximum BW in Mbps	Source Stub Address	Remote Stub Addresses
catispp	********	4	172.18.0.100 172.17.0.103	172.20.0.103 172.17.0.103

Table 3.5: SLA for User *catispp*

User ID	Source Address	Destination Address	Bandwidth in Mbps	Tunnel ID	Activation Time
catispp	172.18.0.100	172.17.0.103	1	140	6:17:20

Table 3.6: Connection Database of the Test Network

source 172.18.0.100 and destination 172.17.0.103) record exists in the connection database prior to this request arrival, the SB now searches the resource database for the availability of a 1 Mbps tunnel. Since the private networks 172.18.0.0 and 172.17.0.0 are connected to routers Appolo (ip address 129.194.90.20) and Goppenstein (ip address 130.92.70.101) respectively, the Broker actually looks for a 1 Mbps tunnel between these two routers. It turns out that tunnel 140 is the exact match and is also available. Therefore, appropriate configuration scripts are created and loaded to the router to establish the requested tunnel.

3.6.3 Performance Results

This section describes the performance of various QoS enabled tunnels that are setup by our Service Broker. The main intention here is not to find out the influence of buffer size or burst length on shaping or policing algorithm to illustrate performance changes, but rather to show the readers that such QoS mechanisms work with VPN tunnels established through our managed system.

For the demonstration of performance, we played some MPEG-I streams over various VPN tunnels. We selected three public domain bit-streams that we believe constitute a reasonable data set. Table 3.8 presents the characteristics of the bit-streams. Here, the three bit streams have different bit rate requirements.We believe the best metric to judge the performance of the QoS VPN tunnels is to measure to what extent the required bit rate is achieved while playing over those tunnels in real time.

User ID	Source Address	Destination Address	Bandwidth in Mbps	Tunnel ID	Activation Time	Termination Time	Price in Cents
catispp	172.18.0.100	172.17.0.103	1	140	6:17:20	9:23:14	606.5

Table 3.7: Billing Database for user *catispp*

Stream	Stream Size (bytes)	Frame Size	Frames/ Second	Bit Rate (Per Sec.) only video	Mux Rate (Per Sec) with audio	I:P:B Ratio
northamerica	31096832	352x240	29.97	1.008	1.2184	18:31:51
heuris	20595900	352x240	29.97	1.152	1.4112	15:44:41
sayit	78021332	352x240	29.97	1.856	2.4576	15:41:42

Table 3.8: Sample MPEG-I Bitstreams

Figure 3.16(a), 3.17(a), 3.18(a) show the bit rate distributions of the MPEG streams of *northamerica*, *heuris* and *sayit*. These rate consider the video frames only although the overall rates are higher. We first established a 1.25 Mbps VPN tunnel between routers Goppenstein and sarah for the web server (source) 172.17.0.103 and station 172.20.0.100 running MpegTV [MPE] player and played stream *northamerica* over that tunnel. As we can see from Table 3.8 that *northamerica* requires a rate 1.2184 Mbps, the trace of the received traffic as shown in Figure 3.16(b) demonstrates that capacity allocation was adequate to transmit this stream smoothly. In another set of experiments we again created 1.5Mbps and 2.5 Mbps VPN tunnel and ran *heuris* and *sayit* respectively. Both of these streams require 1.411 Mbps and 2.457 Mbps respectively. Output traces as shown in Figure 3.17(b) and 3.18 (b) again prove that allocation was sufficient.

Again, *sayit* was run from the web server 172.18.0.100 to station 172.20.0.100 over a tunnel between routers Appolo and sarah without any QoS enabled to it . Figure 3.19 shows the corresponding output trace. Since *sayit* requires 2.457 Mbps and no capacity was specifically allocated to the tunnel the stream ran at a slow frame rate. When at around 175th second some udp traffic was sent over the tunnel to fill up the pipe, this aggressive udp traffic completely stopped the transmission of the MPEG-I stream. When the udp transmission was stopped the stream again started running but still at a much slower rate than actually required for it. This further demonstrate the need for QoS in VPN tunnels and also show that such QoS mechanisms can work with various tunneling methods.

3.7 Conclusion

The proposed policy-based Service Broker architecture and its implementation allows not only network administrators, but also corporate customers to establish and terminate QoS enabled VPN tunnels dynamically on demand in a point-and-click fashion. We have built a system that can cope with the growing network size and complexity and able to operate in a multi-vendor environment.

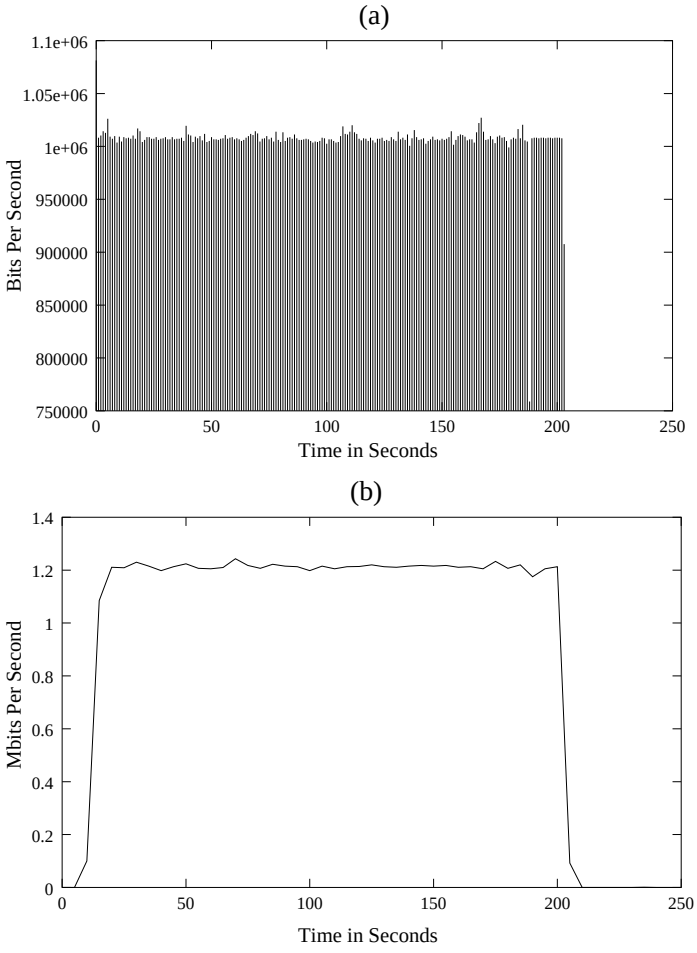


Figure 3.16: (a) Bit Rate Distribution of Video Frames Only for *northamerica*, (b) Output Traces over a 1.25 Mbps Tunnel

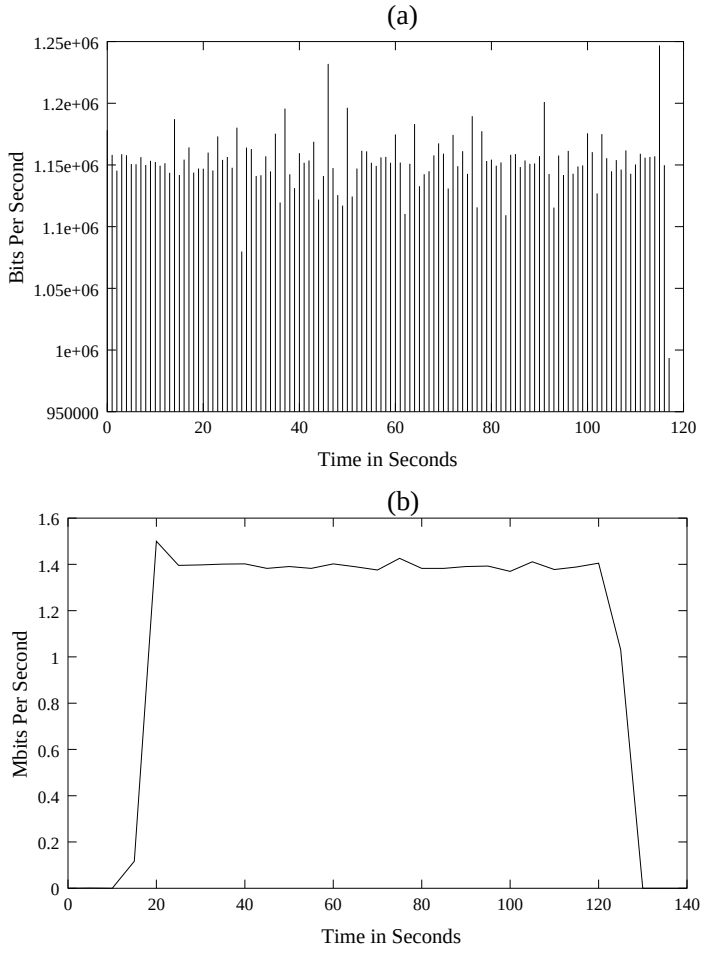


Figure 3.17: (a) Bit Rate Distribution of Video Frames Only for *heuris*, (b) Output Traces over a 1.50 Mbps Tunnel

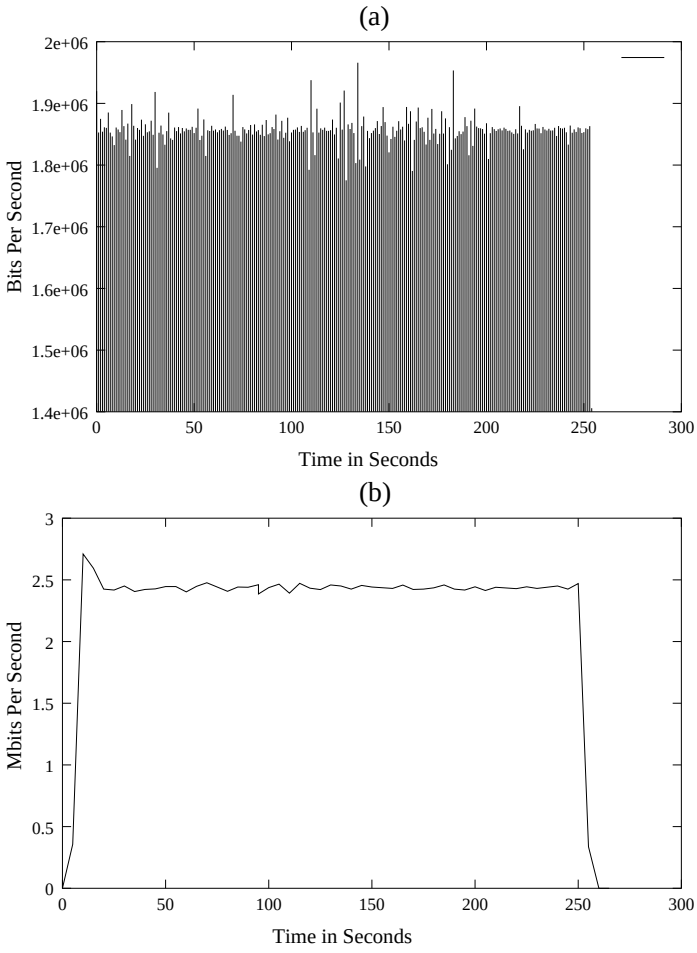


Figure 3.18: (a) Bit Rate Distribution of Video Frames Only for *sayit* , (b) Output Traces over a 2.5 Mbps Tunnel

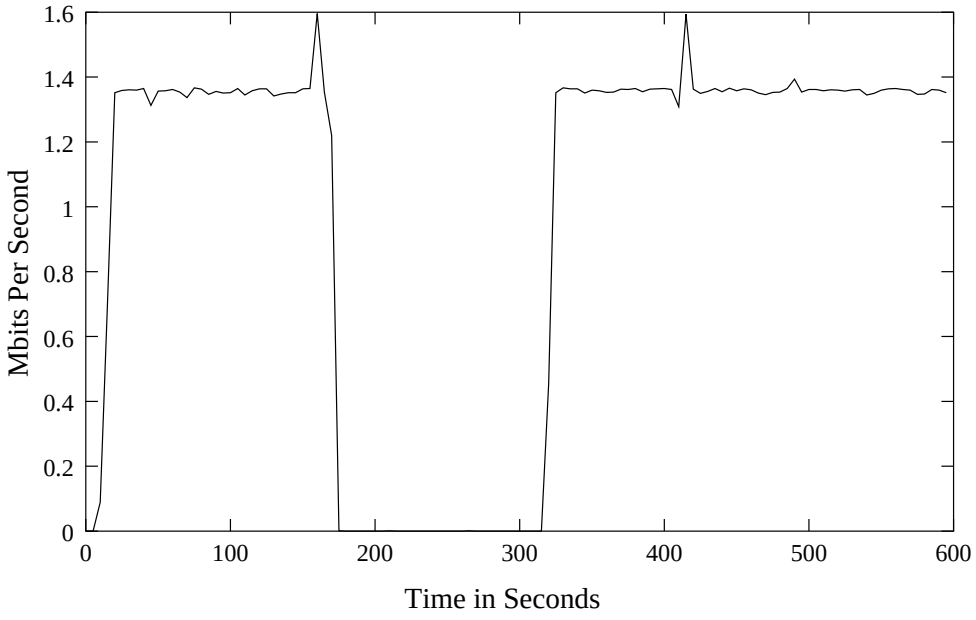


Figure 3.19: Transmission of *sayit* During a Period of Congestion

We realized that service differentiation will not be successful unless there is a pricing mechanism that appreciates the differential behaviour. Therefore, we also proposed and implemented a differential tunnel pricing in Service Broker system that computes the price of a QoS enabled VPN tunnel based on its network resource reservation and the load during the time tunnel is active. From this perspective we believe that if the Service Broker system is deployed by ISPs that would not only alleviate the pain of corporate administrators who often need human resources and huge amount of time in such complex implementations, but also benefit the service providers form the economic point of view.

However, the system lacks advanced SLA mechanism and operate only in a single ISP domain. These issues are addressed in chapter 4, 5 and 6.

Chapter 4

Range-Based SLA and Edge Provisioning

The Interface to the Service Broker described in the previous chapter allows a user to specify bandwidth as a single quantitative value for a VPN tunnel. However, expectedly, many of the customers will be unable or unwilling to predict the load between VPN endpoints. In this chapter we propose a novel range-based SLA that allows VPN customers to specify their bandwidth requirements as a range of quantitative service in the Service Level Agreements (SLAs). To support such services we enhance the Service Broker implemented in chapter 3 that can logically partition the capacity at the edges to various classes (or groups) of VPN connections and manage them efficiently to allow resource sharing among the groups in a dynamic and fair manner. Various algorithms with examples and analyses are presented to provision and allocate resources dynamically at the edges for VPN connections.

4.1 Introduction

To provide Virtual Leased Line (VLL) type point-to-point connection for VPN using Expedited Forwarding (EF) [JNP99] Per Hop Behavior (PHB) we in [KB00a], [BGK01] [KBG00], along with others [QBO00],[Tea99]), have implemented Bandwidth Brokers [NJZ99] that allow users to specify a single quantitative value (i.e. 1 Mbps or 2 Mbps etc.) and based on this specification the edge routers establish

VPN connections dynamically. However, it is apprehended that users will be unable or unwilling to predict the load between VPN endpoints [DGG+99]. Also, from the provider's point of view, guaranteeing exact quantitative services might be a difficult job at the beginning of VPN-DiffServ deployment [BBC+99]. We, therefore, propose that users specify their requirements as a range of quantitative services. For example, users who want to establish VPN connections between stub networks A and B (Figure 4.1), and are not sure whether 0.5 Mbps, 0.6 Mbps or 1 Mbps are needed, and only know the lower and upper bounds of their requirements approximately, can specify a range 0.5- 1 Mbps when they outsource their services to the ISPs. An ISP can offer such multiple options via a web front-end (Figure 4.5) to help customers to select any suitable option to activate services dynamically on the fly.

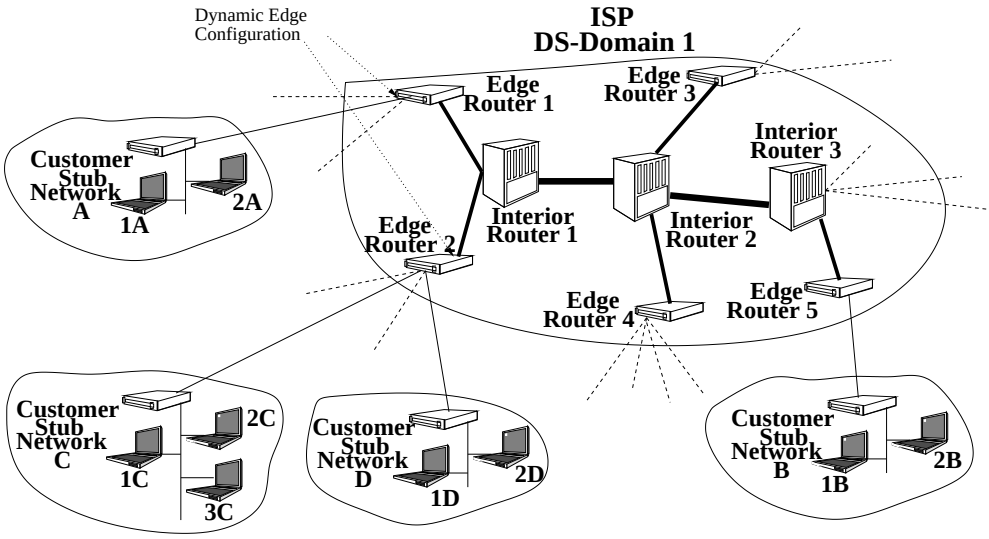


Figure 4.1: Dynamic Edge Configuration of VPN Connections

This approach has several advantages. Users do not need to specify the exact capacity, but it gives them the flexibility to specify only a range. The price that customers have to pay is higher than one pays for the lower-bound capacity but lower than what is normally needed to be paid for the upper-bound capacity. During low load it is possible that the users might enjoy the upper-bound rate (say, 1 Mbps when a range 0.5-1 Mbps is chosen) without paying anything extra. This kind of pricing might be attractive to the users and the ISPs can take advantage of this to attract more customers without breaking the commitment.

This, however, poses a significant challenge to the ISPs, as they would need to

deploy automated provisioning systems that can logically partition the capacity at the edges to various classes or groups of VPN connections and manage them efficiently to allow resource sharing among the groups in a dynamic and fair manner. Here, each group is identified from what it offers. For example, one group could represent the range 0.5- 1 Mbps, another 1-2 Mbps. Also, they must provision the interior nodes in the network to meet the assurance offered at the boundaries of the network. Earlier in chapter 3 we proposed a two-layered model to provision such VPN-DiffServ Networks where the top layer is responsible for edge provisioning and drives the lower layer in charge of interior resource provisioning with the help of a Service Broker. This chapter addresses the top layer on the basis of range-based SLA.

We have restricted this chapter only to edge provisioning because most of the complexities lie at the boundaries of the network and work as the main driving force for overall provisioning. Section 4.2 describes range-based SLA and the model for provisioning, and in section 4.3 various algorithms with examples and analyses have been presented to provision and allocate resources dynamically at the edges. Fairness issues while allocating resources to connections of the various VPN groups have been addressed in section 4.3.4. The enhanced SB performing the required provisioning and connection admission is described in section 4.4. Section 4.5 concludes the chapter with a summary of our contributions.

4.2 Edge Provisioning Model for DiffServ-VPNs

Since edge provisioning is the main driving force of overall resource provisioning in DiffServ networks, it naturally involves complicated procedures and algorithms to dynamically share nodal resources. This becomes even more complicated to support range-based that is able to benefit both customers and service providers economically. In this section we give an overview of the concept of range-based SLA and explain the proposed edge provisioning model.

4.2.1 A Novel Approach: Range-Based SLA

To overcome the difficulties faced by users in specifying the exact amount of quantitative bandwidth required while outsourcing the VPN service to ISPs, our model supports a flexible way to express SLAs where a range of quantitative amounts, rather than a single value, can be specified. Although it has several

advantages, this also makes the edge and the interior provisioning difficult. This complexity can be explained with a simple example. Referring to Figure 4.1, assume that the edge router $R2$ has been provisioned to provide 20 Mbps quantitative resources to establish VPN connections elsewhere in the network with the ISP providing two options via a web interface to the VPN customers to select the rate of the connections dynamically: 1 Mbps or 2 Mbps. It is easy to see that at any time there can be 20 connections each having 1 Mbps, or 10 connections each enjoying 2 Mbps, or even a mixture of the two (e.g., 5 connections with 2 Mbps, 10 connections with 1 Mbps). When a new connection is accepted or an active connection terminates, maintaining the network state is simple and does not cause either reductions or force any re-negotiations to existing connections. If there are 20 connections of 1 Mbps and one connection leaves, then there will be simply 19 connections of 1 Mbps. Admission process is equally simple.

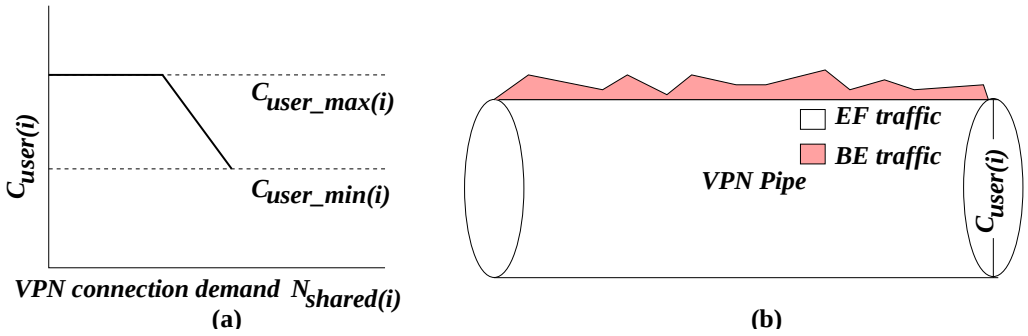


Figure 4.2: The Range-Based SLA Approach

Now, if the ISP provides a new option (Figure 4.5) allowing users to select the range 1-2 Mbps, where 1 and 2 are the minimum and maximum offered guaranteed bandwidth, maintaining the state and admission control can be difficult. When there are up to 10 users, each connection would get the maximum rate of 2 Mbps, but as new connections start arriving, the rate of the existing connections would decrease. For example, when there are 20 connections this rate would be $\frac{20}{20} = 1$ Mbps. At this stage, if an active connection terminates, the rate of every single connection would be expanded from 1 Mbps to $\frac{20}{19} = 1.05$ Mbps. This is a simple case when we have a single resource group supporting the range 1Mbps-2 Mbps. In reality, we might have several such groups to support users requiring varying bandwidth. In such cases, renegotiation for possible expansion of the existing connections, admission control, and maintenance of network states will not be simple. Figure 4.2 illustrates the idea of range-based SLA. Bandwidth

is specified as an interval of $C_{user\_min(i)}$ and $C_{user\_min(i)}$ for any group i. The actual rate of a VPN connection $C_{user(i)}$ varies between this range but never gets below $C_{user\_min(i)}$. $C_{user(i)}$ is the rate that is configured in the edge router as the policing rate. Traffic submitted at a rate higher than this is marked as best effort traffic or dropped depending on the policy.

4.2.2 The Model and Notations

In our model, we address this novel SLA approach and provide policies and algorithms for automated resource provisioning and admission control. However, to support such provisioning we first start by allocating a certain percentage of resources at each node (edge and interior) to accommodate quantitative traffic. At the edge, this quantitative portion is further logically divided between dedicated VPN tunnels (i.e. require 1Mbps or 2 Mbps explicitly) and those connections that wish to have rates defined by a range (i.e. 0.5-1 Mbps or 1-2 Mbps, etc.). Figure 4.3 shows this top level bandwidth apportionment. The notations are :

- C_T is the total capacity of a node interface.
- C_{ded} is the capacity to be allocated to VPN connections requiring absolute dedicated service.
- C_{shared} is the capacity apportioned for VPN connections describing their requirement as a range.
- C_{quan} is the capacity provisioned for quantitative traffic and is equal to (C_{ded}+ C_{shared}).
- C_{qual} is the remaining capacity for qualitative traffic.

While at the edge C_{quan} is the rate controlled by policing or shaping, at the interior this C_{quan} indicates the amount capacity allocated (actually protected) to quantitative traffic. All the values can be different at different nodes. This kind of logical partitioning is helpful because the capacity is never wasted even if portions of resources allocated to quantitative traffic are not used by VPN connections. The unused capacity naturally goes to the qualitative portion and enhances the best effort and other qualitative services. This is true at both the edge and in the interiors. C_{shared}, as shown in Figure 4.3, can be logically divided into multiple groups where each group supports a different range (Figure 4.4).

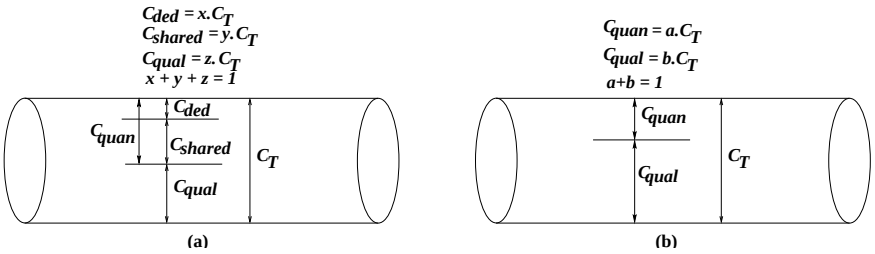


Figure 4.3: Top Level Bandwidth Apportionment: (a) Logical Partitioning at the Edge, (b) Logical Partitioning at an Interior

As there might be multiple of such groups, for any group i we define the following notations:

- $C_{base(i)}$ is the the base capacity for group i which is shared by the VPN connections belonging to that group.
- $C_{user\_min(i)}$ is the ISP offered minimum guaranteed bandwidth that a user can have for a VPN connection.
- $C_{user\_max(i)}$ is the ISP offered maximum guaranteed bandwidth that a user can have for a VPN connection.
- $N_{shared(i)}$ is the current number of shared VPN connections in group i.
- $C_{shared(i)}$ is the amount of capacity currently used by group i.
- $C_{user(i)}$ is the actual rate of active connections in group i and is equal to $\frac{C_{shared(i)}}{N_{shared(i)}}$.
- C_{shared_unused} is the total unused bandwidth from all shared service groups.

We can apply numerous sharing policies to these shared service groups. We call them shared service groups because, in reality, the base capacity is shared by a certain number of VPN connections. A sharing policy might allow a group to share its resources not only among its own connections, but also share with other groups' VPN connections in case of some unused capacity left. This may also apply to dedicated capacity. Priority can be given to certain groups while allocating unused resources. Actually, fair sharing is a challenging problem and we will address all these issues in the following sections while developing the provisioning mechanisms.

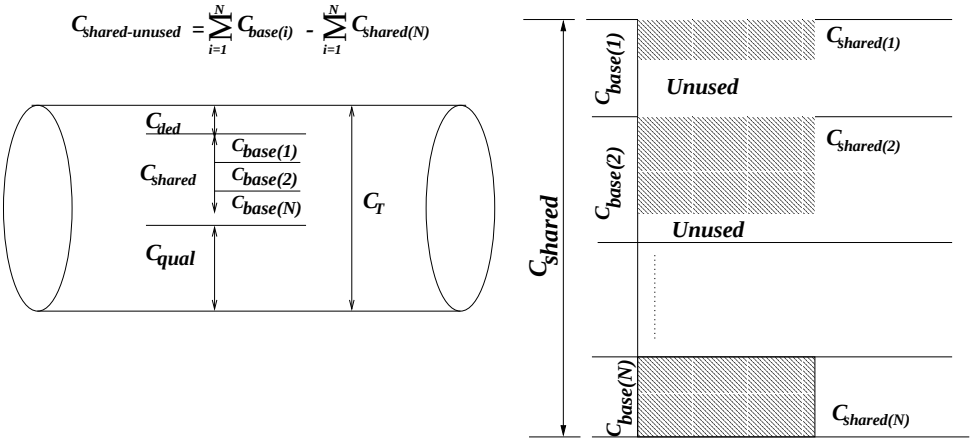


Figure 4.4: Microscopic View of Bandwidth Apportionment at Edge

4.3 Edge Provisioning Policies and Algorithms

Based on the model described in section 4.2, various allocation policies could be adopted by the ISPs at the ingress point to allocate capacity dynamically to maintain and guarantee the quality of service of various types of incoming and existing VPN connections from multiple classes of VPNs. Some suitable policies are :

- **Policy I**: Capacity unused by one group cannot be used by any other groups. This means that if we have multiple shared service groups, the group whose resources have been exhausted while supporting numerous connections does not borrow resources from others even when they have unused capacity. Also, none of the groups are allowed to use unused capacity of the dedicated service group.
- **Policy II**: Capacity unused by one shared service group can be borrowed by another shared service group. However, like the previous policy, they are not supposed to borrow from the dedicated service group.
- **Policy III**: Capacity unused by the dedicated service group can be borrowed by tunnels of the shared service groups. Also, these groups can share resources among themselves.

In this section, we will start with VPN connection acceptance algorithms at the network ingress point where all admission complexities lie. These complexities

are introduced because of the need to partition and share resources to support our model and the policies presented above. Further analyses with examples of algorithms for Policy I,II and III clarify them.

4.3.1 VPN Connection Acceptance at Ingress

The job of admission control is to determine whether a VPN connection request is accepted or rejected. If the request is accepted, the required resources must be guaranteed. For any group i a new VPN establishment request is admitted only if the minimum bandwidth, as stated in the offer, can be satisfied while also retaining at least the minimum requirements for the existing users, i.e. if $\left(N_{shared(i)} \leq \frac{C_{base(i)}}{C_{user\_min(i)}}\right)$ a new VPN connection request can be accepted. This ensures that, an admitted VPN connection will always receive at least the minimum offered bandwidth $C_{user\_min(i)}$ in group i by restricting the number of maximum connections that can join the group. How much capacity the accepted connection will actually hold is decided by the connection states in that group and sharing policies that we are going to discuss in the following subsections.

4.3.2 Capacity Allocation with no Sharing: Policy I

The base capacity allocated to a group is solely used by the VPN connections belonging to that group only. Under no circumstances resources assigned to one group can be borrowed by others, even if that capacity remains unused. This makes allocation simple not only at the edges, but also in the interior. Also, from an implementation point of view it is simple. Since the unused capacity is not used by any other groups, the qualitative services mentioned earlier are also enhanced.

If a VPN connection is accepted, the system checks whether that connection can be allocated the maximum rate. This is possible if the base capacity $C_{base(i)}$ is enough to assign all the existing connections the maximum rate $C_{user\_max(i)}$. Otherwise, the base capacity is shared among all the existing and new VPN connection. Therefore, we can express this admission policy as follows:

$$C_{shared(i)} = min\Big(C_{base(i)}, C_{user\_max(i)}.N_{shared(i)}\Big)$$
$$C_{user(i)} = \frac{C_{shared(i)}}{N_{shared(i)}}$$

Example 1: For the following example assume that the total link bandwidth $C_T = 100$ Mbps, $C_{shared} = 0.3C_T = 30$ Mbps. Also, assume that ISP offers

a group as $C_{user\_min(1)} = 1$ Mbps and $C_{user\_max(1)} = 2$ Mbps. Base capacity $C_{base(1)}$ allocated to this group is 20 Mbps.
$N_{shared(1)} = 1$, $C_{shared(1)} = 2$ Mbps, $C_{user(1)} = 2$ Mbps
.
$N_{shared(1)} = 10$, $C_{shared(1)} = 20$ Mbps, $C_{user(1)} = 2$ Mbps
$N_{shared(1)} = 11$, $C_{shared(1)} = 20$ Mbps, $C_{user(1)} = \frac{20}{11}$ Mbps
.
$N_{shared(1)} = 20$, $C_{shared(1)} = 20$ Mbps, $C_{user(1)} = \frac{20}{20}$ Mbps

Connections are accepted as long as the condition $\left(N_{shared(i)} \leq \frac{C_{base(i)}}{C_{user\_min(i)}}\right)$ is met. When the number of connections exceed $\frac{C_{base(i)}}{C_{user\_min(i)}}$, a new arriving connection is rejected. For example, if the 21st connection in the example is accepted, $C_{user(1)}$ would be $\frac{20}{21}$. The minimum bandwidth could no longer then be guaranteed. Therefore, the connection request is rejected.

4.3.3 Capacity Allocation with Sharing: Policy II

If the capacity allocated to a group is not fully used by VPN connections, this capacity can be borrowed by connections of the other shared service groups, if needed. However, the borrowed capacity must be relinquished when needed by the group from which the capacity was borrowed. And although this borrowing and deallocation adds some complexity in edge provisioning, connections from various groups however have better chances of enjoying higher rates. In the following sections, we present algorithms regarding VPN connection arrival, termination, and possible expansion of the existing connections as a result of the termination of a connection from a shared service group.

VPN Connection Arrival

Like the previous case, VPN connection arrival essentially involves checking the availability of resources that can be used by the new connection and, if available, allocating this capacity to an incoming connection. Even if the base capacity of a certain group allows the new connection belonging to that group to assign the maximum ISP offered rate (i.e. $\left(C_{base(i)} - C_{shared(i)}\right) \geq C_{user\_max(i)}$) because of the resource sharing among various groups, it might happen that the resources from that group would be borrowed by other group(s) not leaving the required

resources (i.e. $C_{shared\_unused} < C_{user\_max(i)}$). In such a case resources must be relinquished from the appropriate groups(s). Any such deallocation from the existing connections leads to rearrangement of capacity of those connections. This capacity should be relinquished the way it was borrowed. The unused capacity can be borrowed numerous ways by competing groups which we will see in sections 4.3.3 and 4.3.4. For the sake of simplicity, the group having the maximum excess bandwidth, $C_{excess(i)} = C_{shared(i)} - C_{base(i)}$ should release first, and then the next, and so on.

/ if the group has enough base capacity to support*
*a new connection with max. offered rate. */*
$if\left[\left(C_{base(i)} - C_{shared(i)}\right) \geq C_{user\_max(i)}\right]$
$\{$
/ if the shared unused capacity is also enough to support*
*the new connection with max. offered rate. See Example 2 */*
$if\left(C_{shared\_unused} \geq C_{user\_max(i)}\right)$
$\{$
$C_{shared(i)} = C_{user\_max(i)}.N_{shared(i)}$
$C_{user(i)} = C_{user\_max(i)}$
$\}$
/ if the shared unused capacity has been borrowed then*
*capacity is relinquished from borrower(s). See Example 3 */*
else
$\{$
relinquish $C_{user\_max(i)}$ *from group(s) which has max excess bw*
rearrange bandwidth of that group(s)
$C_{shared(i)} = C_{user\_max(i)}.N_{shared(i)}$
$C_{user(i)} = C_{user\_max(i)}$
$\}$
$\}$

We have just mentioned that this capacity can be borrowed from one group by the others. Now, when does one group borrow resources? Naturally, when the base capacity is less than what is needed, i.e. $\left(C_{base(i)} - C_{shared(i)}\right) \leq 0$. How much can one group borrow? This depends on how much unused resources are available. If this is at least equal to the maximum offered rate $C_{user\_max(i)}$, then that amount is allocated; otherwise (i.e. $C_{shared\_unused} < C_{user\_max(i)}$), the whole unused resource goes to the group in question and divided among all the

connections in that group.

/ if the shared capacity is equal to or has exceeded the base capacity */*

$if\Big[\Big(C_{base(i)} - C_{shared(i)}\Big) \leq 0\Big]$

$\Big\{$

/ but the unused capacity can still support the new connection with max rate. Capacity is then borrowed. See Example 4 */*

$if\Big(C_{shared\_unused} \geq C_{user\_max(i)}\Big)$

$\Big\{$

$C_{shared(i)} = C_{shared(i)} + C_{user\_max(i)}$

$C_{user(i)} = \frac{C_{shared(i)}}{N_{shared(i)}} = C_{user\_max(i)}$

$\Big\}$

*/*if the unused capacity is less than the max. rate. Capacity is then shared by existing and the new connection. See Example 5 */*

else

$\Big\{$

$C_{shared(i)} = C_{shared(i)} + C_{shared\_unused}$

$C_{user(i)} = \frac{C_{shared(i)}}{N_{shared(i)}}$

$\Big\}$

$\Big\}$

We will now consider several numerical examples in this section to clarify the algorithms and analysis presented above. For all the following examples we assume that the total link bandwidth $C_T = 100$ Mbps, $C_{shared} = 0.3C_T = 30$ Mbps, and there are only two shared users groups i.e. $i = 1, 2$. For group 1 $C_{base(1)} = 10$ Mbps,$C_{user\_min(1)} = 0.5$ Mbps and $C_{user\_max(1)} = 1$ Mbps, and for group 2 $C_{base(2)} = 20$ Mbps, $C_{user\_min(2)} = 1$ Mbps and $C_{user\_max(2)} = 2$ Mbps.

Example 2 : Prior to VPN connection request in group 1:
$N_{shared(1)} = 5, C_{shared(1)} = 5 \times 1 = 5$ Mbps
$N_{shared(2)} = 10, C_{shared(2)} = 10 \times 2 = 20$ Mbps

Here, for group 1, $C_{base(1)} - C_{shared(1)} = 10 - 5 = 5$ Mbps and $C_{user\_max(1)} = 1$ Mbps. Therefore, $C_{base(1)} - C_{shared(1)} > C_{user\_max(1)}$. Also, $C_{shared\_unused} = 30 - (5 + 20) = 5$ Mbps, which is greater than $C_{user\_max(1)}$. Hence, $C_{user(1)} = 1$ Mbps.

Example 3 : Prior to VPN connection request in group 1:
$N_{shared(1)} = 6, C_{shared(1)} = 6 \times 1 = 6$ Mbps

$N_{shared(2)} = 12, C_{shared(2)} = 12 \times 2 = 24$ Mbps

In this example, $C_{base(1)} - C_{shared(1)} = 10 - 6 = 4$ Mbps, which is greater than $C_{user\_max(1)} = 1$ Mbps. This means that group 1 has not used all its base bandwidth and a new connection can have the maximum offered bandwidth 1 Mbps. However, C_{shared_unused} at the time of request arrival is $C_{shared} - \sum_{i=1}^{2} C_{shared(i)} = 30 - (6 + 24) = 0$ Mbps. This indicates that another group has borrowed capacity from group 1. If that group had left at least $C_{user\_max(1)} = 1$ Mbps, the request could have been allocated the desired amount of resource. Therefore, the only option left is to relinquish 1 Mbps from the group that borrowed it. Since the only other group 2 has taken that bandwidth we need to deduct 1 Mbps from group 2 and recompute the individual share of each VPN connection in that group as $C_{user(2)} = \frac{C_{shared(2)} - C_{user\_max(1)}}{N_{shared(2)}} = \frac{24-1}{12} = 23/12$ Mbps. Obviously, $C_{user(1)} = 1$ Mbps and $C_{shared(1)} = 6 + 1 = 7$ Mbps.

Example 4 : Prior to VPN connection request in group 2:
$N_{shared(1)} = 5, C_{shared(1)} = 5 \times 1 = 5$ Mbps
$N_{shared(2)} = 10, C_{shared(2)} = 10 \times 2 = 20$ Mbps

This is a case where one group has used its full allocated base capacity but could borrow resources from the other group which has left some spare capacity. Here, $C_{base(2)} - C_{shared(2)} = 20 - 20 = 0$ Mbps, but the total spared capacity $C_{shared\_unused} = 30 - (5 + 20) = 5$ Mbps. This value is greater than $C_{user\_max(2)}$ (i.e. 2 Mbps). Therefore, the new VPN connection request can be allocated the maximum offered value (i.e. 2 Mbps) by even exceeding the base capacity of group 2.

Example 5 : Prior to VPN connection request in group 2:
$N_{shared(1)} = 8, C_{shared(1)} = 8 \times 1 = 8$ Mbps
$N_{shared(2)} = 11, C_{shared(2)} = 11 \times 2 = 22$ Mbps

The example here depicts a scenario where one group that has already exceeded its base capacity and has to accommodate a new connection request when there is no unused resource left by other group(s). Here, even before the new connection arrival, group 2 has borrowed $C_{shared(2)} - C_{base(2)} = 22 - 20 = 2$ Mbps and $C_{shared\_unused} = 30 - (8 + 22) = 0$ Mbps. So, the current capacity allocated to group 2 will have to be equally distributed among all the existing and the new arriving VPN connections. Therefore, $C_{user(2)} = \frac{C_{shared(2)}}{N_{shared(2)}} = \frac{22}{11+1} = \frac{22}{12}$ Mbps.

VPN Connection Termination

When a VPN connection terminates, the resources might have to be released from the relevant group depending on the current rate being enjoyed by every connection in that group. If the rate is less than or equal to the maximum offered rate, no capacity is released from the group's current share. As a result, all the connections in that group will increase equally. This is because the same capacity is shared by a lower number of connections. If, however, the current rate of every connection is already equal to the maximum offered rate, this termination would trigger a deduction of $C_{user\_max(i)}$ from the shared resource $C_{shared(i)}$. If all the connections were already enjoying $C_{user\_max(i)}$, no rate change would occur in any of the existing connections. The algorithm is stated as follows:

$if\Big(\frac{C_{shared(i)}}{N_{shared(i)}} \leq C_{user\_max(i)}\Big)$ */*See Example 6 */*

$\Big\{$

$C_{shared(i)} = C_{shared(i)}$

$C_{user(i)} = \frac{C_{shared(i)}}{N_{shared(i)}}$

$C_{shared\_unused} = C_{shared\_unused}$

$\Big\}$

$if\Big(\frac{C_{shared(i)}}{N_{shared(i)}} = C_{user\_max(i)}\Big)$ */* Example 7 */*

$\Big\{$

$C_{shared(i)} = C_{shared(i)} - C_{user\_max(i)}$

$C_{user(i)} = \frac{C_{shared(i)}}{N_{shared(i)}} = C_{user\_max(i)}$

$C_{shared\_unused} = C_{shared\_unused} + C_{user\_max(i)}$

$\Big\}$

To clarify the VPN connection, the termination process will now consider similar examples as presented in the previous section.

Example 6: Before VPN connection termination from group 1:
$N_{shared(1)} = 11$, $C_{shared(1)} = 10$ Mbps
$N_{shared(2)} = 10$, $C_{shared(2)} = 20$ Mbps

Here, $\frac{C_{shared(1)}}{N_{shared(1)}} < C_{user\_max(1)}$ since $\frac{10}{11} < 1$. This means that the capacity used by this group before the connection termination will remain unchanged even after the termination. So, the new value of $C_{shared(1)}$ is also 10 Mbps, and each VPN connection will equally share this capacity which is $\frac{C_{shared(1)}}{N_{shared(1)}} = \frac{10}{10} = 1$ Mbps.

Since no capacity is deducted from this group, the total unused shared capacity will also remain unchanged.

Example 7: Before VPN connection departure from group 1:
$N_{shared(1)} = 10$, $C_{shared(1)} = 10$ Mbps
$N_{shared(2)} = 10$, $C_{shared(2)} = 20$ Mbps

In this example, $\frac{C_{shared(1)}}{N_{shared(1)}} = C_{user\_max(1)}$ since $\frac{10}{10} = 1$. Thus, prior to this departure all active VPN connections were using the maximum possible offered bandwidth $C_{user\_max(1)} = 1$ Mbps and in total were having $C_{shared(1)} = 1 \times 10 = 10$ Mbps. Hence, the departure should trigger a deduction of $C_{user\_max(1)} = 1$ Mbps from the total capacity used by this group prior to the departure as the capacity even after the deduction will be good enough to satisfy $N_{shared(1)} = 10 - 1 = 9$ active connections offering the highest possible rate of 1 Mbps. Therefore, $C_{shared(1)} = 10 - 1 = 9$ Mbps and each VPN connection will receive $\frac{C_{shared(1)}}{N_{shared(1)}} = \frac{9}{9} = 1$ Mbps. Since the termination process triggers deduction of $C_{user\_max(1)}$ from the capacity used by group 1, the unused shared capacity will increase by the same value. So, $C_{shared\_unused} = 0 + 1 = 1$ Mbps.

VPN Capacity Expansion

The unused shared capacity left by some groups can be distributed among others with priority given to certain groups while allocating the unused capacity. In the next section we will present various policies to allocate the unused dedicated capacity, and those might apply here as well. Here we consider only one case where preference is given to the needy groups where the need is determined from the ratio $\frac{C_{user(i)}}{C_{user\_max(i)}}$. So, we reorder the groups according to this ratio so that the first one has the lowest and the last one has the highest value of $\frac{C_{user(i)}}{C_{user\_max(i)}}$. Once reordered, the expansion algorithm starts allocating unused bandwidth to the first group, then the next, and so on based on the availability of resources. This can be stated as :

$$if\Big(\frac{C_{shared(i)}+C_{shared\_unused}}{N_{shared(i)}} > C_{user\_max(i)}\Big)$$
$$\Big\{$$
$$C_{shared(i)} = N_{shared(i)}.C_{user\_max(i)}$$
$$C_{user(i)} = \frac{C_{shared(i)}}{N_{shared(i)}}$$
$$C_{shared\_unused(i)} =$$
$$C_{shared\_unused} - [N_{shared(i)}.C_{user\_max(i)} - C_{shared(i)}]$$
$$\Big\}\ /*\ See\ Example\ 8\ */$$

$$if\Big(\frac{C_{shared(i)}+C_{shared\_unused}}{N_{shared(i)}} \leq C_{user\_max(i)}\Big)$$
$$\Big\{$$
$$C_{shared(i)} = C_{shared(i)} + C_{shared\_unused}$$
$$C_{user(i)} = \frac{C_{shared(i)}}{N_{shared(i)}}$$
$$C_{shared\_unused} = 0$$
$$\Big\}\ /*\ See\ Example\ 9\ */$$

Example 8: Before VPN connection termination from group 2:
$N_{shared(1)} = 11$, $C_{shared(1)} = 10$ Mbps
$N_{shared(2)} = 10$, $C_{shared(2)} = 20$ Mbps

After the termination of a VPN connection from group 2, $C_{shared\_unused} = 2$ Mbps. If there is a need of resources by other group(s), this capacity can be used partly or fully. We find that group 1 has need for this resource since $\frac{C_{user(1)}}{N_{user\_max(1)}} < 1$. Now, it remains to be seen to what extent we could use this unused capacity. Here, $\frac{C_{shared(1)}+C_{shared\_unused}}{N_{shared(1)}} = \frac{10+2}{11} = \frac{12}{11}$ and is greater than $C_{user\_max(1)}$ which is 1 Mbps. Therefore, the capacity for group 1 can be expanded to $N_{shared(1)}.C_{user\_max(1)} = 11 \times 1 = 11$ Mbps allocating to each existing connection $C_{user\_max(1)} = 1$ Mbps. The remaining unused capacity will be reduced to $C_{shared\_unused} - [N_{shared(1)}.C_{user\_max(1)} - C_{shared(1)}] = 2 - (11 \times 1 - 10) = 1$ Mbps.

Example 9: Before VPN connection departure from group 2:
$N_{shared(1)} = 14$, $C_{shared(1)} = 10$ Mbps
$N_{shared(2)} = 10$, $C_{shared(2)} = 20$ Mbps

Unlike the previous example where group 1 only needed to use a portion of the unused resources, all the remaining capacity can be allocated to the existing group 1's VPN connections in order to enhance the service. $C_{shared(1)}$ will be increased to $10+2 = 12$ Mbps with each existing connection receiving $\frac{C_{shared(1)}}{N_{shared(1)}} = \frac{12}{14}$ Mbps.

4.3.4 Fair Allocation of Unused Resources: Policy III

In the previous section we discussed sharing methods where one shared service group could borrow resources from another similar group. In this section, we will discuss the possibilities of sharing the unused dedicated resources among various shared service groups. If the shared service groups are allowed to borrow resources from the unused dedicated resources, we then define a new term:

$$C^{+}_{shared} = C_{shared} + C_{ded\_unused}$$

The question here is how we can allocate the unused dedicated resources fairly among the competing groups. If all VPN tunnels want the maximum bandwidth as offered in ISP policy offer, it is possible that at some point:

$$\sum_{i=1}^{N} N_{shared(i)}.C_{user\_max(i)} > C^{+}_{shared}$$

If $\left[\sum_{i=1}^{N} N_{shared(i)}.C_{user\_max(i)} - C^{+}_{shared}\right]$, the quantity needed to allocate the maximum possible offered rates to all connections even after allowing the unused dedicated resources to be used by the shared service groups is greater than 0, we need to define a fair set of user throughput values (i.e. $C_{user(i)}$) given the set of the maximum offered rates $C_{user\_max(i)}$ and C^{+}_{shared}. In other words, we need to basically divide this extra capacity C_{ded_unused} among all the needy groups in a fair manner. However, fair sharing of extra resources is not a trivial issue and was addressed by others for different network situations [ZC93],[Jaf81], [Wd89],[WSF82]. Some proposals [Jaf81] are in favor of sharing the bottleneck capacity equally among users independent of their requirements and others [ZC93],[Wd89] advocate to penalize users causing overloads.

While we do share the resources among VPN connections in each group, equal sharing of unused dedicated capacity will not help much to some groups where connections are already enjoying rates close to $C_{user\_max(i)}$. At the same time, it also does not alleviate the problem of other groups having rates above $C_{user\_min(i)}$ but much less than $C_{user\_max(i)}$. The fairness criterion of [ZC93] also does not fit here as that would deprive the heavy user groups to gain share from the unused dedicated resources even when they are enjoying rates much below $C_{user\_max(i)}$. Our case is further complicated by the fact that while penalizing the heavy user groups we cannot reduce their current share. This is what might happen in certain

cases while trying to maximize the rates of lower user groups. In the following sections we will discuss various fair sharing methods at the edges.

Allocation of Resources to Lower User Groups First

In this case, we first need to order the user groups based on their $C_{user\_max(i)}$ values to satisfy the lower user groups first by trying to allocate the maximum offered values while the higher user groups have less chances to acquire resources left by the dedicated service group. The rationale behind this is that more VPN users can be satisfied and allocating to the higher user groups might bring little changes in many cases if sufficient extra resources are not available.

If the ordering leads to service groups $1, 2, 3, ..., K-1, K, K+1, ...N-1, N$, it is possible that if we expand K groups the VPN tunnels belonging to those groups will enjoy the maximum offered bandwidth, the $(K+1)$ th group receives the rest of the unused dedicated resource, and other tunnels remain unchanged. The total enhanced shared capacity can then be computed as follows:

$$\begin{aligned} C^{+}_{shared} &= \sum_{i=0}^{K} N_{shared(i)}.C_{user\_max(i)} \\ &+ C_{shared(k+1)} + \Big[C_{ded\_unused} \\ &- \sum_{i=1}^{K}[N_{shared(i)}.C_{user\_max(i)} - C_{shared(i)}]\Big] \\ &+ \sum_{i=K+2}^{N} C_{shared(i)} \end{aligned}$$

The above computation helps us to view how C^{+}_{shared} is shared by different groups. However, this general case is true when $K \geq 1, (N-K) \geq 2$. The other cases are:

$$C^{+}_{shared} = \begin{cases} C_{shared(1)} + C_{ded\_unused} & \text{if } K = 0, (N-K) = 1 \\ \left[C_{shared(1)} + C_{ded\_unused}\right] + & \\ \sum_{i=2}^{K} C_{shared(i)} & \text{if } K = 0, (N-K) \geq 2 \\ \sum_{i=1}^{K} N_{shared(i)}.C_{user\_max(i)} + & \\ C_{shared(k+1)} + C_{ded\_unused} - & \\ \sum_{i=1}^{K} [N_{shared(i)}.C_{user\_max(i)} - & \\ C_{shared(i)}] & \text{if } K \geq 1, (N-K) = 1 \end{cases}$$

In practice, when there is unused dedicated capacity the process starts by asking the first group if the unused capacity is enough to satisfy all the VPN connections. If so, each connection receives a maximum value $C_{user\_max(i)}$ and then queries the second group. Otherwise, the whole amount of capacity is allocated to the first group and divided among the competing connections. The process continues as long as the unused capacity is a positive figure.

Example 10 : Assume a situation where we have 3 groups with VPN connections in each of them having capacity below their respective $C_{user\_max(i)}$. Also, $C_{shared} = 30$ Mbps and for group 1: $C_{base(1)} = 5$ Mbps, $C_{user\_max(1)} = 0.5$ Mbps, $C_{user\_min(1)} = 0.25$ Mbps; for group 2: $C_{base(2)} = 10$ Mbps, $C_{user\_max(2)} = 1$ Mbps, $C_{user\_min(2)} = 0.5$ Mbps; and for group 3: $C_{base(3)} = 15$ Mbps, $C_{user\_max(3)} = 2$ Mbps, $C_{user\_min(3)} = 1$ Mbps. Prior to the availability of $C_{ded\_unused} = 7$ Mbps we had :
$N_{shared(1)} = 15, C_{shared(1)} = 5$ Mbps $C_{user(1)} = 0.333$ Mbps
$N_{shared(2)} = 12, C_{shared(2)} = 10$ Mbps $C_{user(2)} = 0.833$ Mbps
$N_{shared(3)} = 15, C_{shared(3)} = 15$ Mbps $C_{user(3)} = 1.00$ Mbps

Here the groups are already ordered. Applying the algorithms we see that the first two groups can be allocated the maximum rates. Therefore, they are both expanded to $15 \times (0.5) = 7.5$ Mbps and $12 \times 1 = 12$ Mbps respectively. The rest of the unused capacity $C_{ded\_unused} - \sum_{i=1}^{2} [N_{shared(i)}.C_{user\_max(i)} - C_{shared(i)}] = 7 - (7.5 - 5 + 12 - 10) = 2.5$ Mbps goes to the third group.

Allocation of Resources to the Neediest Group First

This is much like the process as described above with the only difference that the groups are ordered based on their needs. Apportionment mechanisms and

algorithms remain the same. As mentioned earlier, need is determined from the ratio of $\frac{C_{user(i)}}{C_{user\_max(i)}}$. So, the groups with lower ratios get preference over the groups with higher ratios. Therefore, the process starts feeding the most needy group and continues as long as it has some unused capacity.

Example 11 : From example 10 of the previous section: $\frac{C_{user(3)}}{C_{user\_max(3)}} = 0.5$, $\frac{C_{user(1)}}{C_{user\_max(1)}} = 0.67$, and $\frac{C_{user(2)}}{C_{user\_max(2)}} = 0.83$. Clearly, group 3 is the most needy group. If we have $C_{ded\_unused} = 5$ Mbps, it can serve the the most needy group 3 and enhance its service. The new $C_{user(3)} = \frac{20}{15} = 1.33$ Mbps and $\frac{C_{user(3)}}{C_{user\_max(3)}} = 0.67$. In the previous example, this group never had the chance to grab a portion of the unused bandwidth, but the new policy here allows it to improve the service substantially.

Allocation of Resources Based on Proportional Needs

Although the above mechanism seems to be fair since it allocates based on the group's need, in many cases there will be several needy groups with little differences in their needs. In such cases, the apportionment might not be always fair if the unused dedicated resources are exhausted while trying to feed the first few groups and others remain deprived to get a share. In this section, we, therefore, present a way to allocate the unused resources based on proportional need. Any group that is in need of resource, i.e. having the ratio $\frac{C_{user(i)}}{C_{user\_max(i)}} < 1$ receives a portion of the unused resource proportional to the group's need. Therefore, any group i, after receiving the extra resource based on this proportional need, is expanded to $C_{shared(i)} = \frac{C_{ded\_unused} \cdot C_{shared\_excess(i)}}{C_{shared\_excess}} + C_{shared(i)}$. Here, the need for group i is actually the excess quantity needed to offer all connections in that group the maximum value $C_{user\_max(i)}$. Therefore, $C_{shared\_excess(i)} = [C_{user\_max(i)} - C_{user(i)}]N_{shared(i)}$.

Example 12: Once again, let us consider example 10 to illustrate the use of proportional needs. No ordering is needed here as the allocation of extra capacity is solely based on the proportional need. Here for group 1: $\frac{C_{user(1)}}{C_{user\_max(1)}} = 0.67$, for group 2: $\frac{C_{user(2)}}{C_{user\_max(2)}} = 0.83$, and for group 3: $\frac{C_{user(3)}}{C_{user\_max(3)}} = 0.5$. Application of this allocation policy will expand the capacity of group 1 to:

$C_{shared(1)} = \frac{7[(0.5)15-5]}{[(0.5)15-5]+[(1)12-10]+[(2)15-15]} + 5 = 5.897$ Mbps. As a result, connections improve with new $C_{user(1)} = 0.393$ Mbps, $\frac{C_{user(1)}}{C_{user\_max(1)}} = 0.79$. Similarly, for group 2: $C_{shared(2)} = 10.71$ Mbps, $C_{user(2)} = 0.89$ Mbps, $\frac{C_{user(2)}}{C_{user\_max(2)}} = 0.89$, and

for group 3: $C_{shared(3)} = 20.39$ Mbps, $C_{user(3)} = 1.36$ Mbps, $\frac{C_{user(3)}}{C_{user\_max(3)}} = 0.68$. This clearly shows that proportional sharing fairly enhances the rate of the most needy group 3. This would not have been the case had we applied other fairness methods.

4.4 Enhanced SB for Dynamic Configuration

The prototype SB presented in chapter 3 has been enhanced to optimally configure network edge resources and support edge provisioning policies discussed in this chapter. Driven by the need to provide range-based SLA the underlying network may provide different classes of QoS enabled VPN services. This requires enhancement of not only core functional engine of the Service Broker, but also the earlier presented web interface that allows the users to customize their own VPN services. The new SB web interface as shown in Figure 4.5 has the required policy options to realize range-based SLA and its benefits. The resource database format also changes to support new range-based SLA. Here, we will only briefly discuss the relevant parts that are mostly responsible for dynamic resource allocation at the edge devices. Details of the implementation, operation and examples of dynamic VPN establishment have already been described in chapter 3 and can also be found in [KB00a], [BGK01], [KBG00]. We will also present some examples of the dynamic rate allocations of VPN connections in commercial routers to illustrate the methods presented in earlier sections.

The basic operation (Figure 3.11) for successful connection setup remains the same. While the SB invokes an *SLA database* to check the validity of the user request, it essentially needs to maintain a *connection database* containing a list of the currently active VPNs and an *edge resource database* to keep track of records of quantitative resource available (base capacity) and current resource consumption of various router interfaces.

4.4.1 Examples of Dynamic Configuration

A resource controller in the SB checks resource and connection databases whenever there is any new connection arrival or departure that might trigger the modification of rates of the existing connections. For a better understanding of how the edge routers are dynamically configured to meet the user demand and conform SLA, we will now demonstrate some examples of the dynamic rate allocations of

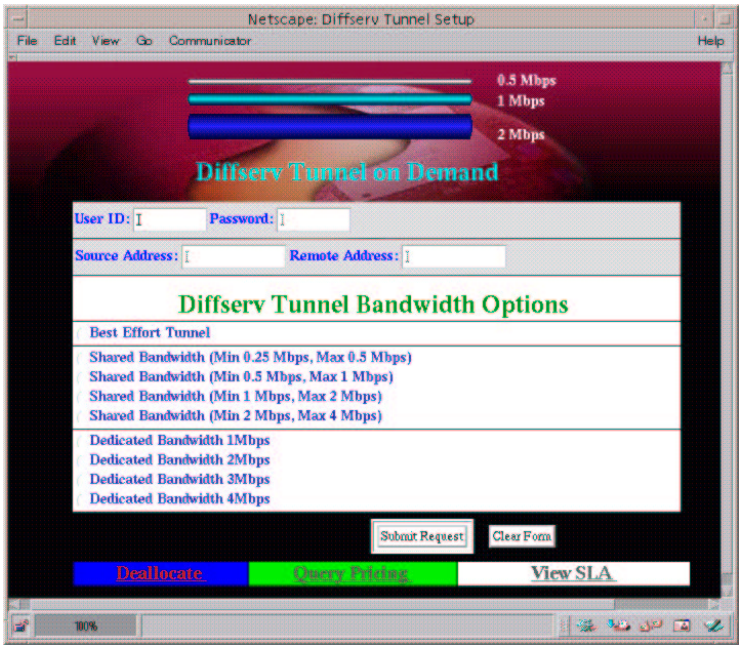


Figure 4.5: SB WEB Interface Supporting Range-Based SLA

the VPN connections in commercial Cisco IOS routers. By considering similar examples, as detailed in section 4.3, we will see how the simple algorithms are really applied to the edge devices. Let us consider an experimental setup (Figure 4.6) of DiffServ-VPNs where we have three VPN and QoS capable edge routers each having a private network behind them.

Configuration 1: User 'A' wants to establish a VPN connection for source 172.17.0.100 and destination 172.20.0.100 and chooses an option (1-2 Mbps) from ISP provided web site and submits a request. Figure 4.7 shows the resource group definition and edge resource database entries. Applying the algorithm presented in section 3, the policing rate $C_{user(1)}$ configured in edge router 130.92.70.101 is $C_{user(1)} = C_{user\_max(1)} = 2$ Mbps. If user 'B' chooses the same option the same rate $C_{user(1)} = 2$ Mbps is allocated since capacity in group 1 has the ability to support that. Assume that two more users 'C' and 'D' decide to have VPN connections (for sources and destinations specified in the connection database of Figure 8) with capacity varying between 0.5 and 1 Mbps. Group 2 can support both the connections with the maximum available rate of 1 Mbps. Therefore, $C_{user(2)} = C_{user\_max(2)} = 1$ Mbps is also configured in the router for these connections, as we see in the following:

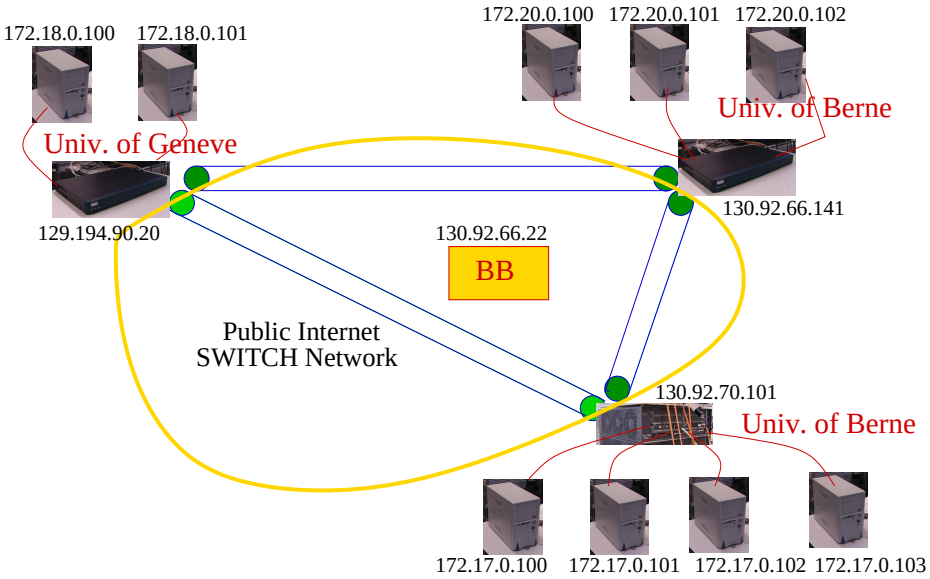


Figure 4.6: Experimental Setup for Demonstration of Range-Based SLA

```
/*policing individual VPN connection at the inbound with C_user(1) = 2 Mbps */
for users 'A' and 'B' and C_user(2) = 1 Mbps for users 'C' and 'D'*/
   rate-limit input access-group 140 2000000 2000000 8000000
     conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
   rate-limit input access-group 141 2000000 2000000 8000000
     conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
   rate-limit input access-group 142 1000000 2000000 8000000
     conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
   rate-limit input access-group 143 1000000 2000000 8000000
     conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
/*Classifying the requested VPN traffic/
   access-list 140 permit ip host 172.17.0.100 host 172.20.0.100
   access-list 141 permit ip host 172.17.0.101 host 172.20.0.101
   access-list 142 permit ip host 172.17.0.102 host 172.20.0.102
   access-list 143 permit ip host 172.17.0.103 host 172.20.0.102
```

Here, we only show the ingress router policing and marking since DiffServ is unidirectional. We assume that bit precedence 1 is used for EF traffic marking and traffic that exceed the specified rate are marked as best effort (bit precedence 2). Users not familiar with Cisco IOS routers should only notice the first of the traffic rate parameters (for example `2000000` in '`2000000 2000000 8000000`') in `rate-limit` policing and marking commands. This is the rate we refer to as $C_{user(i)}$ for any group i. The other two are burst parameters.

Configuration 2: Now, if users 'A' and 'B' also want to establish connections

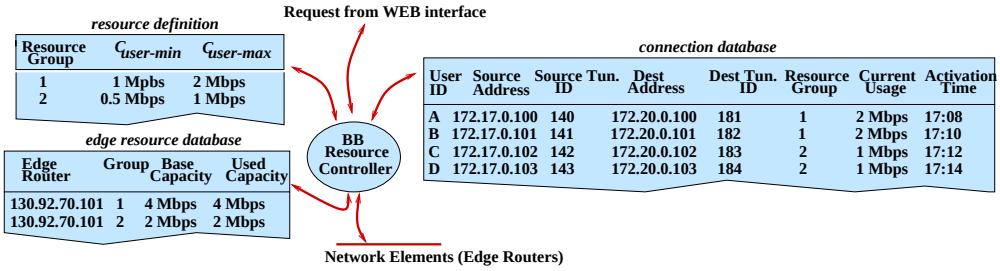


Figure 4.7: Partial Entries in Connection and Resource Databases: A Scenario when all Connections Receive the Maximum Offered Value

from the same sources to 172.18.0.100 and 172.18.0.101 respectively and choose an option (0.5 - 1 Mbps) i.e. group 2, we see that group 2 is exhausted of its capacity. Therefore, these two new connections along with the other two existing connections share the base capacity of 2 Mbps when each connection is configured with $C_{user(2)} = C_{user\_min(2)} = 0.5$ Mbps. This is shown in Figure 4.8 and the new set of configuration commands that are loaded to the router at this point is as follows:

```
rate-limit input access-group 142 500000 2000000 8000000
  conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
rate-limit input access-group 143 500000 2000000 8000000
  conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
rate-limit input access-group 144 500000 2000000 8000000
  conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
rate-limit input access-group 145 500000 2000000 8000000
  conform-action set-prec-transmit 1 exceed-action set-prec-transmit 2
access-list 144 permit ip host 172.17.0.100 host 172.18.0.100
access-list 145 permit ip host 172.17.0.101 host 172.18.0.101
```

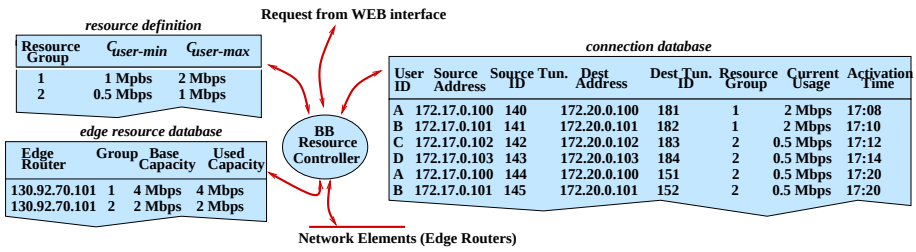


Figure 4.8: A Scenario when Rate of Existing Connections are Reduced to Accommodate New Connections

4.5 Conclusion

In this chapter, we have proposed a novel range-based SLA that allows customers to specify their requirements as a range of quantitative service for VPN connections since they are unable or unwilling to predict the load between the VPN endpoints. To support such services, we have proposed and developed a prototype SB that can logically partition the capacity at the edges to various service classes (or groups) of VPNs and manage them efficiently to allow resource sharing among the groups in a dynamic and fair manner. Various algorithms with examples and analyses have been presented to provision resource dynamically at the edges to support QoS for VPN connections.

One obvious advantage of our system is the pricing gain. The price that customers have to pay is higher than one pays for the lower-bound capacity but lower than what is normally needed to be paid for upper-bound capacity. During low-load it is possible that users might enjoy the upper-bound rate without paying anything extra. Such pricing might be attractive to users and ISPs can take advantage of this to attract more customers.

We have restricted this chapter to edge provisioning only considering the fact that most of the complexities lie at the boundaries of the network and is the main driving force for overall provisioning. However, the ISPs must provision the interior nodes in the network to meet the assurance offered at the boundaries of the network. Core provisioning that works in unison with the proposed edge resource allocation policies here has been addressed in the next chapter.

Chapter 5

Edge Driven Virtual Core Provisioning

In chapter 4 we proposed a range-based Service Level Agreement (SLA) [KB00b] approach and edge provisioning in DiffServ capable Virtual Private Networks (VPNs) to customers that are unable or unwilling to predict load between VPN endpoints exactly. With range-based SLAs customers specify their requirements as a range of quantitative values rather than a single one. Various suitable policies and algorithms dynamically provision and allocate resources at the edges for VPN connections. However, we also need to provision the interior nodes of a transit network to meet the assurances offered at the boundaries of the network. Although a deterministic guaranteed service (single quantitative value approach) provides the highest level of QoS guarantees, it leaves a significant portion of network resources on the average unused. In this chapter, we show that with range-based SLAs providers have the flexibility to allocate bandwidth that falls between a lower and upper bound of the range only, and therefore, take advantage of this to make multiplexing gain in the core that is usually not possible with a deterministic approach. But dynamic and frequent configurations of an interior device is not desired as this will lead to scalability problems and also defeats the purpose of the DiffServ architecture which suggests to drive all the complexities towards edges. We, therefore, propose virtual core provisioning that only requires a capacity inventory of interior devices to be updated based on VPN connection acceptance, termination or modification at the edges.

5.1 Introduction

To build Virtual Leased Line (VLL) type services, VPN traffic is classified and marked with the DSCP for EF at the edges of the of the network. In the interior of the network, with the help of DSCP-PHB mapping [NBBB98], [BBCF01], this quantitative traffic can be allocated a certain amount of node resources. However, if best effort routing based default paths do not meet the requirements of requested VPN connections, MPLS can be used to create pinned paths and force VPN traffic to follow paths that are provisioned with sufficient QoS. In chapter 3 of this thesis, we have proposed the implementation of a Service Broker that allows users to specify a guaranteed service (i.e. a single quantitative value like 1 Mbps or 2 Mbps etc.) and based on this specification the edge routers establish VPN connections dynamically and police traffic according to the specified rate. However, providing guaranteed services exactly as specified by users has the following limitations:

- Although a deterministic guaranteed service provides the highest level of QoS guarantees, it leaves a significant portion of network resources on the average unused.
- It is expected that users will be unable or unwilling to predict load between VPN endpoints [DGG+99]. Also, from the providers point of view guaranteeing exact quantitative service might be a difficult job at the beginning of VPN-DiffServ deployment [BBC+99].

To address these issues we proposed in chapter 4 that users specify their QoS requirements as a range of quantitative services [KB00b]. For example, a user who wants to establish a VPN between stub networks A and D (Figure 5.1), and is not sure whether he needs 0.5 Mbps or 0.6 Mbps or 1 Mbps, and only knows the lower and upper bounds of his requirements approximately, can specify a range 0.5- 1 Mbps as his requirement from the ISP when he outsources the service to the latter. From the resource provisioning point of view ISPs can take advantage of the fact that as long as the lower bound of the bandwidth is guaranteed the SLA will be fulfilled, and thus provision the core in a way that gains from the multiplexing effect. Core provisioning, therefore, is the main focus of this chapter and complements our earlier work of edge provisioning.

In this chapter, we propose virtual core provisioning in a Service Broker architecture where an edge router selects an explicit route and signals the path through

the network, as in a traditional application of MPLS. Router interfaces along these routes are pre-configured to serve a certain amount of quantitative VPN traffic. A new VPN connection is subject to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration of edge devices, but only resource state updates of core routers interfaces in the Service Broker database - hence the naming 'virtual core provisioning'. We propose an architecture for such provisioning and show various ways to update the database in order to support VPN connections with range-based SLAs. We also show how we can exploit range-based SLAs to simplify core provisioning, make multiplexing gain and guarantee at least lower bounds of bandwidth ranges even under heavy VPN demand conditions. Simulation results support our claims and analysis.

5.2 Virtual Core Provisioning Architecture

In DiffServ enabled networks, edge provisioning drives interior (i.e core) provisioning since SLAs are contracted at the boundaries. These are coupled with each other to a high degree in a way that each has direct influence on the other and it would not make much sense to offer guarantees only at the edges which are not met in the interior. Our Virtual Core Provisioning architecture is based on this principle where edge devices maintain the complexity of provisioning, core devices require no explicit configuration and advance reservation states at the core are maintained in a capacity inventory of the Service Broker system. The architecture illustrated in Figure 5.1 comprises policy based edge provisioning and capacity inventory of core devices.

In order to provision the interior based on edge provisioning policies, we first need to know the amount of traffic that would traverse each interior node. Although provisioning a large network for such quantitative services is a difficult problem, computation of resources needed for VPN connections at various nodes can be feasible because of the following facts:

- Both ingress and egress points are known in the case of traffic submitted for quantitative VPN services. Therefore, the direction of traffic is known and traffic admitted into the network is governed by edge provisioning rules.
- Routing topology is often known in advance and stored in the Service Broker database. So, VPN traffic stemming from an ingress node and directed

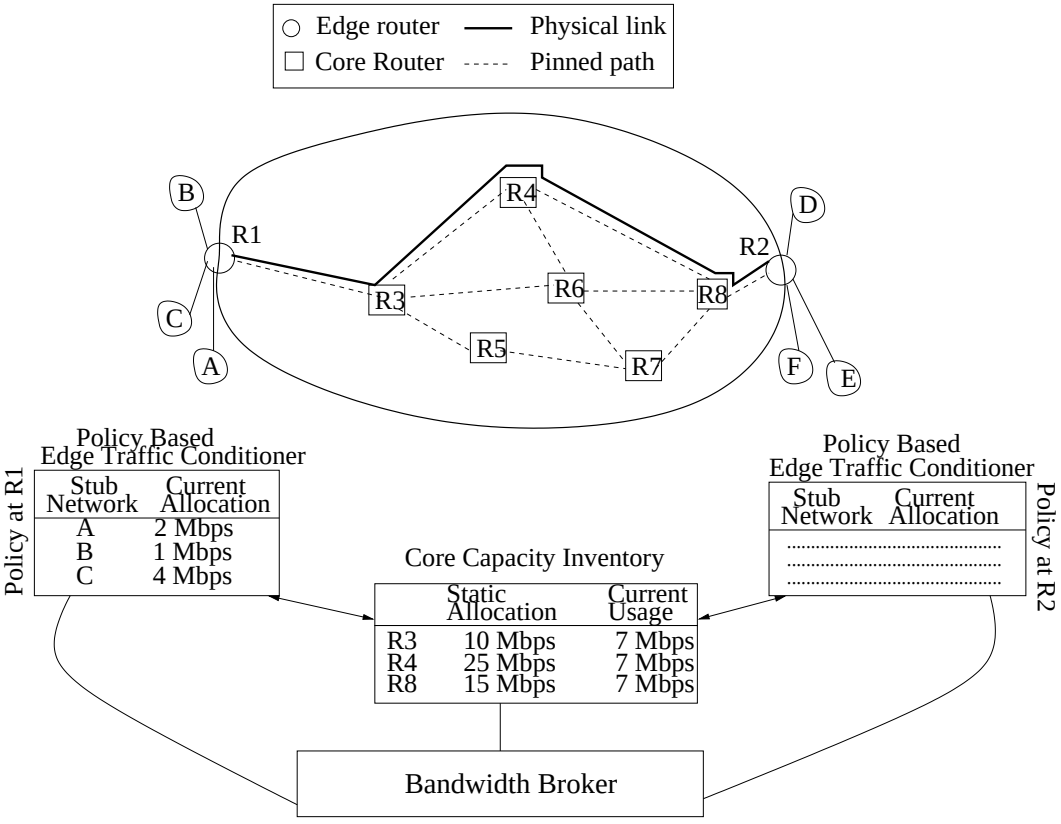


Figure 5.1: Virtual Core Provisioning Architecture

towards an egress node traverses through some specific nodes in the interior network governed by MPLS and route pinning.

In the proposed Service Broker based virtual core provisioning architecture an edge router selects a MPLS enabled pinned path for a VPN connection. Router interfaces along these routes are pre-configured to serve certain amount of quantitative VPN traffic. A new VPN connection is subject to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration at the edge device, but only resource state updates of core router interfaces in the Service Broker database. As shown in Figure 5.1, an explicit path has been setup from router $R1$ to $R2$ that traverses core routers $R3$, $R4$ and $R8$. Each of these core routers is pre-configured to allocate 10, 25 and 15 Mbps of EF marked traffic. If a new stub network, say G (not shown in Figure 5.1), gets hooked up to edge $R1$ and wants to have a 2 Mbps VPN connection to stub network D, this connection request will be accepted if edge $R1$ permits (core devices $R3$, $R4$ and $R8$ have enough capacity left to support this 2 Mbps connection). As a result of this acceptance, $R1$ will actually be configured with appropriate policing, shaping parameters, but only the current usage value for the core devices will be updated (9 Mbps for each) in the core capacity inventory. This inventory only maintains actual pre-configured allocation and the amount reserved for accepted VPN connections.

Although it may seem that like IntServ or ATM based hop by hop approach, a VPN session is established by sending a signaling message to reserve resources for the new flow at each hop along the path, capacity reservation states in the latter case are actually stored in a Service Broker based inventory and not in the core routers. Therefore, unlike the traditional IntServ approach, which has the fundamental scalability limitations because of the responsibility to manage each traffic flow individually on each of its traversed routers, our virtual provisioning approach does not suffer from the same problem.

Virtual core provisioning algorithms cooperate with the dynamic edge provisioning algorithms introduced in chapter 4 and update of core capacity inventory is driven by edge policy rules. This, along with the range-based SLA that gives providers the flexibility to allocate bandwidth between lower and upper bounds of the range only, makes the proposed Service Broker based virtual provisioning architecture advantageous to achieve multiplexing gain in the core that is usually not possible with an IntServ like deterministic approach.

5.3 End-to-End Admission Control

Like edge nodes, only a specific amount of bandwidth will be allocated to VPN traffic in each interior node. If a VPN connection is accepted at the edge but does not find enough resources provisioned for quantitative services at any of the interior nodes along its path, the connection request will be finally rejected.

Based on the earlier discussion we will describe a simple method to estimate the capacity needed at any interior node to support range-based traffic contract promised at the edges. Edge driven capacity estimation at the core could have been simpler if we had used single quantitative values for bandwidth allocation instead of bandwidth ranges. While range-based SLA offers certain economical benefits to both customers and ISPs, it also poses some challenges in terms of implementation. Before discussing the capacity estimation algorithm we first need to define the following terms:

- $e(I, E)$ denotes an edge pair for a VPN connection originating from ingress point I and ending at egress point E where $I \neq E$. If we have total n boundary points then $I = 1, 2, 3, ...n$ and $E = 1, 2, 3, ...n$.

- $\Re$ is the set of all edge pairs in a DiffServ domain, i.e. $\Re \in [e(1,1), e(1,2), e(1,3).....e(n, n-1)]$.

- $IN(i, j)$ denotes interior routers i's jth interface where $i = 1, 2, 3, ...m$ and $j = 1, 2, ..k_i$ if we have m interior routers and any interior router i has maximum k_i interfaces.

- $\Re_{i,j}$ is the set of edge pairs that establish VPN connections which traverse through interior routers i's jth interface.

- $C(i, j)_{e(I,E)}$ is the capacity required at interior i's jth interface for VPN connections between ingress point I and egress point E.

- θ is the set of interior points in DiffServ domains, i.e. $\theta \in [IN(1,2), IN(1,2), IN(1,3)....IN(m, k-1_m), IN(m,k)]$.

- $\theta_{e(I,E)} \in \theta$ is the set of interior interfaces that are traversed by VPN connections having ingress point I and egress point E.

	$IN(1,1)$	$IN(1,2)$	...	$IN(m,k)$
$e(1,2)$	$C(1,1)_{e(1,2)}$	$C(1,2)_{e(1,2)}$	...	$C(m,k_m)_{e(1,2)}$
$e(1,3)$	$C(1,1)_{e(1,3)}$	$C(1,2)_{e(1,3)}$	...	$C(m,k_m)_{e(1,3)}$
$e(1,4)$	$C(1,1)_{e(1,4)}$	$C(1,2)_{e(1,4)}$	...	$C(m,k_m)_{e(1,4)}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$e(n,n-1)$	$C(1,1)_{e(n,n-1)}$	$C(1,2)_{e(n,n-1)}$	...	$C(m,k_m)_{e(n,n-1)}$

Table 5.1: Generalized Resource Table for End-to-End Connection Admission Control

Therefore, $C(i,j)$, the resources needed for all VPN connections that traverse through a router i's j th interface can be expressed as:

$$C(i,j) = \sum_{\Re_{i,j} \in \Re} C(i,j)_{e(I,E)}$$

This is actually computed from the matrix shown in Table 1. In Table 1, each cell represents $C(i,j)_{e(I,E)}$. The horizontal labels indicate interfaces of interior routers and the vertical labels denote ingress/egress edge pairs. Not all cells carry numerical values since only a few of the interfaces are met by VPN traffic for a certain edge pair. Therefore, many of the cells will actually contain null values. Information regarding which interfaces are met by a VPN flow is extracted from the routing topology database used in the Service Broker.

There are various ways to use this matrix for connection admission and resource provisioning. This matrix is basically a representation of resources currently reserved for quantitative traffic at various interior nodes for VPN traffic stemming from edges. For admission control purposes, ISPs can define a similar matrix where each cell represents an upper bound value $C(i,j)_{upper}$ for quantitative traffic reservation. $C(i,j)_{upper}$ can be exactly equal to C_{quan} as shown in Figure 4.3(a) or an over-estimated value of C_{quan} to take advantage of the multiplexing effect in the interior routers where several connections are bundled and allocated an aggregated capacity. For example, if in reality $C_T = 500$, and $C_{quan} = 0.2C_T = 100$ Mbps for an interior router i's jth interface, ISP can set $C(i,j)_{upper} = 1.5C_{quan} = 150$ Mbps to gain from multiplexing and knowing the fact that not all connections will be sending at the highest rate at the same time. So, setting this value depends on how much risk ISPs want to take.

Whenever a new VPN connection request arrives at an ingress point destined towards an egress point, all the valid cells (not containing null values) are checked row-wise for that edge pair. If the capacity at each of the interfaces are sufficient

,i.e. does not exceed the upper bound values even after being accepted, then with this acceptance all the cells are updated to show the most recent reservation. In fact, end-to-end admission can be presented as follows:

$if\Big(N_{shared(i)} \leq \frac{C_{base(i)}}{C_{user\_min(i)}}\Big)$
{
compute $C_{user(i)}$;
$if\Big(C(i,j)_{upper} > C(i,j)_{computed} + C_{user(i)}\Big)$
for all $\theta_{e(I,E)} \in \theta$
{
accept connection request;
$C(i,j)_{e(I,E)} = C(i,j)_{e(I,E)} + C_{user(i)}$ *for* $\theta_{e(I,E)} \in \theta$
allocate and provision resources;
}
}

Here $C(i,j)_{computed}$ is the most recent updated value of $C(i,j)$. This is because, a connection arrival, for example, might trigger changes in existing connections and if such things happen then $C(i,j)$ is computed taking these changes into consideration before the end-to-end admission algorithm can decide correctly. The same algorithm can be repeated for alternate routing paths (also stored in the topology database) if the default or the MPLS based pinned path does not satisfy the requirements.

5.4 Cases of Core Capacity Inventory Update

Based on the dynamic edge provisioning policies a new connection arrival or departure of a connection might require existing connections to reduce current rates or re-negotiate for possible expansion. Actually, such an arrival or departure might force several connections to change rates not only at the edges but also in interior nodes on connection by connection basis. Although this poses some difficulties, ISPs need to maintain up-to-date interior network state. Here we will present the possible cases that might happen in a network.

- Case I: A new connection request arrives triggering reductions of existing VPN connections at the ingress edge.

	$IN(1,1)$	$IN(1,2)$	$IN(1,3)$	$IN(2,1)$	$IN(2,2)$	$IN(2,3)$
$e(1,2)$	-	0	-	-	-	-
$e(1,3)$	-	-	10	-	10	-
$e(1,4)$	-	-	20	-	-	20
$e(2,1)$	0	-	-	-	-	-
$e(2,3)$	-	-	15	-	15	-
$e(2,4)$	-	-	25	-	-	25

Table 5.2: Resource Table Before Connection Arrival

	$IN(1,1)$	$IN(1,2)$	$IN(1,3)$	$IN(2,1)$	$IN(2,2)$	$IN(2,3)$
$e(1,2)$	-	0	-	-	-	-
$e(1,3)$	-	-	9.67	-	9.67	-
$e(1,4)$	-	-	19.33	-	-	19.33
$e(2,1)$	0	-	-	-	-	-
$e(2,3)$	-	-	15	-	15	-
$e(2,4)$	-	-	25	-	-	25

Table 5.3: Resource Table After Relinquishing 1 Mbps of Capacity From Group 2

- Case II: A new call arrives which does not cause changes of existing VPN connections at the edge.
- Case III: A call departs leaving extra capacity at the edge (as unused resources) but the active connections do not need to use any portion of it.
- Case IV: A call departs leaving extra resources for existing connections to be shared at the edge.

5.4.1 Case I

In such a case, when a new connection request arrives, existing connections of that group or other group(s) have to reduce their rate at the ingress because of respective sharing policy. From the resource management point of view reduction of rates of existing connections do not cause renegotiation in the interior of the network. Only the new connection negotiates at various interior points between its ingress and egress point and if it finds sufficient resources at all points then

	$IN(1,1)$	$IN(1,2)$	$IN(1,3)$	$IN(2,1)$	$IN(2,2)$	$IN(2,3)$
$e(1,2)$	-	0	-	-	-	-
$e(1,3)$	-	-	10.67	-	10.67	-
$e(1,4)$	-	-	19.33	-	-	19.33
$e(2,1)$	0	-	-	-	-	-
$e(2,3)$	-	-	15	-	15	-
$e(2,4)$	-	-	25	-	-	25

Table 5.4: Updated Resource Table After Connection is Provisioned

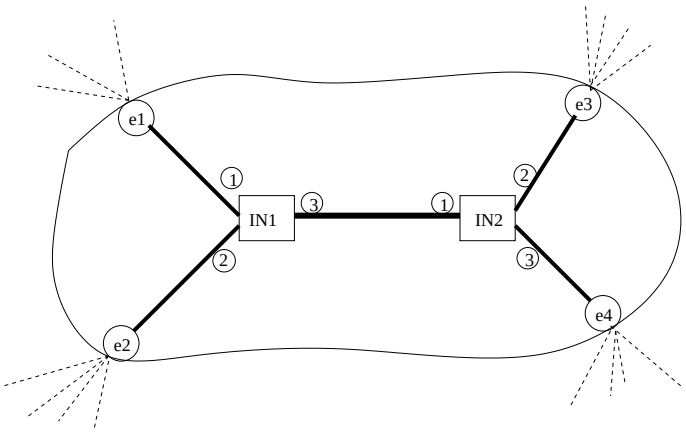


Figure 5.2: Topology of Network for Example 5.3.1

the request is accepted and the resource table for the interior is updated for this acceptance. We will present a detailed example of this case that will explain the analysis and algorithms presented in earlier section.

Consider a scenario as shown in Figure 5.2. In this simple case we have only two interior routers and four edge routers. For QoS allocation only uni-directional traffic flow guaranteeing and policing VPN traffic from $e1$ and $e2$ towards $e3$ and $e4$ is taken into consideration . Assume that quantitative capacities reserved by the ISP at various interfaces are as follows:

$C(1,1)_{upper} = 50$ Mbps at $IN(1,1)$
$C(1,2)_{upper} = 50$ Mbps at $IN(1,2)$
$C(1,3)_{upper} = 80$ Mbps at $IN(1,3)$
$C(2,1)_{upper} = 75$ Mbps at $IN(2,1)$
$C(2,2)_{upper} = 50$ Mbps at $IN(2,2)$
$C(2,3)_{upper} = 50$ Mbps at $IN(2,3)$

For this example, however, only $C(1,3)_{upper}$, $C(2,2)_{upper}$ are of interest if we consider only unidirectional QoS allocation. Consider that at ingress point $e1$ capacity sharing policies are:

Group 1: $N_{shared(1)} = 6, C_{shared(1)} = 6 \times 1 = 6$ Mbps, $C_{base(1)} = 10$ Mbps, $C_{user\_min(1)} = 0.5$ Mbps, $C_{user\_max(1)} = 1$ Mbps and
Group 2: $N_{shared(2)} = 12, C_{shared(2)} = 12 \times 2 = 24$ Mbps, $C_{base(2)} = 20$ Mbps, $C_{user\_min(1)} = 1$ Mbps, $C_{user\_max(1)} = 2$ Mbps

A detailed traffic distribution before the arrival of a VPN connection request in group 1 (all coming from $e1$) is:

Group 1: 2 connections towards $e3$, 4 connections towards $e4$
Group 2: 4 connections towards $e3$, 8 connections towards $e4$

At the same time, VPN connections stemming from ingress point e2 and having egress at e3 and e4 require 15 Mbps and 25 Mbps respectively, lead to the overall capacity matrix as follows:

$$C = \begin{array}{c} \\ e1 \\ e2 \end{array} \begin{array}{c} \begin{array}{cccc} e1 & e2 & e3 & e4 \end{array} \\ \begin{bmatrix} 00 & 00 & 10 & 20 \\ 00 & 00 & 15 & 25 \end{bmatrix} \end{array}$$

By extracting relevant data from the topology database for this simple network the resource table can be easily seen as in Table 2.

Clearly, $C(1,3) = C(1,3)_{e(1,3)} + C(1,3)_{e(1,4)} + C(1,3)_{e(2,3)} + C(1,3)_{e(2,4)} =$ 10+20+15 +25 = 70 Mbps. Similarly, $C(2,2) = 10 + 15 = 25$ Mbps, and $C(2,3) = 20 + 25 = 45$ Mbps.

An arrival of a request (at $e1$) in group 1 for a connection towards $e3$ will allow this connection and all other existing connections in group 1 to have 1 Mbps at the ingress because $C_{base(1)} - C_{shared(1)} = 10 - 6 = 4$ Mbps and this means that group 1 has not used all its base bandwidth and a new connection can have the maximum offered bandwidth of 1 Mbps. This, however, reduces the share of each connection in group 2 to $\frac{23}{12}$ Mbps as that group had borrowed $C_{shared(2)} - C_{base(2)} = 24 - 20 =$ 4 Mbps. Therefore, with the newly computed rates for existing connections and without taking the new connection request into consideration of computation, we have: $C(1,3)_{computed} = (2 + \frac{23}{12} \times 4) + (4 + \frac{23}{12} \times 8) + 15 + 25 = 69$ Mbps. Also, $C(2,2)_{computed} = (2 + \frac{23}{12} \times 4) + 15 = 24.67$ Mbps. Resource table after relinquishing 1 Mbps of capacity from group 2 is shown in Table 3.

Now, the application of the end-to-end admission algorithm shows that $C(1,3)_{upper} > C(1,3)_{computed} + C_{user}(1)$ and $C(2,2)_{upper} > C(2,2)_{computed} + C_{user}(1)$. Therefore, the new connection request is accepted and the resource table is updated as shown in Table 4.

5.4.2 Case II

Consider the scenario of the previous example, but assume that before the arrival of a VPN connection request in group 1 (at $e1$) towards $e3$ or $e4$, we have $N_{shared(1)} = 5$ ($i.e. C_{shared(1)} = 5$ Mbps) and $N_{shared(2)} = 10$ ($i.e. C_{shared(2)} = 20$ Mbps). Since no existing connections are modified at the edge, the resource table (core capacity inventory) keeping track of interior resources do not need to be updated before the admission process for the requested connection can take place. However, the new connection request must check all the appropriate interior points before being finally admitted. Once accepted, the core capacity inventory is updated.

5.4.3 Case III

This is a case when a call departs and does not trigger changes of existing connections in that group and also in other groups. In the previous example if $N_{shared(1)} = 10$ ($i.e. C_{shared(1)} = 10$ Mbps) and $N_{shared(2)} = 10$ ($i.e. C_{shared(2)} = 20$ Mbps) and a VPN connection departs from group 1, neither group 1 nor group 2 needs to change the rate of active connections. Interior points through which the connection had been established are detected and the resource table is updated accordingly.

5.4.4 Case IV

When a VPN tunnel is disconnected leaving extra resources for existing connections to be shared at the edge, the expandable connections having a rate less than $C_{user\_max(i)}$ need to renegotiate for possible expansion at each appropriate interior nodes. To illustrate this we continue to consider an example of case I. The final state at the edge $e1$ was:

Group 1: $N_{shared(1)} = 7, C_{shared(1)} = 7 \times 1 = 7$ Mbps and
Group 2: $N_{shared(2)} = 12, C_{shared(2)} = 12 \times \frac{23}{12} = 23$ Mbps

Obviously, we had the interior resource state as shown in Table 4. Now assume that a connection departs from group 1. That leaves 1 Mbps of unused capacity that can be used to expand the existing connections in group 2. For this simple case although it is quite clear that all the existing connections will be allowed to

expand to 2 Mbps and we will eventually return to the starting point of example in case I, there will be cases when not all the connections in a group will find sufficient resources at each of their appropriate interior nodes to make an end-to-end renegotiation successful. In such a case connections in the same group will have different rates. This is because, although the connections in the same group can have equal resources at the edge, it is very unlikely that connections traversing through different transit paths in interior network will find equal resources on the respective path. While some connections may find only minimum offered bandwidth, others might still find maximum offered bandwidth on an end-to-end basis.

Therefore, we need to look at each connection individually and apply the end-to-end admission algorithm of section 5.3 in the same way we had earlier described it in example of case I. Once again, we first have to decide how to share the unused capacity and who should have the priority to grab this resource. Such fairness issues were discussed in detail in [KB00b]. For simplicity, the group with lowest base capacity has the highest priority.Since the connections might have varying rates, the capacity consumed by a certain group can be $C_{shared(i)} = \sum_{l=1}^{N_{shared(i)}} C_{user(i,l)}$. $C_{user(i,l)}$ is the rate of the l-th connection of group i where $l = 1, 2, 3.....N_{shared(i)}$ and $i = 1, 2, 3.....N$. Some or all of the existing connections in each group that need to expand are also sorted according to the rate $C_{user(i,l)}$.

We will basically consider two cases. First, we need to check the condition $\left(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} \leq C_{user\_max(i)}\right)$. Here, we try to do equal expansion to all connections regardless of their current rate $C_{user(i,l)}$ by offering the addition of $\frac{C_{unused}}{N_{shared(i)}}$ to each of the connections. The goal, as usual, is to bring the rate of the expandable connections equal or close to $C_{user\_max(i)}$. Therefore, if the condition is true, then the connection is considered for possible expansion. But before we can do that, we have to check if this expansion is permitted along all the interior nodes between the VPN end points (ingress and egress). Positive answers for all the nodes finally leads to end-to-end expansion. C_{unused} is updated as $C_{unused} = C_{unused} - \frac{C_{unused}}{N_{shared(i)}}$.

The second case, if found true, will also lead to similar end-to-end expansion. It says that even if $\left(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} > C_{user\_max(i)}\right)$, $C_{user(i,l)}$ might be less than $C_{user\_max(i)}$. This implies that equal expansion might cause the current rate to exceed the maximum offered rate, but otherwise is less than the maximum offered rate, and therefore, eligible for end-to-end expansion. So, the connection in question is expanded to $C_{user\_max(i)}$ and unused resource is updated as $C_{unused} =$

$C_{unused} - [C_{user\_max(i)} - C_{user(i,l)}]$. The end-to-end admission algorithm can be presented as :

for each ordered group i where $i = 1, 2, 3.....N$
{
compute $C_{shared(i)} = \sum_{i=1}^{N_{shared(i)}} C_{user(i,l)}$
sort connections $l = 1, 2, 3.....N_{shared(i)}$ *according to*
rate $C_{user(i,l)}$
for $l = 1$ *to* $N_{shared(i)}$
{
$if\Big(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} \leq C_{user\_max(i)}\Big)$
{
do end-to-end admission at interior points
if OK then expand connection to $C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}}$
$C_{unused} = C_{unused} - \frac{C_{unused}}{N_{shared(i)}}$
$N_{shared(i)} = N_{shared(i)} - 1$
}
else if $\Big(C_{user(i,l)} < C_{user\_max(i)}\Big)$
&& $\Big(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} > C_{user\_max(i)}\Big)$
{
do end-to-end admission at interior points
if OK then expand connection to $C_{user\_max(i)}$
$C_{unused} = C_{unused} - [C_{user\_max(i)} - C_{user(i,l)}]$
$N_{shared(i)} = N_{shared(i)} - 1$
}
}
}

Now let's go back to the example again. We are to find out what happens if a connection terminates from group 1. As it can be easily seen, this will make the resource table look like as shown in Table 3. Now scanning through all the connections of group 2 and applying condition $\Big(C_{user(i,l)} + \frac{C_{unused}}{N_{shared(i)}} \leq C_{user\_max(i)}\Big)$ of the above algorithm (actually doing admission test at each interior point in a similar way as explained in example of case I) we see that $C_{user(2,1)} + \frac{1}{12} \leq 2$, $C_{user(2,2)} + \frac{1}{12} \leq 2$, ..., $C_{user(2,11)} + \frac{1}{12} \leq 2$, $C_{user(2,12)} + \frac{1}{12} \leq 2$. Since re-negotiations of all connections are successful in the example, the resource table will finally look like what we have previously seen in Table 2.

With all the examples in this section we have clearly showed how a core capacity inventory can be updated based on edge provisioning policies. The four cases that we have explained with examples outline all possible states that a node might have with a connection arrival or termination. Although we did not show by an example how a connection could choose an alternate route in case the primary route does not meet admission criterion, it is easily understood that the application of the same end-to-end admission algorithm will produce the desired result should the latter (i.e. the alternate route(s)) have sufficient resources.

5.5 Simplified Core Update

To maintain exact capacity reservation states of core interfaces the update cases presented in the previous section require a significant amount of computation in the Service Broker system and makes the VPN connection acceptance or expansion complicated in certain situations. In case I, to admit a new connection existing connections not only reduced rates at edges, but the core capacity inventory was updated for every single connection at the appropriate interfaces. Even worse, in case IV, existing connections were required to renegotiate for capacity expansion at several core interfaces and a success in renegotiation triggered several core capacity updates.

Although the purpose of virtual core updates is to make reservations at the core accurate and consistent with edge provisioning, such complexities can actually be avoided while still guaranteeing the bandwidth promised at the edge. In fact, we can simply update the appropriate core interfaces with the minimum guaranteed bandwidth each time a VPN connection is accepted and release the same if terminated. This is done by taking advantage of the fact that with range-based SLAs only lower bound capacity needs to be guaranteed and the multiplexing effect in the core leaves enough room to adopt a more aggressive approach and actually accommodate more connections than it is possible if $C_{user(i)}$ is used for virtual core updates.

We will explain with an example here before presenting simulation data to support our idea. Consider a scenario (Figure 5.3) where edge $e1$ accommodates group 1 requiring (0.5-1) Mbps with $C_{base(1)}$= 3 Mbps. Another edge $e2$ supports group 2 requiring (1-2) Mbps with $C_{base(2)}$= 6 Mbps. Core router $R1$ is configured to allocate 4.5 Mbps premium traffic. (i.e $C(i,j) = C(i,j)_{upper} = 1.C_{quan}$=4.5 Mbps).

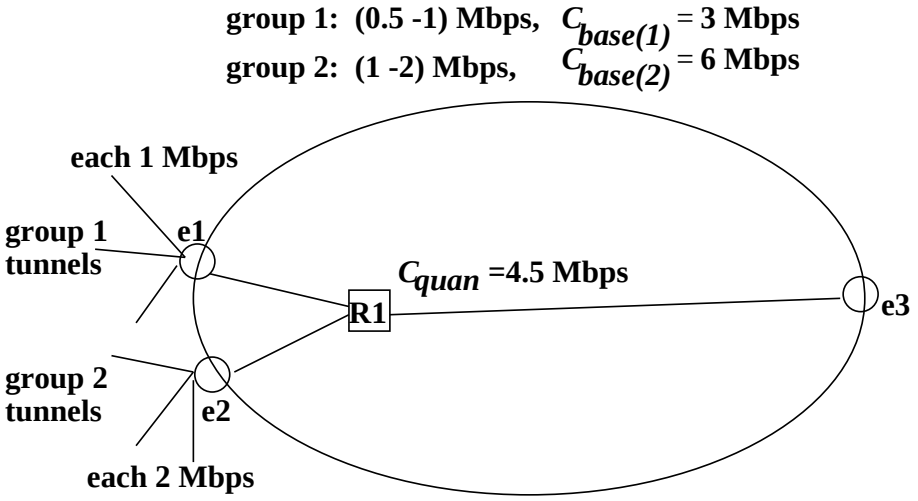


Figure 5.3: Worst Case Scenario: If all connections send traffic at max. configured rate some of them might not get minimum guaranteed capacity

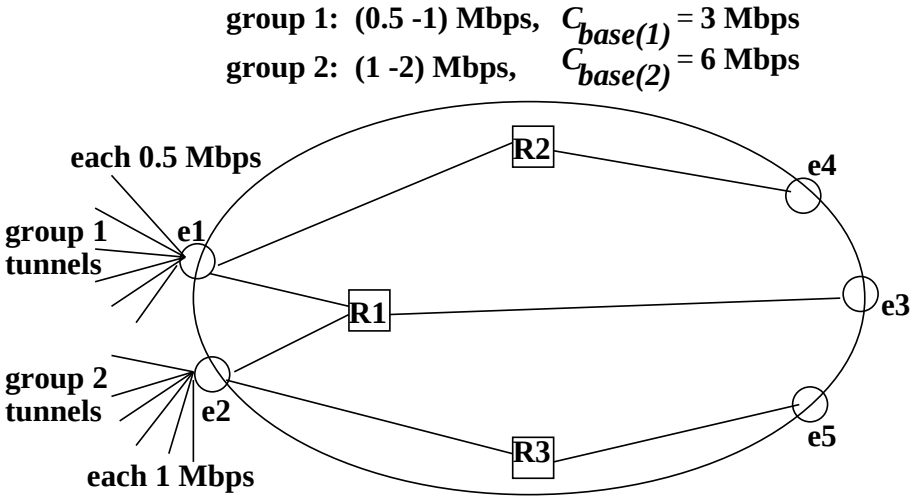


Figure 5.4: Heavy VPN Demand: Arrival of more connections make sure that old connections get at least minimum guaranteed bandwidth

Currently, three 1 Mbps VPN connections at $e1$ and another three 2 Mbps connections are active. As we update the core capacity inventory with $C_{user\_min(i)}$ rather than $C_{user(i)}$, each time a (1-2) Mbps connection gets accepted we increment $C(i,j)$ (for core router $R1$) with $C_{user\_min(2)}$=1, and also similarly for (0.5-1) Mbps connection acceptance. Although the the probability of acceptance increases (i.e blocking probability decreases), in the worst case if all the accepted connections send traffic at the maximum configured rate at the same time, some connections might not even get the minimum guaranteed bandwidth.

However, by law of large number, as more connections are accepted at edge, the probability of each connection getting the minimum bandwidth increases. This is true in our example where acceptance of 3 more connections of existing types at both $e1$ and $e2$ (destined towards $e4$ and $e5$) ensures that every single accepted connection gets the lower bound of the bandwidth range even in the worst case. The example is illustrated in Figure 5.4.

5.6 Simulation

In this section, we present simulation results to show the average rate achieved by accepted VPN connections in a relatively large network under different demand conditions. Simulation studies presented here obviously consider simplified core update cases and confirms earlier analysis presented in the previous section.

A recent trend on achieving multiplexing gain relies on the assumptions that connections (flows) are statistically independent and smoothed by deterministic regulators at the connections input to the network since statistical characterization of traffic sources is not often reliable [BBLO00], [RRR98]. Not surprisingly, this exactly resembles our case. VPN connections are rate controlled based on provisioning policies at the provider edge. In fact, many of the results derived in those will, therefore, be valid in our case too. One interesting result [RRR98] is: By statistically multiplexing rate controlled (at edge) traffic in the core network the number of accepted connections can be three times higher than that of Generalized Processor Sharing [PG93], [PG94] or any other deterministic service discipline [Cru95].

The simulation setup that we consider for our experiment is as shown in Figure 5.5. This network has 10 edge nodes and a total of 14 core interfaces from 3 core routers. Each edge node can accept a maximum of 10 connections from

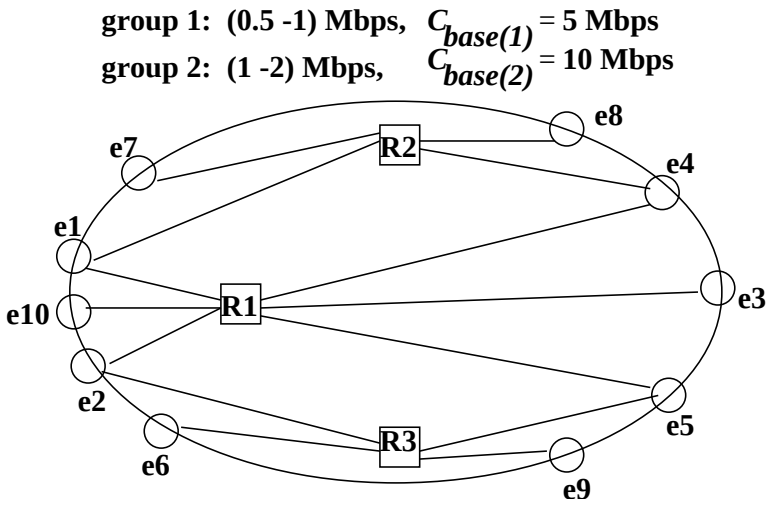


Figure 5.5: Experimental Setup for Simulation

each group when sending at the lower bound rate. As there are 10 edge nodes, a total of 150 Mbps might enter the transit network at a time. Also, since there are 14 interior interfaces, we configure each interface with 11 Mbps (approx.) on average.

Figure 5.6 plots the average bandwidth achieved by 20 connections from each group over a period of 1 hour. During this one hour period 70 connections from each group were actively sending traffic between a range of minimum and maximum allowable bandwidth (i.e 0.5-1 Mbps for group 1 and 1-2 Mbps for group 2) to the network. However, the 40 connections (20 from each group) selected for plotting were accepted at the edge to send traffic at the highest possible rate and were actually spraying traffic at that rate (i.e. 1 and 2 Mbps for group 1 and 2 respectively). Figure 5.7 also shows the average of 20 connections (from each group), but in this case 60 connections from each group were active. Obviously, the average rate improved slightly in this case. It is important to note that although we provision and update the core with less capacity than that is needed for maintaining exact core capacity inventory, accepted VPN connections were receiving almost the upper bound capacity.

One fundamental drawback of deterministic service is that, by its very nature, it must reserve resources according to a worst case scenario, and hence has limits in its achievable utilization. To overcome the utilization limits of a deterministic service, statistical multiplexing must be used assuming that a worst case scenario

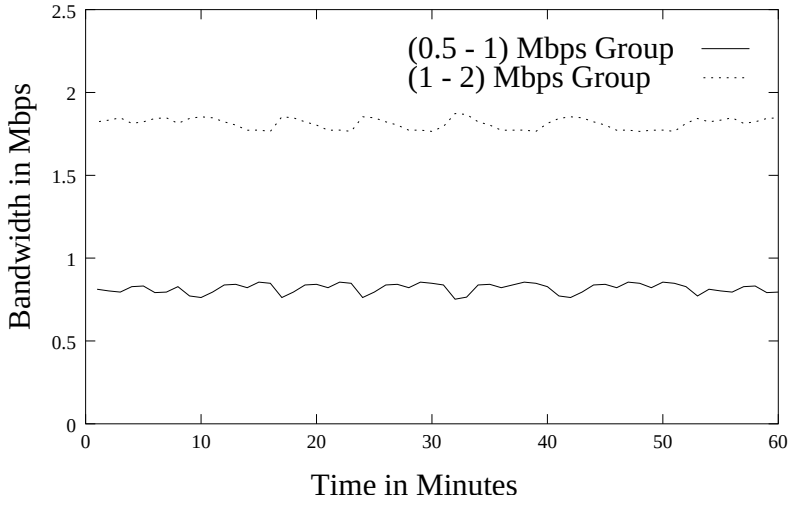


Figure 5.6: Simulation Result 1: Average of 20 connections, total accepted connections 70 from each group

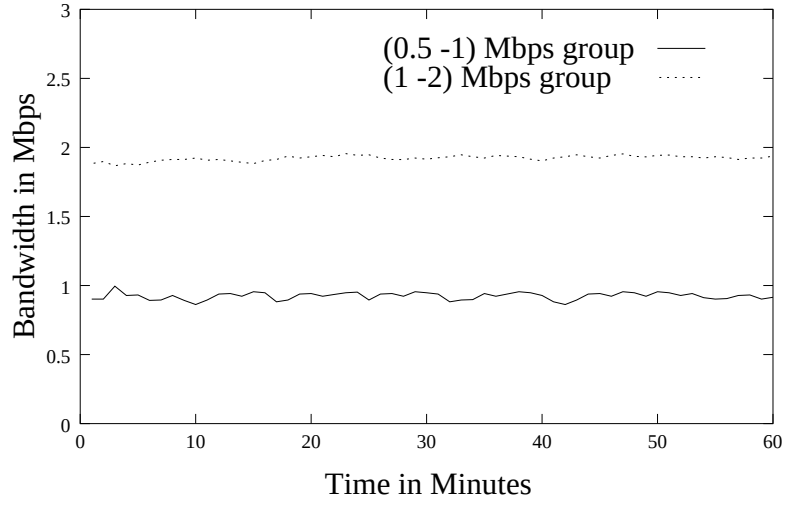


Figure 5.7: Simulation Result 2: Average of 20 connections, total accepted connections 60 from each group

will quite rarely occur. The worst case scenario is a bit different in our case. This might happen when a core interface is configured to support the minimum guaranteed bandwidth no matter what the edge allocates to accepted connections, and all the connections start sending at their fullest configured rate. However, as the number of accepted connections increases, the probability that the worst case might happen starts diminishing. This is shown in Figure 5.8 where we plot the average of 30 accepted connections from each group where each connection was configured with the lower bound capacity at the edge and the number of total accepted connections during the 1 hour measurement period was 85 from each group. This also confirms our previous analysis in section 5.5.

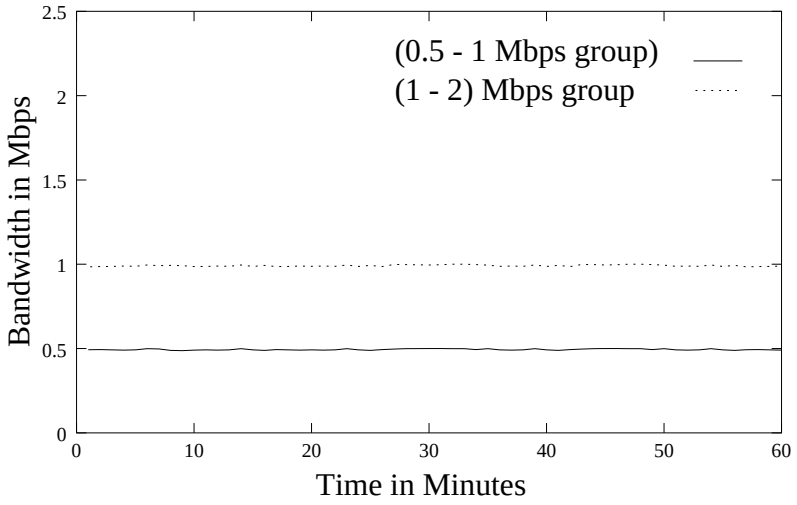


Figure 5.8: Simulation Result 3: Average of 30 connections, total accepted connections 85 from each group

5.7 Conclusion

In this chapter, we have proposed virtual core provisioning in a Service Broker architecture for QoS enabled VPN connections. As users of such connections are unable or unwilling to predict load between the VPN endpoints, we proposed in chapter 4 that customers specify their requirements as a range of quantitative values in the Service Level Agreements (SLAs) for VPN connections. This chapter shows how we can exploit range-based SLAs to simplify core provisioning, make multiplexing gain and guarantee at least lower bounds of bandwidth range even under heavy VPN demand conditions. Simulation results support our claims and

analysis.

In our virtual core provisioning architecture, an edge router selects an explicit route and signals the path through the network, as in a traditional application of MPLS. Router interfaces along these routes are pre-configured to serve certain amount of quantitative VPN traffic. A new VPN connection is subject to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration of edge device, but only resource state updates of core routers interfaces in the Service Broker database. Other works that propose guaranteed services without per flow provisioning at core are: [SZ99], [SSZ98], [CK00], [ZDGH00]. However, all of them consider short-lived flows while VPN connections in our case are usually rate-controlled long-lived flows that are often provisioned for larger time-scale.

The centralized SB in its role as a global network manager maintains information about all the established real-time VPN tunnels and the network topology, and can thus select an appropriate route for each real-time connection request. If a pinned path or pre-selected alternate routes fail to reserve requested resources for a VPN connection, QoS routing can then be used efficiently. Since the objective of any routing algorithm is to find a qualified path with minimal operational overheads, a centralized SB based QoS routing might be very effective. This is an issue we have not addressed and can be a future research topic.

Chapter 6

Capacity Reservation in Multi-Domains

The Service Broker implementation in chapter 3 and its enhancements in chapter 4 and 5 to dynamically provision edge and core bandwidth consider providing such services only in a single provider domain. However, many of the ultimate customers might actually want to extend Virtual Leased Lines (VLLs) up to the periphery of other providers. In this chapter, we describe the implementation of a Bandwidth Broker (BB) that uses simple signaling mechanism to communicate with other cooperative Brokers to enable customers to dynamically create VLLs over multiple Diffserv domains. QoS configuration is a subset of a Service Broker's functionalities. In this chapter, we call the automated broker as *Bandwidth Broker* since we are essentially dealing with only with QoS, i.e bandwidth.

6.1 Introduction

To take advantage of the new DiffServ technology Bandwidth Brokers that can dynamically create VLL on demand have been proposed in [NJZ99], [Sch98] and refined in [TWOZ99], [Tea99]. New Bandwidth Broker models based on the existing architectures have also been proposed in [KBG00], [KB01a], [MlCPT00], [ZDGH00] and several implementations have been reported in [KBG00], [TWOZ99], [Tea99], [KB00b]. Most of these implementations of Bandwidth Brokers have the following characteristics in common:

- They are mostly responsible for a single Diffserv Domain. The Bandwidth Brokers are capable of advance or immediate reservation only in the domains they maintain.

- All the concepts propose policing users traffic at the ingress edge only. Except [KB00a] most of the BBs do not consider interior provisioning. Also, we addressed this issue in chapter 5.

- Almost all except [KBG00], [KB01a] and [KB00a] rely on RSVP for both intra-domain and inter-domain signaling.

While the existing Bandwidth Broker implementations do not yet have mechanisms to communicate with other neighboring domains, they mostly propose modified RSVP or similar mechanisms as the method for both inter-domain and intra-domain signaling. Although both of these have the potential to be integrated in the future advanced Bandwidth Brokers, at the moment when there are core Service Level Agreements (SLAs) and resource provisioning issues yet to be solved [BBC+99], they (i.e. the use of RSVP like signaling) might further complicate implementation issues and delay easy and rapid deployment. For example, as mentioned in [Sch98], for resource reservation over a Diffserv network using such Bandwidth Brokers, both sending and the receiving hosts need to be present during reservation and also during the period the reserved interval starts. In reality, the sender or receiver might not even exist during the reservation process.

In this chapter, keeping this in mind, we present a simple approach to make advance reservations in the absence of senders or receivers in a multi-domain scenario. Rather than using RSVP in inter-domain signaling to reserve capacity across domains, we use a novel method to identify domains, and hence the Bandwidth Brokers that are responsible for maintaining them. Section 6.2 presents basic components and ingredients for making reservations over several Diffserv domains with Bandwidth Brokers. Section 6.3 describes implementation architecture and the components in that architecture. In section 6.4, we describe operational details and system flows of the BB, and in section 6.5 we clarify the operational details by presenting some real examples. Finally, in section 6.6, we conclude this chapter with a summary and future research directions.

6.2 End to End Capacity Reservation

6.2.1 An Example Scenario

Although in chapter 2 we have discussed DiffServ and its applications in details, let us review how this applied in multi-domain scenario. This will help us to identify the additional functionalities required to make the earlier prototype SB of chapter 3 work in multi-ISP environment.

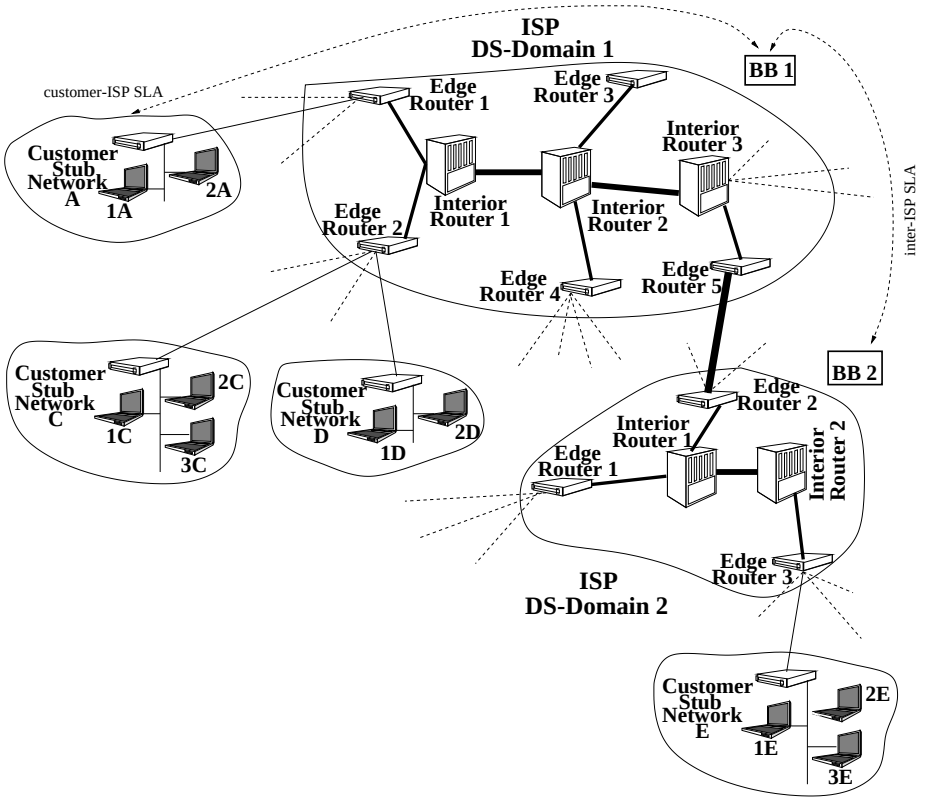


Figure 6.1: Capacity Reservation across Multiple Diffserv Domains

Consider the scenario as shown in Figure 6.1. The domains are Diffserv [BBC+98] enabled and under different administrative control. This means that if stub networks C or D in domain 1 want to establish VLLs with stub network A in the same domain or with stub network E in domain 2, traffic entering domain 1 is classified and possibly conditioned at the boundaries (edge router 2) of the network, and assigned to different behavior aggregates. Each behavior aggregate is identified by a single DS codepoint (DSCP for EF). In the interior (and also

egress) of the network, with the help of DSCP-PHB mapping certain amount of node resources can be allocated for this quantitative traffic.

6.2.2 Functions of Bandwidth Broker in Multi-Domain

In the example above if the administrative control of each ISP is given to an automated system like a Bandwidth Broker, its responsibilities will be :

- Check request validity. In the example, for the VLL over domains, BB 1 needs to check the validity of stub network A's request and BB 2 needs to check the request of ISP domain 1.
- Perform admission control in its domain. In a simple case, this can be only checking resource availability at the border routers as these are the obvious points that will traversed by a VLL connection. In a more advanced case this can be checking resource availability at all the routers along the path from ingress towards egress.
- Coordinate with other separately administered Bandwidth Brokers. In the example, Bandwidth Broker 1 will need to signal to Bandwidth Broker 2 for resource reservation on behalf of the stub network A. If there are several ISP domains each managed by such Brokers, the job of this coordination also means identifying the right domains and corresponding brokers for a certain resource allocation request (RAR).
- Configure ingress router of its domain if a request is accepted. Configuring ingress router means dynamically marking the traffic as EF and policing to limit the volume of traffic.

The simple Service Broker implementation in chapter 3 addressed all these issues except the signaling mechanism required for coordination with other Brokers to make dynamic VLL reservation over multi-DiffServ domains.

6.2.3 Service Level Agreements

Service Level Agreements are generally contracts between network service providers and their customers to guarantee particular quality of service levels for network performance. SLAs exist between a customer and its provider (called

intra-domain or customer-ISP SLA) and also between two providers (called inter-domain or inter-ISP SLA).

A customer normally has a contract with the local ISP regarding the maximum amount of traffic he can send or receive for a VLL service. Such customer-ISP SLA, however, does not automatically guarantee that a customer will always receive the service upon request - it only indicates the upper limit of the request and successful reservation of the requested VLL depends on admission tests at different points along the VLL. Referring to Figure 6.1, if a customer wants to establish a VLL between stub network A and C, an intra-domain SLA would suffice. However, for a VLL to be established between stub network A and B and inter-domain SLA between domain 1 and 2 must be in place. Based on inter-domain SLA the sending domain can send a maximum, say X Mbps aggregated traffic, to a receiving domain, and ensures that it does not send more than X Mbps by shaping at the outgoing interface connected to receiving domain. The receiving domain polices the incoming traffic at the ingress router's inbound interface to make sure that the sending domain does not send more than X Mbps. As shown in Figure 6.2, domain 1 aggregates all the individual requests and shapes them to 10 Mbps at router R2's outgoing interface and domain 2 sets a policing rate at router R3 incoming interface so that no more than 10 Mbps can enter its domain. This is the inter ISP SLA between domain 1 and 2. It can be assumed that the interior of the receiving domain is QoS rich (pre-provisioned to support enough EF marked traffic) and 10Mbps traffic will be transported to various destinations where those destinations could be in the receiving domain (for example, a stub network connected to R5) or domains connected to it (for example, domain 3 or 4).

6.2.4 Service Provisioning and Call Admission Control

For the establishment of a new VLL, the Bandwidth Broker needs to determine the resources required for quantitative VLL traffic at each node along its path from ingress to egress. Although an ISP naturally knows from the SLA the amount of quantitative VLL traffic that will enter the transit network through a specific edge node, this volume cannot be estimated with exact accuracy at various interior nodes that will be traversed by VLL connections. However, known routing topology and ingress-egress points can help in the estimation process, and hence in admission control decision. For simplified provisioning and admission control we assumed the following:

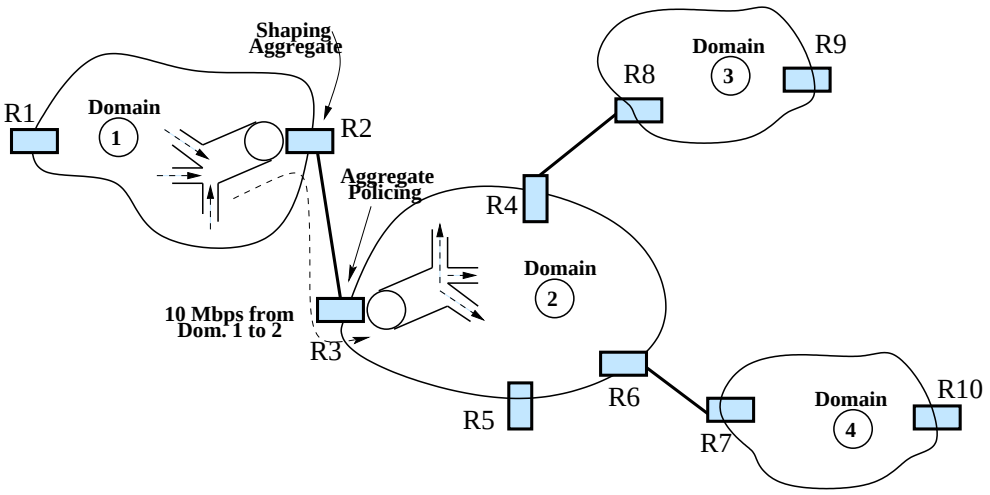


Figure 6.2: Illustration of Inter ISP Service Level Agreements

- Pre-configure interior and other border routers with scheduling mechanism like Priority Queuing (or CBQ, WFQ) so that traffic marked as EF are served by high priority queue.
- Traffic follows a known path.

If the domain is QoS rich, for a simple model it might suffice only to perform CAC at the edge points. For a more sophisticated model, considering the necessity of interior provisioning the BB may also check the availability of resources at the interior points that would be traversed by a VLL. In such a case, virtual core provisioning [KB00a] might be suitable that only requires a capacity inventory of interior devices to be updated based on VLL connection acceptance, termination or modification. This can be best explained if a Diffserv domain (or collection of domains) is viewed as One-Way IP super-highway (Figure 6.3).This highway can be imagined as serving EF marked traffic only and ISP knows the maximum traffic that can pass through specific nodes as those nodes are supposed to be pre-provisioned.

6.2.5 End-to-End Signaling

A user sends a request to the Bandwidth Broker that maintains the user's ISP domain. The request contains source and destination addresses and also the

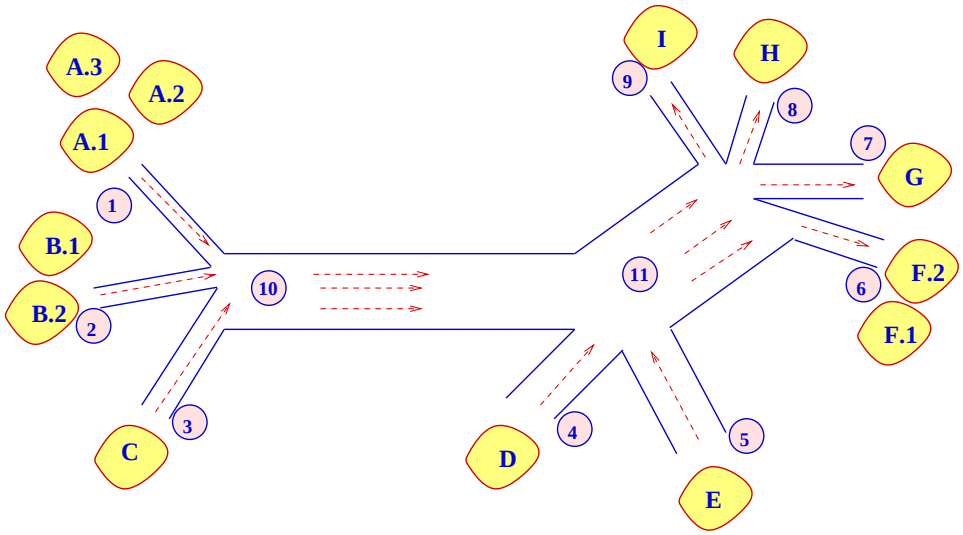


Figure 6.3: Provisioning of the Diffserv Networks Viewed as One-Way IP Super-Highway.

requested bandwidth. While the source naturally resides in the stub networks attached to the ISP's network, the destination might well be in the stub network that is attached to another ISP's domain. That domain might not be the final domain and there might be one or more domains in between. If both the source and destination addresses are in the stub networks of the same ISP domain, the Broker that maintains the domain can find the ingress and egress routers by some simple lookup in the various Broker related databases (explained in the next section). The Broker performs admission control at the points (ingress and egress) before deciding whether the request should be granted or not. If the destination is in another domain other than the source domain, then the Broker must identify the followings:

- the domain that has the destination stub connected to it.
- intermediate domains (if any) that will be traversed by VLL connection if the request is accepted.

Before we investigate the above two issues it would be useful if we give a brief overview of Resource Allocations Requests.

BB Resource Allocation Request Format

A Resource Allocation Request (RAR) may come from an individual user to a BB or from one BB to another neighbor BB. We call the first one intra-domain RAR while the latter one is referred as inter-domain RAR. Their formats are:

- **intra-domain RAR**: To setup a new VLL this request contains user id and password, source and remote addresses of the VLL to be established and the bandwidth required for it:

  newflow -u userid -p password -s source -d remote -b bandwidth

- **inter-domain RAR**: Inter-domain RAR is automatically generated by a broker when it detects that the remote address indicated in the request is not attached to its domain. The request is then sent to another neighbor domain. Since the actual requester is a domain broker, the recipient broker needs to check its validity as an inter-domain request.

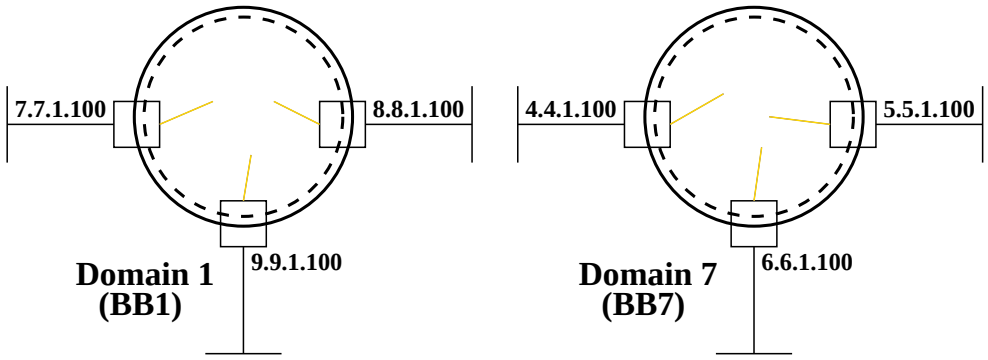
  newflow -bb brokerid -p password -s source -d remote -b bandwidth -tbb final_domain

Domain Identification

A scalable and simple way for each Broker would be to send boundaries of the domain that it maintains to other cooperative domains. By boundaries we mean the IP addresses of the edge devices. Let us consider domain 1 and 2 in Figure 6.4 where each of the domain is actually constituted from several edge devices. All these edge devices have unique IP addresses. If we can identify an edge router by the destination IP address in the RAR, then we can readily identify the domain, and hence the Bandwidth Bandwidth Broker that represents the domain. For example, when a user wants to establish a VLL to send traffic from any of the stub networks 7.7.x.x to one of the sub networks 5.5.x.x, Broker BB1 can easily identify that it has to finally communicate with BB7 by reading a domain identification database.

Bandwidth Broker Message Forwarding

When the Bandwidth Broker identifies the Final Broker there might be one or more intermediate Brokers that need to be contacted as well for end-to-end capacity reservation. How does Broker (first Broker or any intermediate Brokers)



Stub Networks	Bandwidth Broker
7.7.0.0	BB1
8.8.0.0	BB1
9.9.0.0	BB1
4.4.0.0	BB7
5.5.0.0	BB7
6.6.0.0	BB7

(c)

Figure 6.4: (a) Domain 1 (b) Domain 2 (c) Domain Identification Database

determine the next appropriate Broker when there are several neighbor Brokers and a VLL needs to established over several domains? In the previous example the VLL needs to be established over domains that are managed by BB1, BB2 and BB7. If each Broker knows the neighbor brokers and by exchanging that information every Broker can build a message forwarding table as shown in Figure 6.5(c) and 6.5(d). From the table it is obvious that BB2 is the intermediate broker that needs to contacted first by sending an Inter-domain RAR from BB1 before BB2 finally sends another inter-domain RAR to BB7 on behalf of BB1.

6.3 Multi-Domian BB Implementation

6.3.1 Layered Implementation Architecture

Based on the requirements for end-to-end capacity reservation the Bandwidth Broker has been developed to dynamically establish VLL on customer's request.

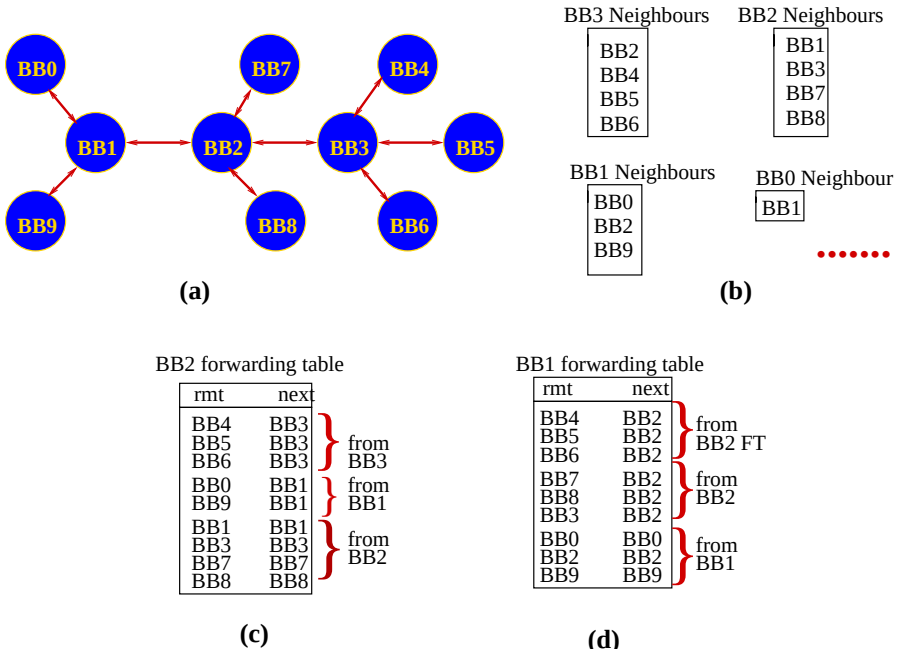


Figure 6.5: (a) Several Diffserv Domains Represented by Bandwidth Brokers (b) Neighbors Tables of some Brokers (c) BB Message Forwarding Table in BB2 (d) BB Message Forwarding Table in BB1.

Our earlier analysis and functional requirements of BB resulted in a four layer implementation architecture of Figure 6.6. The top layer is responsible for validating both intra- and inter-domain requests. The two middle layers are composed of several databases that are invoked for admission and signaling purposes of valid requests. The bottom layer decides and configures edge routers based on processing of requests in the three above mentioned layers. In the subsequent sections we will describe the components of these layers.

6.3.2 The Databases of the Bandwidth Broker

Of the several databases that are required for a Bandwidth Broker to work in multi-ISP scenario some of them have already been discussed in chapter 3. They are:

- **customer-ISP SLA database**: *<User ID, Password, Maximum BW in Mbps, Source Stub Address, Remote Stub Address>*

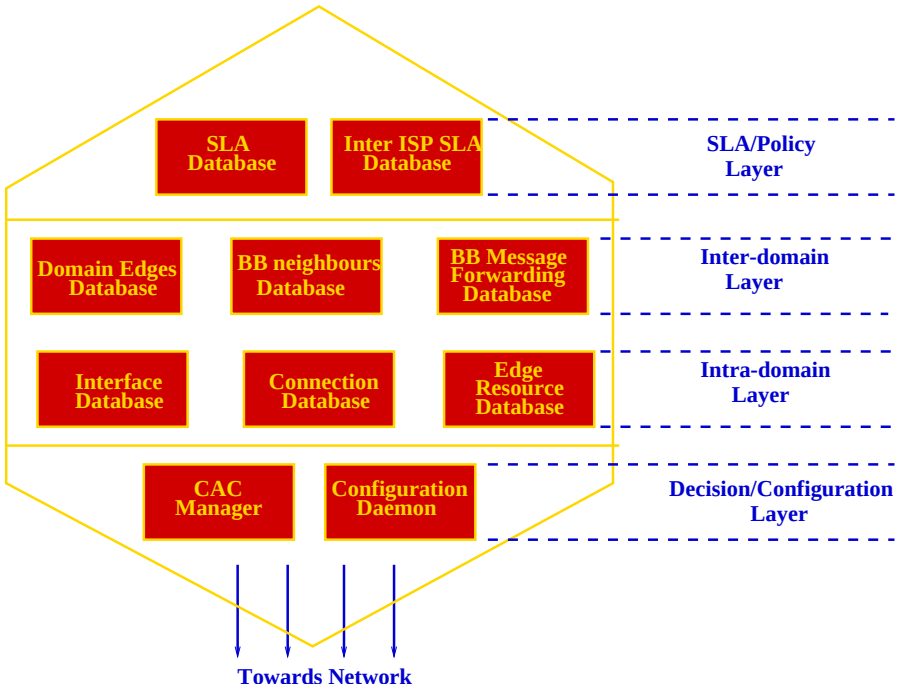


Figure 6.6: Layered Implementation Architecture: Components of a BB for Resource Provisioning, Admission Decision and End-to-End Signaling

- **interface database**:< *stub network, edge router, generic router name, inbound interface, out-bound interface* >
- **connection database**: <*user id, source address, VLL ID, rmt address, bandwidth, activation time*>
- **edge resource database**:< *edge router,* C_{TOTAL}, $C_{allocated}$ >
- **VLL ID database**:< *edge router, Tunnel ID, Status* >

The additional ones necessary for two BBs from two different domains to communicate and exchange SLAs are:

The **inter-ISP SLA database** is invoked by a domain when it receives inter-domain RAR. By doing so the receiving domain can check the validity of the request sent by the sending domain. Here this validity means identification of the sending domain and the maximum amount of bandwidth it can reserve on a certain edge interface in the receiving domain that is directly connected to it.

<Domain ID, Domain Password, Maximum BW>

The **BB Neighbor Database** holds records of neighbor Bandwidth Brokers IP addresses as well as IP address of the router interfaces (both in-bound and out-bound) that interconnect the peer domains.

< Neighbor BB, InsideInterface, OutsideInterface >

The **Domain Edges (or Identification) Database** holds records of the networks that reside at the periphery of a domain. Its purpose is described in details in the previous section. An example entry of this database is also shown in Figure 6.4.

< Stub Network, BB >

The **BB Message Forwarding Database** contains next hop BB's IP address to send resource allocation request to final Remote Bandwidth Broker.

< Remote BB, NextBB >

6.3.3 CAC Manager and Configuration Daemons

CAC manager is part of functional engine of BB that basically invokes the databases described above and decides whether an incoming request can be granted resources or not by performing simple admission control (as described in chapter 3) at various network nodes. *Configuration Daemons* are intelligent provisioning agents that are able to translate user requests and BB generated pseudo rules into device specific rules to configure the routers/switches since we might have several different devices from various vendors.

6.3.4 Inter-domain Signaling

From earlier sections we have seen that a Bandwidth Broker not only receives RARs from a customer of its own domain or other BBs, but also sends RARs to neighbor BBs. Therefore, we have designed a Bandwidth Broker that consists of server and a client Socket program. When a Broker's Server receives a request from a client and finds itself to be the final destination BB it can convey the CAC decision back to the client, otherwise the Server tells the client of that Broker to talk to the appropriate neighbor Broker's server. So, there is a chain of communications which are handled by client-server concatenations. This is shown in Figure 6.7.

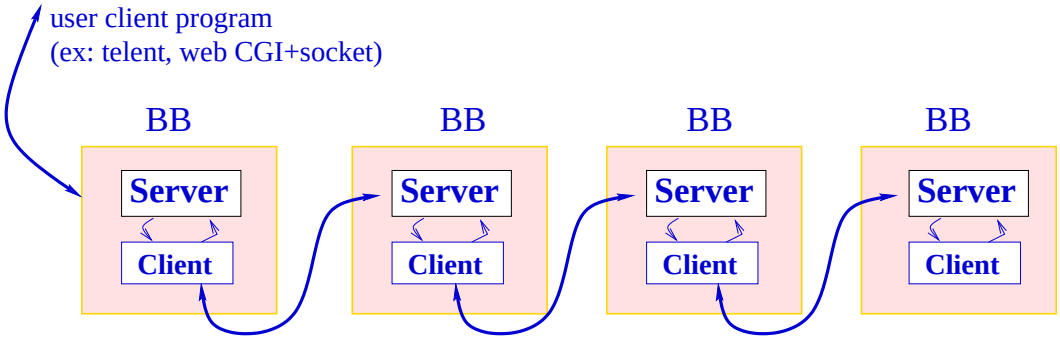


Figure 6.7: Client-Server Concatenation for Inter-Domain Signaling

6.4 Operational Details and System Flows

Similar to chapter 3, we describe the system flows of VLL connection establishment or termination in both intra and inter-domain cases. Although intra-domian QoS-VPN activation was discussed in chapter 3, the interactions with the system databases are different.

6.4.1 Dynamic VLL Establishment

Figures 6.8 and 6.9 show all the communications involved in setting up a VLL connection between two stub networks or simply between an originating host and a remote host. Both intra and inter-domain cases are explained. Although an intra-domain scenario is not the focus in this chapter, yet we describe it because its similarities in system dynamics with an inter-domain case, and many of the communications involved in an intra-domain scenario are actually repeated in the latter one. We will start describing the operational details by referring to the communications marked on Figure 6.8. Considering each communication in turn:

- 1) A user sends a VLL connection request message to the BB via http or other interfaces able to communicate to the BB server.

- 2,3) The BB contacts the customer SLA database that is responsible for validating the user and his request. If the user is identified correctly, his source and remote address conforms the contract, and also the bandwidth requested is less than or equal to the agreed traffic contract, it proceeds further.

- 4,5) The BB contacts the configuration daemon to check its status. The status can be busy, available, or down. Only in the case of availability the user request can be processed further.

- 6,7) The BB contacts the connection database to check the existence of an exactly similar VLL. This is because for a source and destination pair only one VLL can remain active.

- 8,9) The BB reads domain edges database to find out whether the VLL is needed to be created only in the domain under its supervision or might well span over other autonomous domains.

Intra-domain VLL Setup

If (Figure 6.8(a)) the BB finds that both source and destination are in the same domain,i.e. the VLL is needed to be created over a single domain, it proceeds as follows :

- 10,11) The BB reads the interface database to find out ingress and egress edge routers. One or both are configured depending on a traffic contract.

- 12,13) Once the edge routers are detected from the interface database the BB communicates with the resource database and performs admission control on certain router interfaces to allocate a VLL of the requested amount. It might perform admission control on only the appropriate edge router interfaces or even on the interior routers interfaces that can be detected from the topology database. The resource database responds to the BB and either allocates the resource or denies based on resource availability.

- 14) The BB tells the configuration daemon to create appropriate configuration scripts. It is to be noted that a configuration script is created only for the ingress edge router because this is the only router that is configured to mark and police the incoming traffic. In the case of double edged SLA the egress router is configured as well for allocating QoS in the other direction. In the meantime, the resource and the connection database update their records. Another point to note is that the BB also fetches a VLL ID from the VLL ID database that is unique for each VLL and needed while configuring the router. The new connection request data is appended to the connection database and the VLL ID that has just been allocated from the VLL ID database is marked as used.

- 15,16) The CD puts a busy signal on itself and creates the routing scripts. It then sends configuration scripts to the routers. The routers send signals to the CD.

- 17,18) The CD removes the busy signal from itself and sends acknowledgment to BB which sends a notification to the user.

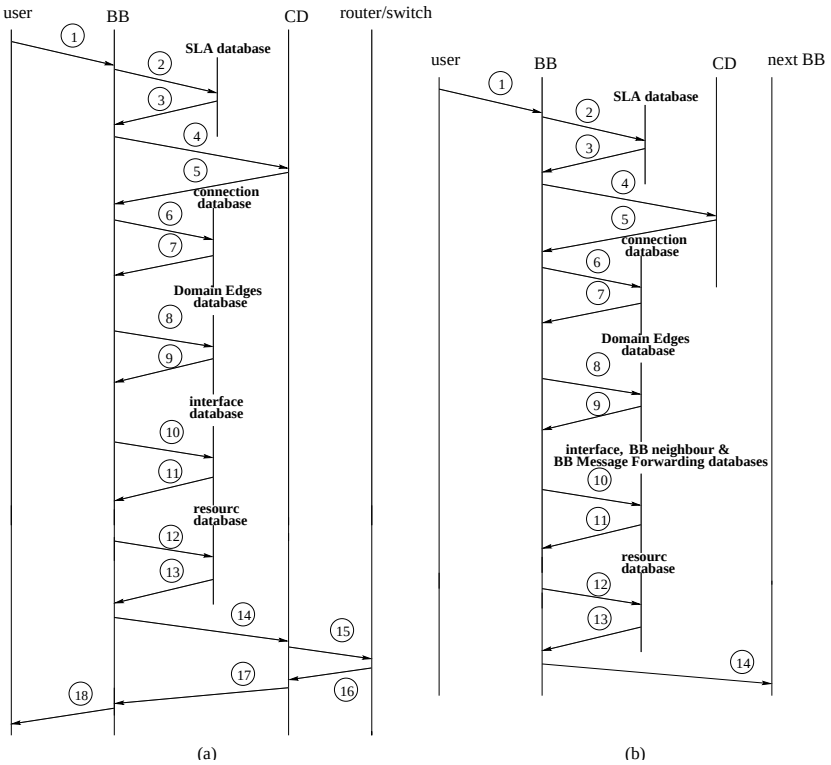


Figure 6.8: VLL Establishment System Flow: (a) Intra-domain Case, (b) Inter-domain Case

Inter-domain VLL Setup

In the case a VLL (Figure 6.8(b))is supposed to be established over several Diff-Serv domains the BB follows the steps described below:

- 10,11) Once the final destination domain has been determined the Bandwidth Broker finds out the next hop BB by reading the BB Message Forwarding Database. Now a search in the BB Neighbor Database gives current

domain's outgoing interface towards the next hop BB. The BB also fetches the appropriate ingress router interface from the interface database.

- 12,13) These steps are similar to the steps 12 & 13 in the previous case. As the BB now knows the ingress and egress router interfaces, it performs admission control on those interfaces.

- 14) A positive CAC response leads to sending an inter-domain RAR to the next hop BB.

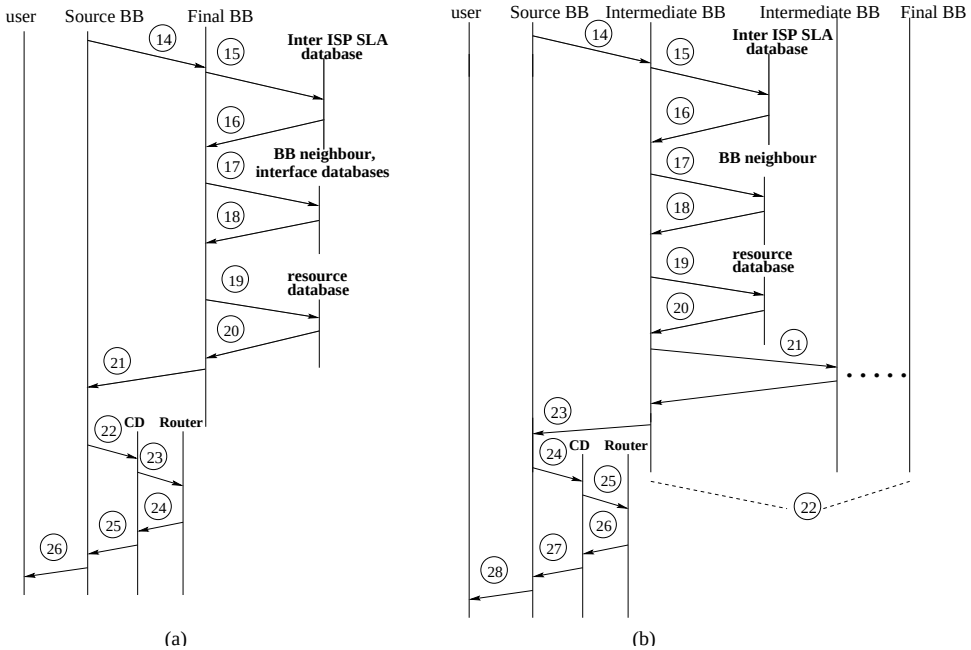


Figure 6.9: VLL Establishment System Flow: Inter-domain Case cont'd. (a) Next Hop BB as Final BB (b) Next Hop BB as Intermediate BB

Next Hop BB as Final BB

If the next hop BB finds that the destination stub is in the domain maintained by it, the following steps are followed (Figure 6.9(a)):

- 15,16) Upon receiving an inter-domain RAR the next hop BB contacts inter-ISP SLA database to check the validity of the request.

- 17,18) It reads the BB Neighbor and interface databases to identify ingress and egress interfaces.

- 19,20) BB contacts the resource database and performs admission control on the previously identified ingress and egress interfaces.
- 21) The BB sends CAC decision to the sender BB.
- 22-26) If the response received by the sender BB is positive then it contacts the appropriate configuration daemon to configure the ingress edge router. These steps are essentially the same like what we have seen in steps 14-18 in the intra-domain case.

Next Hop BB as Intermediate BB

The behavior of an intermediate BB is similar to that of a final BB with the exception that this one generates an inter-domain RAR based on positive CAC response from the resource database . The RAR is sent to the next BB which might be another intermediate BB or a final one. Figure 6.9(b) illustrates this case.

6.4.2 VLL Termination and VLL Request Rejection

The VLL termination process involves the followings:

- The VLL connection entry is deleted from the connection database of the origin domain. Only the ingress edge router is configured to reflect the connection release.
- The resource databases are updated in all the domains that are traversed by the VLL, i.e. as resources are released $C_{allocated}$ is update as $C_{allocated} + C_{vll}$ where C_{vll} is the capacity of terminated connection.

A VLL request is rejected if

- the user's SLA profile does not match in the origin domain, or in the case when inter-domain RAR is sent from one domain to the next neighbor domain but interISP SLA profile of ISP that sends RAR does not match in the received domain.
- VLL connection already exists in the connection database of the origin domain.
- Admission control fails in any of the domains that are traversed by the VLL.

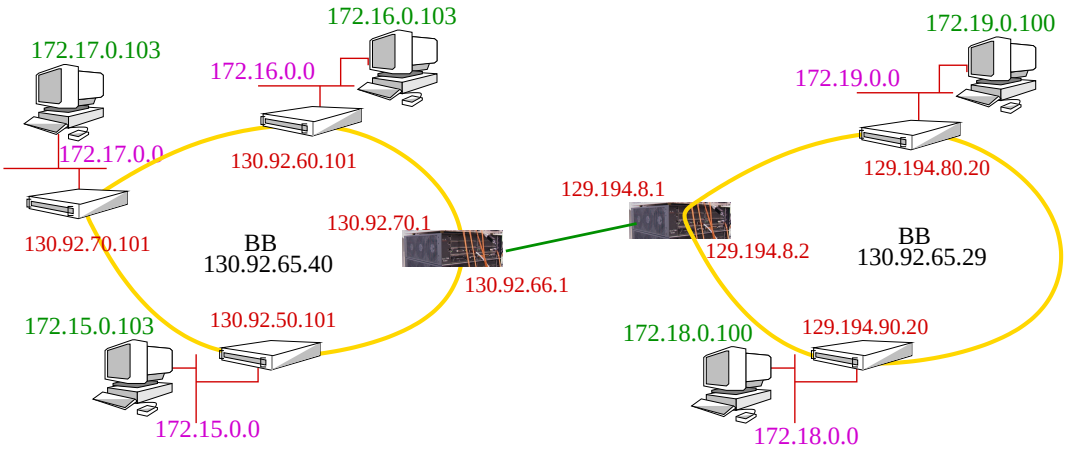


Figure 6.10: Experimental Setup for Demonstration of Dynamic VLL Creation over Multiple Domains

6.5 Examples of Dynamic VLL Setup

The topology we used to run some experiments over the public SWITCH [SWI] network between Bern and Geneva is shown in Figure 6.10. We have two domains with several end-systems that have private addresses and all these machines are connected to routers having public IP addresses. The domain in Geneva is represented by Broker 130.92.65.29 and the domain in Bern is managed by Broker 130.92.65.40. We also statically created VPN tunnels between these private stub networks so as to allow transparent connections between them. A Bandwidth Broker is only expected to dynamically configure the ingress edge router assuming that the routers along the way from source to destination have been pre-configured with CBQ or WFQ.

Based on the setup as shown in Figure 6.10 we will now explain how a VLL is established over several Diffserv domains (Figure 6.11, 6.12). Assume that user *ibrahim* plans send traffic from 172.17.0.103 to 172.18.0.100. The broker 130.92.65.40 receives the request as: newflow -u ibrahim -p ****** -s 172.17.0.103 -d 172.18.0.100 -b 3. As the Broker realizes that 172.18.0.0 is in domain Geneva it performs admission control in its domain (i.e. at 130.92.66.1) and then sends an inter-domain RAR to 130.92.65.29 in form of newflow -bb 130.92.66.29 -p ****** -s 172.17.0.103 -d 172.18.0.100 -b 3 -tbb 130.92.65.29. When the broker 130.92.65.29 receives this request it knows that the request has come from another neighbor Broker (because of the tagging -bb) and therefore checks the interISP

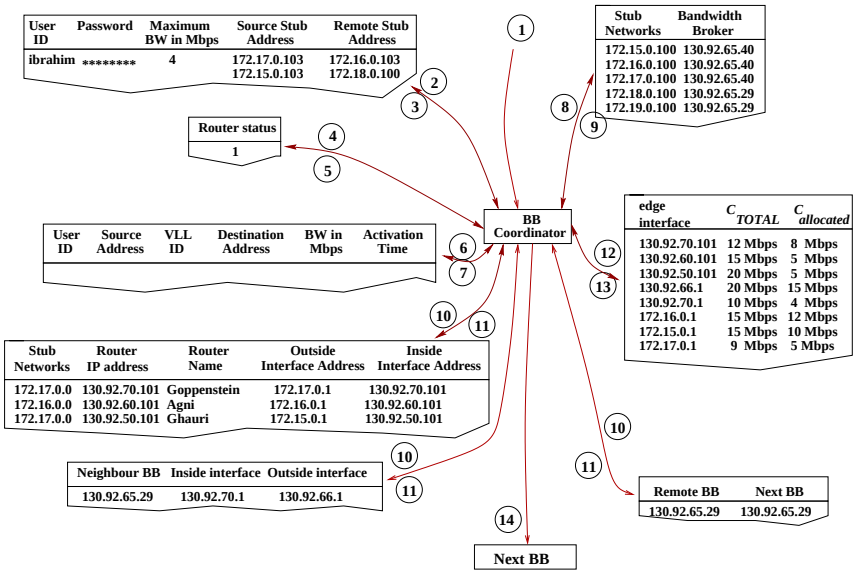


Figure 6.11: An example of VLL Setup in Multi-Domain Scenario.

SLA database to check the validity of the request. The Broker also identifies that 172.18.0.0 is located in the stub network attached to its domains since this is the final domain (from -tbb 130.92.65.29). While running trough the steps as described in the previous section it identifies the ingress and egress router interfaces to be 129.194.8.2 and 172.18.0.1, performs admission control on those and finally conveys the decision to the sender Broker 130.92.65.40. Upon receiving the decision Broker 130.92.65.40 talks to the VLL ID database to pick up an ID, configures the edge router 130.92.70.101, and then conveys an acknowledgment back to the user.

6.6 Conclusion

In this chapter we have described the implementation of a Bandwidth Broker that uses a simple signaling mechanism to communicate with other cooperative Brokers to enable customers to dynamically create VLLs over multiple Diffserv domains. We have presented a simple approach to make advance reservations in the absence of senders or receivers in a multi-domain scenario. Rather than using RSVP or COPS in inter-domain signaling to reserve capacity across domains, we used a novel method to identify domains, and hence Bandwidth Brokers that are

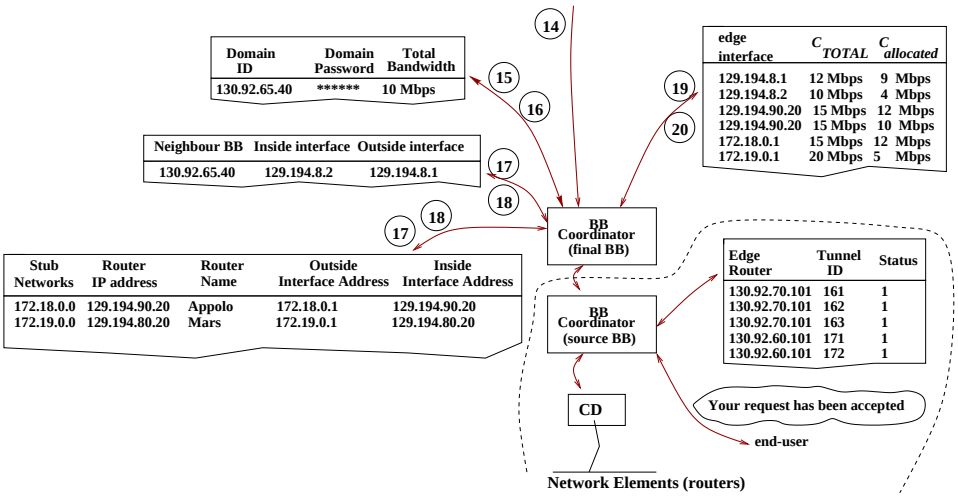


Figure 6.12: Multi-domain VLL Setup Example Cont'd from Figure of 6.11

responsible for maintaining them. A detailed implementation of the system and its operational details and some practical examples show how a simple resource reservation can be made dynamically over several cooperative Diffserv capable domains. Further simulation work might be useful to examine the scalability and effectiveness of our approach and is a topic of future research.

Chapter 7

Intelligent Provisioning in Large Networks

The Service Broker implemented in chapter 3 to dynamically configure and manage QoS-VPNs uses simple push-based configuration delivery mechanism where configuration is generated first from a policy template and then loaded to the network devices using agents. However, using a simple push-based model to configure network services while ignoring dependencies among configuration elements may easily lead to configuration inconsistencies resulting in failure or inefficiencies. In this chapter, we have taken a new approach to configuration modeling that is device neutral and based on which any emerging IP services can be presented by encapsulating service semantics, including service-specific data. We have developed new mobile intelligent provisioning and audit agent architectures that use the knowledge built upon configuration dependency modeling. To fully exploit the benefits of intelligent agents we propose a novel distributed architecture where highly mobile and intelligent agents can take the responsibilities of not only provisioning, but also configuration audit management in a timely fashion. Although recent research have focused on using mobile agents for network monitoring or simple push-based device configuration in a distributed architecture, their ability have not been exploited in dynamic IP service provisioning. Examples of intelligent agent are presented to complement the proposed management architecture.

7.1 Introduction

In large scale Internet service deployment scenarios Internet Service Providers (ISPs) and large enterprises often face the daunting task of provisioning huge number of network devices in a short period of time and continue ongoing device management. Manual provisioning methods and manual element interconnection across multiple discrete elements leaves a high margin for errors. Carriers and service providers are, therefore, looking to migrate from manual, static provisioning models to the more dynamic service-oriented automated provisioning models to meet customer demands for rapid service turn-up and obtain more customers and maximize revenue opportunities.

Existing Network management architectures ([MFZH99]) deploying SNMP [Sta98] as the management protocol do not really fit well to suit the needs of automated provisioning in a large scale environment due to some well known limitations. Not only the centralized approach of SNMP has severe scalability limitations, but it is also a less powerful tool for making modifications to the network. SNMP Management Information Base (MIB) implementations for emerging services like IPSec [KA98], MPLS [FWD+01], DiffServ [BBC+98] do not even exist in most vendors equipments to enable SNMP `set` command for dynamic configuration. Attempts have been made ([Slo94], [KK95], [LMB97], [KLMB96],[SA98]) to address the scalability problem by decentralizing processing and control to distribute processing load and reduce the traffic around the management station. Most of the efforts have been made by using intelligent and mobile agents [HB99] that are able to perform management functions by carrying code, instructions,data and executing tasks on any network node. However, while addressing only scalability issues the vast majority of these agents [PT99], [BGP97], [GY98], [CCLM99], [BPW98], [EDB99], [Gün01] have been mainly used in traffic analysis, fault management, network monitoring and performance management.

Despite of the ability of mobile agents in numerous fields, in the area of network device configuration management [CDF+95], [BCS00], [SA98] and automation they have played only a limited role. In this regard, they mostly have been found up-loading management scripts that were compiled at machines hosting agents [SQK00]. The Service Broker implemented in chapter 3 to dynamically configure and manage QoS VPNs uses simple push-based configuration delivery mechanisms. In reality, many of the dynamically created services at least partially depend on the current configuration states of the target devices. Using a simple push-based model to configure network services while ignoring dependencies

among configurations elements may easily lead to configuration inconsistencies resulting in failures or inefficiencies. This often neglected but important issue of configuration modeling was however addressed in [YKF00] with limited use in useful and complex IP service creation.

In this chapter, we have taken a new approach to configuration modeling that is device neutral and based on which any existing or emerging new IP services can be presented in abstract form by encapsulating service semantics, including service-specific data. We have developed a new intelligent mobile provisioning agent architecture that uses the knowledge built upon configuration dependency modeling. We have also proposed a hierarchical look like high level network management architecture that has a central policy and data repository and deploys several intelligent agents playing the role of managers. The agents can take the responsibilities of not only provisioning but also perform ongoing device management often termed as *auditing*. The basic objective of audit management is to ensure that the expected configuration as stored in the policy repository is equivalent to the actual configuration in a physical device. This is necessary because accidental manual mis-configuration or configuration performed by any external process other than the automated system can easily result in unexpected behaviors of network devices. Intelligent audit agents periodically sent to visit network devices have procedures in place to reestablish lost service. Several real world examples of intelligent provisioning are presented to show the applicability of our approach.

7.2 Provisioning Requirements in Large Networks

As new IP services like VPN, QoS, MPLS are emerging, and network management getting more complex, the growing trend among corporate customers is to out-source such complicated management services to Internet Service Providers (ISP). They now see challenges in deploying and managing the network services efficiently and cost effectively while meeting the demands of customers. It is not unlikely that a single ISP might need to timely manage thousands of network devices. The effective way to meet such challenges is to provide automated service provisioning. While we addressed this issue of automation by implementing Service Brokers in previous chapters, we realize that in a large-scale environment the ability of the system to process a large number of transactions is enhanced using multi-threading in service components. Managing devices not only means one

time service activation for customers, but also providing ongoing device management often termed as audit management. Ongoing device management is needed for several reasons such as:

- Services configured by human administrators many lead to misconfiguration or cause conflicts with existing services activated via automated systems.
- If network devices crashes due to power failure or for some other reasons, valuable configuration information that were thrown into the devices might well be lost partially or fully.
- One can not rule out the possibility of an inside or outside intruder changing some configuration settings.

In summary, configuration performed by any process other than automated system can easily result in malfunctioning of the managed services in network devices. If the Service Level Agreement (SLA) states that should any such problem occur that it would deal within x minutes of the occurrence, then the automated system would need to send a delegated entity capable of fixing the problem in a time period close to x. That delegated entity, if we term that as an intelligent agent, should have the knowledge and ability to perform audit management in an appropriate manner. Such agents will need to understand the complete dependencies in a configuration storage of a network device while provisioning a new service or modify an existing one.

7.3 Distributed Provisioning and Audit Architecture

The proposed hierarchical look-like network management architecture shown in Figure 7.1 addresses the provisioning requirements mentioned above in a large-scale environment. In such an environment, with a large number of different types of network devices, service providers will need to deploy a network architecture that can use a huge number of automated agents acting as managers to provision and perform on going device management and provide distributed management access to the large number of potential customers that want to create and modify services dynamically on demand. The proposed architecture simply aims to achieve this. It comprises several management interfaces, a central repository

system with agent communicator, and several service provisioning agents mainly performing the role of managers. Having several management interfaces not only facilitates dynamic service creation by the customers that out-source services to ISPs, but also allows several system administrators to manage a large enterprise or ISP network from a network operations center or distributed locations.The agent coordinator performs no management tasks, but only delegates service requests to the provisioning agents. Therefore, the agent coordinator does not become the processing bottleneck.

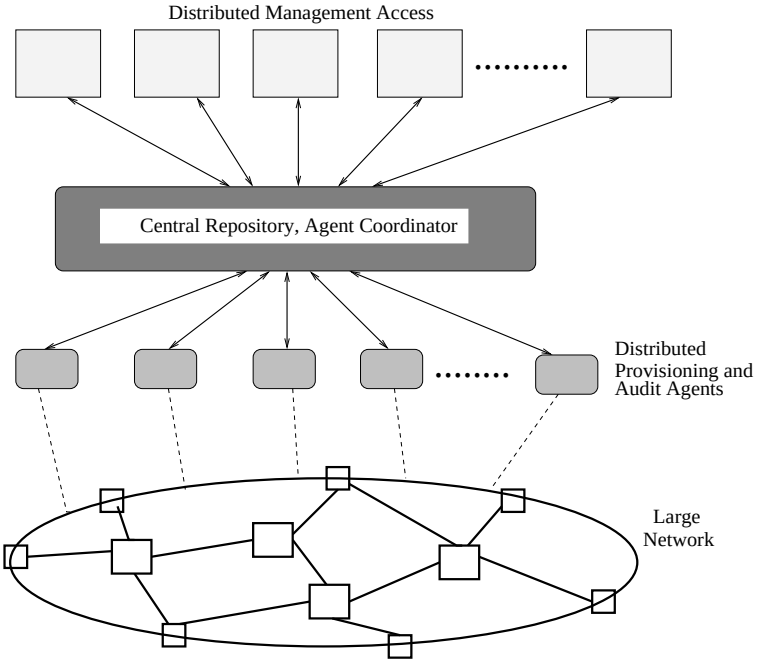


Figure 7.1: Distributed Provisioning and Audit Architecture

The service-specific intelligent provisioning agents build a generic service environment to enable customization of service creation and control of network resources. By encapsulating service semantics, including service-specific data and the logic to interpret the data in service provisioning agents we can provide an abstract network interface, separating services from the underlying network. As shown in Figure 7.2, these agents are able to translate user requests and pseudo rules extracted from data/policy repository into device-specific configurations to automate service provisioning process.

By using intelligent provisioning agents we address both the scalability problem

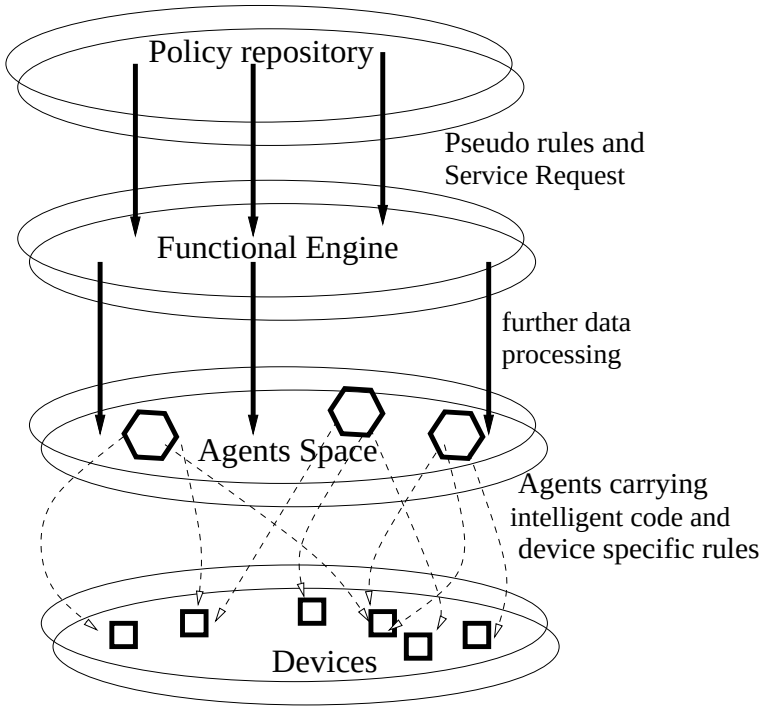


Figure 7.2: Intelligent Provisioning Agents Carrying Code and Device Specific Rules

and the lack of ability of traditional network management approaches to automatically provision a very large network. The scalability problem is well handled by the agents' ability to decentralize processing and control, and, distribute processing load to reduce the traffic around the management station. Provisioning agents are able to filter and process data locally at the network nodes without the need for transmission to the Top Level Manager (TLM) like agent coordinator. In fact, traditional distributed management architectures ([MFZH99] do not help much to alleviate the scalability problem as the management repository is partitioned and replicated, and thus another protocol is required to maintain the consistency of distributed repositories. Unfortunately, perfect consistency is impossible to achieve making such approach less effective in time-sensitive distributed provisioning.

In this architecture, by using mobile agents we have not only provided decentralization, but by embedding intelligence we have also added dynamism to the agents that are able to perform configuration and audit even in hostile situation when actual device state might be different from expected device state. We

build intelligent provisioning agent as an autonomous software entity embedding mobile code and configuration data that has the capability of moving itself in the network and executing onto specific network nodes. On a network node, the provisioning agent can analyze and retrieve local configuration information, install and execute code, take decisions, thus leading to fully decentralized network management activities. This obviates the need for MIB implementation in the network devices. None of the previous works [PT99], [BGP97], [GY98], [CCLM99], [BPW98], [EDB99] have considered performing such level of complex provisioning and audit management.

7.4 Configuration Modeling of Network Devices

Dynamic IP service creation in network devices relies on the fundamental assumption of having an accurate and consistent abstraction of the underlying network, particularly a view of essential parts of configuration information stored in those devices spread across different device-specific repositories in different formats. The complexity of large IP networks and the scarcity of commercial configuration tools means that this is often an unrealistic assumption. Hence, populating a service model must be closely coupled with checking for possible configuration mistakes. Relationships between different configuration elements are implicit with repositories containing replicated and interdependent configuration information leading to inconsistency. Many of the dynamically created services at least partially depend on the current configuration states of the target devices. Using a simple push-based model to configure network services while ignoring dependencies among configurations elements may easily lead to configuration inconsistencies resulting in failure or inefficiencies.

In this section, we present a new device neutral approach to configuration modeling and based on which any existing or emerging new IP services can be presented by encapsulating service semantics, including service-specific data. We present a generalized view of configuration information in a network device and try to model it as clusters of configuration elements. For this purpose, we show examples of IPSec VPN configurations (Figure 7.3) in a widely used Cisco IOS [Sit]device. Our previous experience [KBG00], [KB00b], [BGK01] with VPN provisioning in chapter 3 and 4, and also complexity of VPN configurations in various commercially available devices are the main motivations behind using these specific examples here.

```
crypto isakmp policy 1
 hash md5
 authentication pre-share
 lifetime 500
crypto isakmp key GENEVA-BERN address 129.194.90.20
crypto ipsec transform-set ah-md5-hmacANDesp-des ah-md5-hmac esp-des
crypto ipsec transform-set ah-md5-hmac ah-md5-hmac
crypto ipsec transform-set ah-sha-hmac ah-sha-hmac
crypto ipsec transform-set esp-encryption esp-des
 crypto map genbern 160 ipsec-isakmp
 set peer 129.194.90.20
 set transform-set ah-md5-hmacANDesp-des
 match address 140
interface FastEthernet0/0
 ip address 130.92.70.102 255.255.0.0
 no ip directed-broadcast
 traffic-shape group 150 1000000 100000 100000 1000
 crypto map genbern
access-list 140 permit ip  host 172.17.0.102 host 172.18.0.100
access-list 150 permit ip  host 130.92.70.101 host 129.194.90.20
```

Figure 7.3: Partial IOS Configuration File of Cisco Router 130.92.70.102 as shown in Figure 7.4

Basically, no widely accepted standard for network configuration exists, and SNMP, in reality, simply remains a tool for network monitoring. Access mechanisms and manipulations of configuration information in device repositories are largely vendor-specific. While many devices are accessible via TELNET/FTP, others need to be accessed via HTTP or LDAP. The configuration modeling process is independent of the device access mechanism. However, the intelligent agents built upon the knowledge derived from configuration modeling use various device access drivers (i.e telnet driver, HTTP driver, etc.) as required.

7.4.1 Configuration Element

The configuration element is a customizable object with several attributes in a configuration space (i.e repository) of a network device. Configuration space, which we will describe later, represents the complete configuration information of a single device. The attributes of a configuration element reflect, how it would behave when embedded in a service and also how they can be linked to other elements to create a service. The attributes of a basic configuration element as shown in Figure 7.5 are:

- *body*: identifies the type of action a configuration element performs.

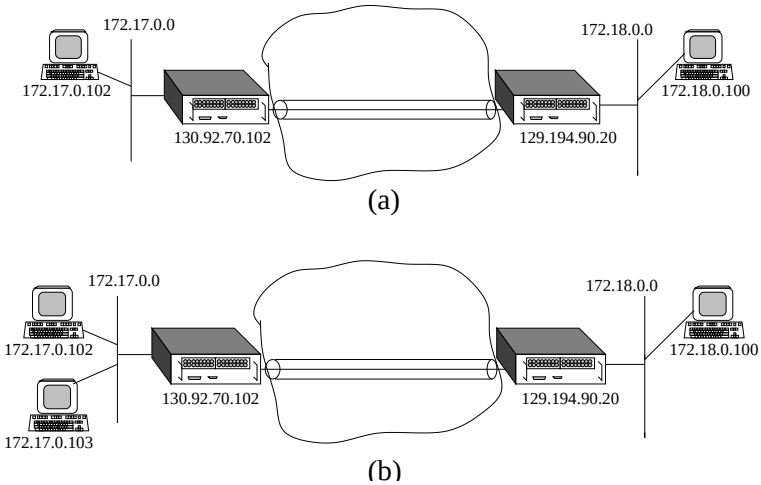


Figure 7.4: (a) Setup for VPN Network (b) Expanded Setup for VPN Network

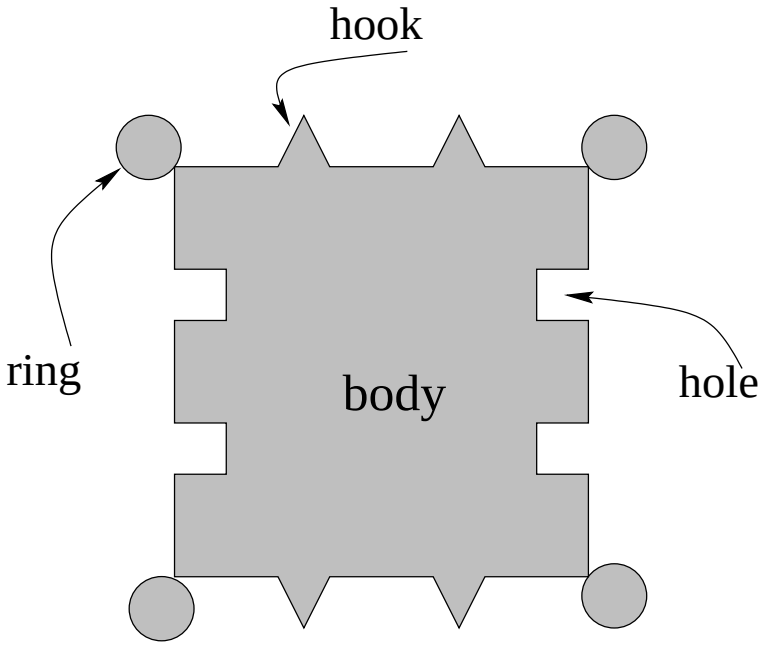


Figure 7.5: Configuration Element and its Attributes

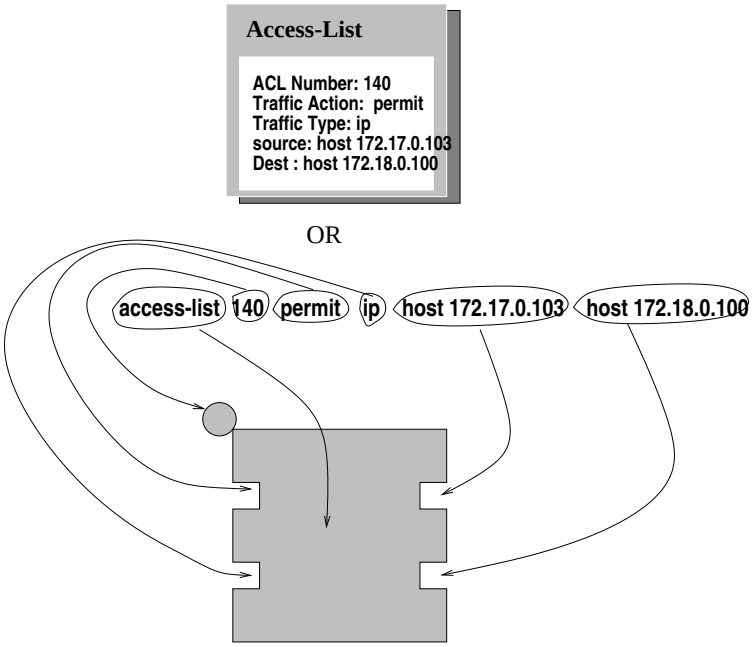


Figure 7.6: Modeling a Configuration Element in a Network Device

- *hook*: configuration elements are usually linked to each other in a configuration space. A hook helps to attach one element to the other.
- *ring*: is the identifier of a particular configuration element and it is that part to which hook of another element attaches in order to create a composite element or service.
- *hole*: defines the behavior of configuration element. Holes are filled up with parameters that actually determine the behavior.

Figure 7.6 shows an example of modeling a basic configuration (non-composite) element. This particular element classifies traffic based on source and destination addresses. Clearly, the body of the element is `access-list` as this identifies what it does (i.e classifies traffic). `permit` (action on traffic), `ip` (type of traffic), `host 172.17.0.102` (source) and `host 172.18.0.100` are the parameters of several holes of the element and determine its behavior; `140` uniquely identifies this element, i.e. acts as a reference to this. The element in the device can be viewed as a simple text line or can be presented via a GUI. Regardless of how it is viewed in heterogeneous devices, we can always model an element with the same sets of attributes.

7.4.2 Composite Element

A composite element is a collection of several basic configuration elements. This is created by hooking an element to the ring of another.

Figure 7.7 shows an example of modeling a composite configuration element relating to IPSec tunnels configuration in device 130.92.70.102 for the setup shown in Figure 7.4(a). Three non-composite configuration elements with body labels `access-list`, `crypto iksamp key` and `crypto ipsec transform-set` are linked to another element having a body `crypto map` that binds them with the hooks and create a composite element basically defining the tunnel configuration part in peer device 130.92.70.102.

7.4.3 Composite Service

When one or more basic or composite configuration elements are bundled together to perform a network service, the bundle is called the composite service, partial or full, depending on the nature of the service. For example, traffic policing based QoS configuration in an edge device can be a complete composite service while for a VPN tunnel service creation two network devices will have to be configured. Therefore, in a VPN tunnel service creation each device will have a partially complete composite service. The example shown in Figure 7.7 is a partially complete VPN composite service in network device 130.92.70.102.

7.4.4 Configuration Space

Figure 7.8 shows the whole configuration information of a single device that can be logically mapped to a configuration space viewed as clusters of cells, where each cell contains one or more of the configuration elements. However, a cell consists of only homogeneous configuration elements. For example, all `access-list` configuration elements as shown in Figure 7.3, can be viewed to be part of a cell having body tag `access-list`. Similarly, all the configuration elements having the body `crypto ipsec transform-set` are parts of another cell in the configuration space.

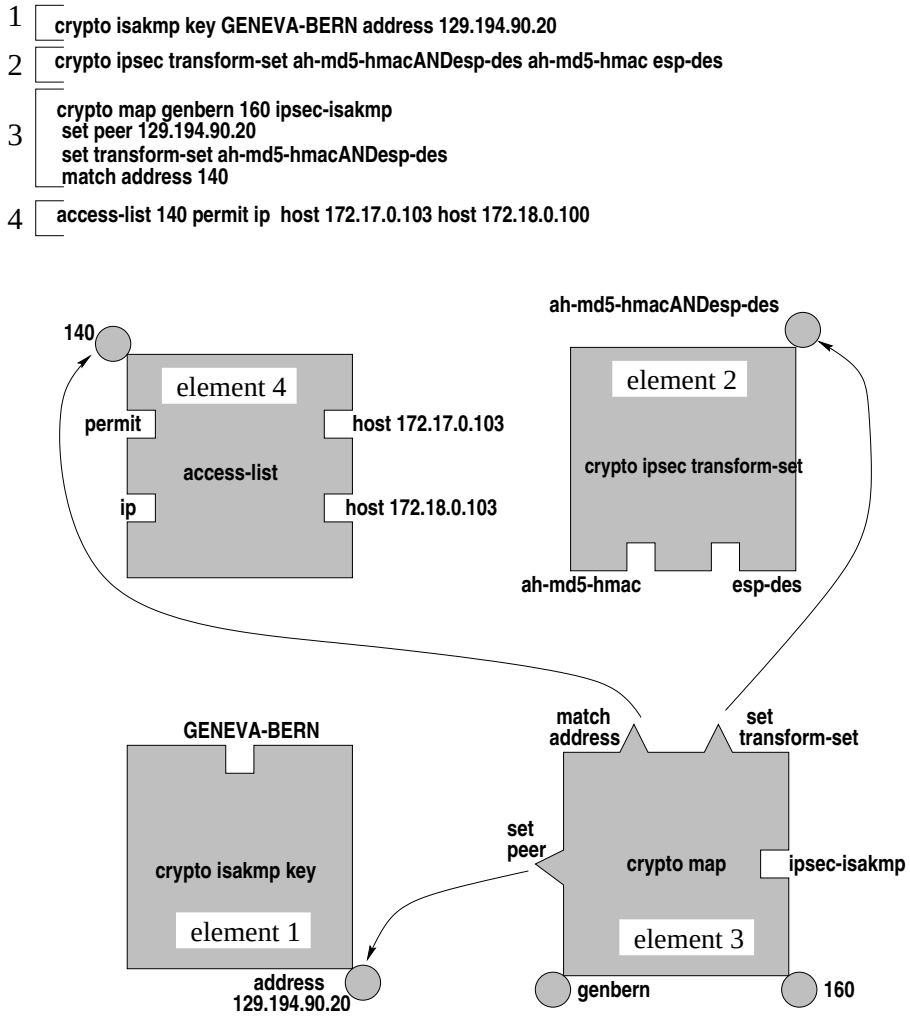


Figure 7.7: Example of Modeling a Composite Configuration Element

7.5 Intelligent Agents for Service Provisioning

The goal of our automated provisioning architecture is to develop distributed collections of intelligent software agents that operate mostly independently to perform a variety of service-specific configuration information retrieval and provisioning tasks in a network device. In this section, we develop an architecture for an intelligent provisioning agent whose service-specific intelligence is derived from configuration modeling and device-specific rules. The agents can translate

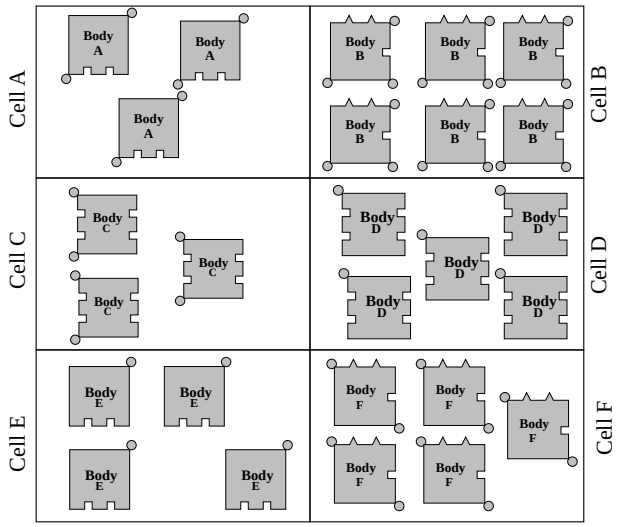


Figure 7.8: Mapping Configuration Space

user requests and pseudo rules extracted from data/policy repository into device-specific configurations to dynamically provision network devices.

7.5.1 Problem of a Conventional Provisioning Agent

Automated agents [SQK00] that exist today for provisioning network devices consider translating device-specific rules, but are not truly interactive and intelligent. These agents basically push configuration scripts to the devices and can be characterized as follows:

- These provisioning agents assume that since the scripts are correct in syntax the devices will be configured correctly. This always does not happen. The basic problem of this approach is that it does not consider the current configuration state in the devices. Consequently, potential conflicts might arise while an agent pushes a configuration script to the certain network devices. Also, this approach does not allow to provide an device independent repository.

- Most of the agents keep device state in an external repository and assuming that to be the actual device state at all times. However, human system administrators might as well manually configure those devices, in which

case, the external repository based device state no longer reflects the actual device state.

- Because such agents lack interaction, in case of a conflict no way remains to identify it.

7.5.2 Intelligent Agent Architecture

We considered the general problems of the conventional provisioning agents and have designed our agent architecture that does not suffer from the same problems. Figure 7.9 shows the internal architecture of our proposed intelligent provisioning agent. The main functional components of the agent follow:

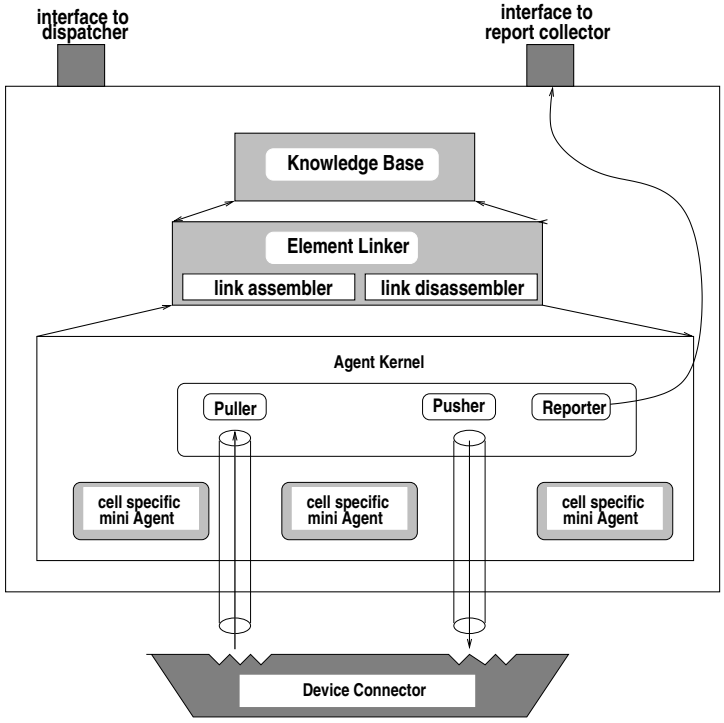


Figure 7.9: Intelligent Provisioning Agent Architecture

Knowledge Base

In section 7.4 we discussed about modeling a configuration element and mapping such elements in a configuration space. The main motivation was to to create

a knowledge base for a specific service. The service, for example, could be dynamic VPN service creation, or QoS configuration, or QoS enabled VPN, in which case both VPN and QoS configuration is necessary in a network device. For a particular service, the knowledge base, contains complete dependencies of configuration elements constituting the service, their composition in a configuration space including the device-specific rules.

Element Linker

This uses the knowledge base to create links between various configuration components to build a new service. Rings of some configuration elements are attached to the hook of another one to create links. The link assembler does this job when requested by cell-specific mini agents in an agent kernel. Link disassembler traces back the links of an existing service to modify one of the configuration elements to add or remove a new service.

Agent Kernel

It is the main functional engine of an intelligent agent to support automated intelligent configuration. The essential parts forming the kernel are:

- *Puller*: It retrieves configuration information (for example Cisco IOS file) from the target device to read the current configuration state in that device. This pulled information can be a simple ASCII file or HTML embedded. Whatever the format is, the kernel, with its parser, extracts appropriate values and configuration elements.
- *Cell-Specific Mini Agents*: These are specialized in managing configuration elements that are cell-specific. A particular mini agent only knows how to configure elements with similar bodies. The agent kernel has several mini agents that work cell-wise and the results from these when combined with the help of the element linker (which of course derives knowledge from the knowledge base) produce the desired action needed to create the required service.
- *Pusher*: This sends the ultimate combined configuration results to the network device through device connector.

Interface to Dispatcher

This is a communication port of the agent for the Central Dispatcher to send service requests. This is usually achieved by having the server socket program waiting for service request on a specific port. The interface also serves as an indicator of the service type the agent can provide, since agents are service-specific.

Interface to Report Collector

This is yet another communication port that is actually a client socket program. This interface basically sends all event reports prepared by the reporter of the agent kernel to report collector located in the central repository system.

Device Connector

As the configuration sent by the pusher needs to be sent in different ways to the devices, Device Connector has several plug-ins like TELNET, HTTP, FTP, SNMP etc. to facilitate such service in a heterogeneous environment where devices have various access mechanisms.

7.5.3 Example of Intelligent Provisioning

In this section we will show some VPN service-specific implementation examples of the intelligent provisioning agents.

The first example in Figure 7.10 shows a C++ code fragment of a cell- specific mini agent that intelligently pulls out the available traffic-classifier number (usually called `access-list` number) from a network device. Every time there is new a source-destination pair for VPN traffic a new `'access-list'` number is needed that uniquely identifies the traffic. If the valid starting number is 100 for a particular device and there is no traffic classifier present in that device the mini agent returns 100; otherwise, it returns the next unused number. It does so by searching for configuration elements having body `access-list` in the configuration space represented as `ConfigFile` in the code. The string `ConfigFile` contains the whole Cisco IOS configuration file as shown in Figure 7.3. This helps us to avoid storing device-specific data in the policy repository. In this

```
/* searching for all ACL Numbers */
 regx re("access-list (\\d+) .*");
 regx::iterator acl = re.find_n(ConfigFile);
 if (acl == re.end()) {
    int StartACL = 100;
    return StartACL;
  }
 /* Vector V would contain all the ACL Numbers */
 vector <int> V;
 int c=0;
 for (regx::iterator i = re.begin(); i != re.end(); i++) {
  if (1&c) V.push_back(atoi(re[c].c_str()));
  c=c+1;
 }
 /* Searching for the Maximum ACL Number. *it is the highest
   number, so return (*it +1) to be used as the next ACL. */
 vector<int>::const_iterator it = max_element(V.begin(), V.end());
 return (*it+1);
```

Figure 7.10: Partial Code for Smart Provisioning: Example 1

example, we do not need to store `access-list` numbers because this is device-specific and having only source-destination pairs as VPN traffic in the policy repository should be enough for other device-specific agents to translate that into appropriate device-specific configurations by deploying similar agents.

The second example in Figure 7.11 shows a C++ code fragment of another cell-specific mini agent that smartly configures an IPSec VPN service in a network device. Assume that we currently have a network setup as shown in Figure 7.3(a) and a VPN administrator wishes to create a tunnel between device 130.92.70.102 and 129.194.90.20 for a source-destination pair 172.17.0.103 and 172.18.0.100 using security parameters AH authentication and ESP encryption. Figure 7.4(b) shows the expanded network setup. An automated provisioning agent that ignores configuration dependencies among different configuration elements would probably generate the following configuration for the network device 130.92.70.102.

```
crypto map genbern 161 ipsec-isakmp
set peer 129.194.90.20
set transform-set ah-md5-hmacANDesp-des
match address 141
access-list  141 permit ip host 172.17.0.103 host 172.18.0.100
```

However, the intelligent agent enters the target device, reads the existing configuration, and analyzes it using the knowledge base and detects that a tunnel already exists with the same security parameters between the same peer routers although for different LANs. In doing so, it actually searches configuration elements in a cell having body tag `crypto map` whose attributes are the same as the newly requested one. As one such already exists, it discovers that adding the above configuration lines would be unnecessary. The mini agent, therefore, pulls the existing traffic classifier number (i.e.`access-list` number 140), and simply adds the line `access-list 140 permit ip host 172.17.0.103 host 172.18.0.100` to the current configuration.

7.6 Intelligent Device Auditing

When configuration faults occur on the network device, it is imperative that problems be resolved quickly to decrease the negative impact on user productivity. Network managers must respond quickly and have procedures in place to reestablish lost services and maintain beneficial service levels. Several reasons justify why such problems might occur. Services configured by human administrators may lead to mis-configuration or cause conflict with the existing services activated via the automated system. Also, if network devices crash due to power failure or some other reasons, valuable configuration information thrown into the devices might well be lost partially or fully. One cannot rule out the possibility of an inside or outside intruder changing some configuration settings. Intelligent audit management handles such problems to recover from any mis-configuration caused by any process other than the automated system. Audit management ensures that the configuration state in the policy repository is equivalent to actual device state as illustrated in Figure 7.12. Similar to provisioning agents intelligence is built in the audit agents that gather and sort the configuration data to quickly identify the reason and location of faults in a network device and automatically fix the problem when it occurs. Depending on SLAs with customers, audit agents can be periodically sent to the network devices to check the consistency of network device configuration.

7.6.1 Audit Management Architecture

With the objective that the configuration state in the repository must be equivalent to the actual device state, we have proposed an intelligent audit management

```
/* setting the values of source, dest address that would be
    tunneled. 'peeridaddr' is the remote crypto endpoint for the
    tunnel and the rest are ipsec tunnel authentication and
    encryption parameters. */
   string source = "172.17.0.103";
   string dest= "172.18.0.100";
   string peeripaddr ="129.194.90.20";
   string AH_authentication="ah-md5-hmac";
   string ESP_encryption="esp-des";

/* setting the regular expression search string */

   string ipsectransformset=AH_authentication;
   ipsectransformset +="AND";
   ipsectransformset +=ESP_encryption;
   string searchstr;
    searchstr  = "crypto map cati-tunnel \\d+ ipsec-isakmp.*\n";
    searchstr += ".*set peer "+peeripaddr+".*\n";
    searchstr += ".*set transform-set "+ipsectransformset+".*\n";
    searchstr += ".*match address (\\d+).*";
 /* searching in configuration space for a similar tunnel. */
    ConfigFile contains complete configuration information */
     regx re(searchstr);
     regx::iterator ipsec = re.find(ConfigFile);
     if (ipsec == re.end()) {

     No match found. Code to proceed as usual.
           ⋮
      }
/* if match found re(1) contains traffic access list number.
  Just generate the ACL config line and that's enough for
  tunnel configuration */
    string configline;
    configline = "access-list ";
    configline += re[1];
    configline += " permit ip host ";
    configline += source;
    configline += " host ";
    configline += dest;
 /* send this configuration line down to the router */
  ConfigRouter(configline);
```

Figure 7.11: Avoiding Device-Specific Data in Repository: Example 2

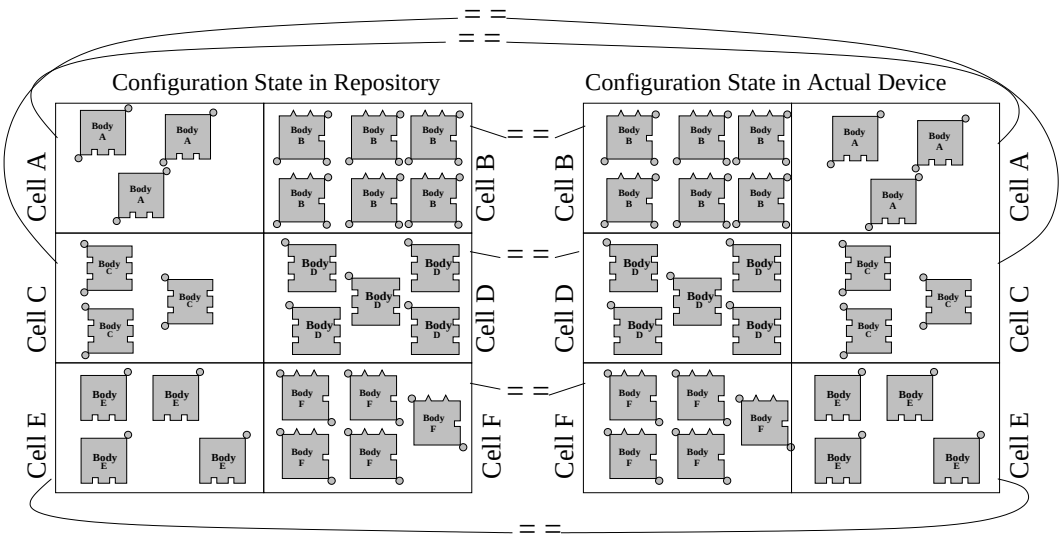


Figure 7.12: Objective of Audit Management for Device Configuration: Configuration State in the Repository must be Equivalent to Actual Device State

architecture (Figure 7.13). The audit management process is done periodically for each device under the control of an automated management system and basically goes through the following cycle of actions:

Audit Request Generation

The central repository system maintains records of all the service requests provisioned by the automated system. For a specific device, the Audit Manager (i.e. audit agent) can extract all service requests and generate the expected configuration state for that particular target device. This is done by the device-specific interpreter (DSI) which utilizes the same knowledge base as used in a provisioning agent. This step, therefore, is quite similar to original service provisioning of a new service request. However, unlike the latter case when an intelligent provisioning agent actually configures a service, the DSI simply prepares the expected configuration for provisioned services and passes that to the configuration violation detector (CVD) to identify any unwanted modification in the physical device.

Problem Finding

At this stage the audit management agent first pulls the current configuration state from the device and the CVD compares that with the expected configuration state received from DSI. If any configuration violation is detected (i.e. expected configuration and actual device configuration do not match) the violation detector signals that to the configuration violation locator (CVL). Since we map the whole configuration space by dividing them into clusters of cells, the violation locator locates cells where one or more configuration elements may have been changed.

Problem Fixing

For a specific service, changes or mis-configuration in one configuration element may easily affect other attached elements that constitute the service. Therefore, to audit a service all appropriate affected cells are located to be fixed by the re-provisioning process. The problem fixing stage basically sets the configuration state in the device to the expected state by making specific corrections in corrupted cells (of the configuration map) located by the CVL. Violation corrector determines what needs to be changed in the corrupted cells. Usually some configuration elements might need to be added, modified or deleted. For these actions to take place, cell-specific mini agents re-provision the network device to bring the configuration state of the device back to the desired state. Although not shown explicitly in Figure 7.13, both in problem finding and fixing stages the knowledge base and other components in the intelligent agent architecture needs to be invoked by the re-provisioning agents to take appropriate actions.

7.7 Implementation Architecture

In this section we will present our implementation architecture (Figure 7.14) which actually depicts a deployment scenario. The implementation generally tries to fulfill the objectives set in the proposed network management architecture of section 7.3. While there are several components and functionalities that actually build the complete system, the three major players are the Central Repository, Central Dispatcher and the agents that reside in clusters of workstations. The central repository is hosted by a web server in order to facilitate distributed management access via web. The goal of this section is not to describe the details of

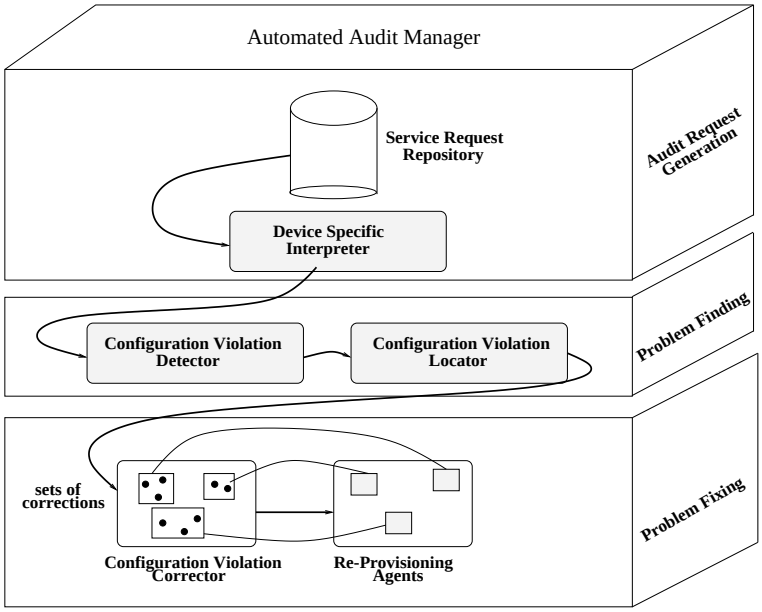


Figure 7.13: Automated Audit Management Agent Architecture

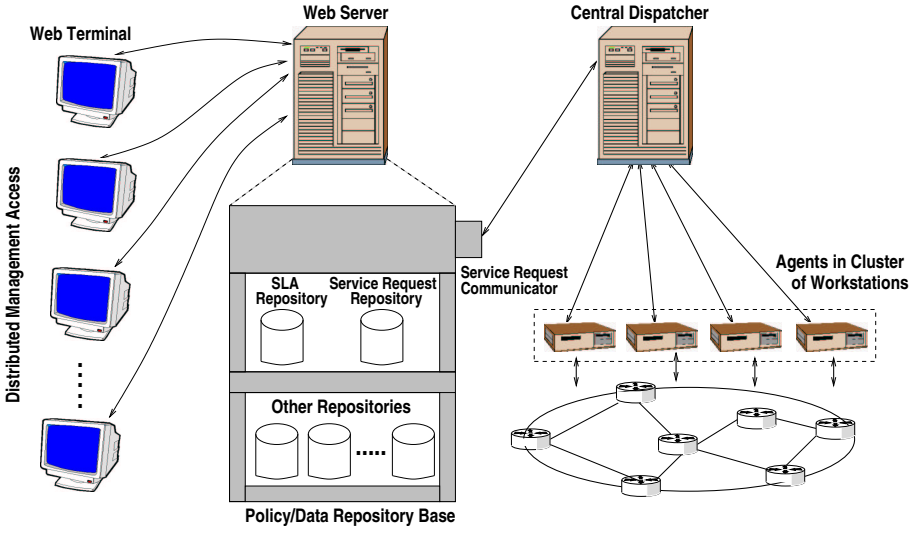


Figure 7.14: Prototype Implementation Architecture

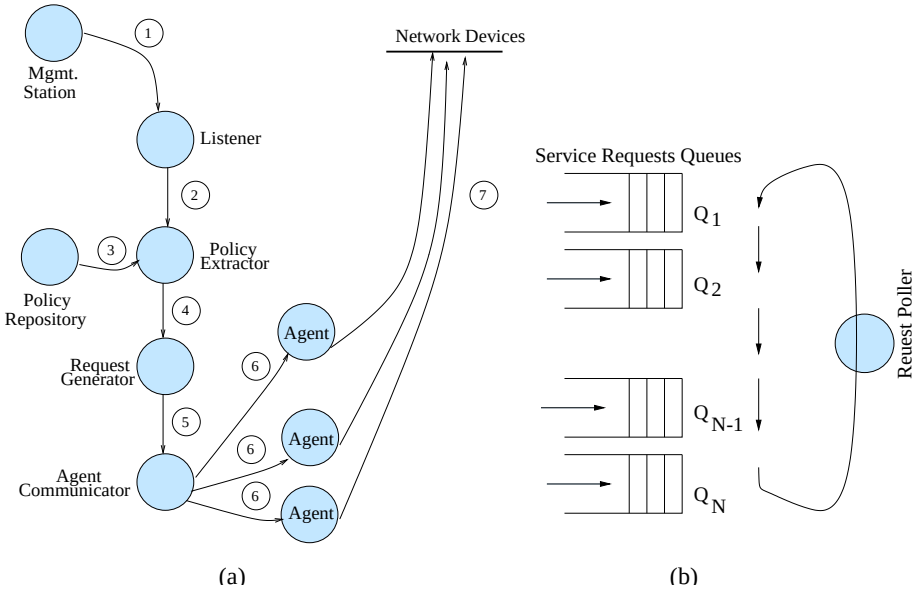


Figure 7.15: (a) Request Processing Life Cycle (b) Periodic Polling of Service Requests in Cyclic Order

the implementation as that is not within the scope of this thesis, but to shade lights on the most important parts that constitute the system and the communications involved to make dynamic provisioning in large scale environment a scalable solution.

7.7.1 Central Repository

The central repository system not only maintains customer SLAs and service request repositories, but also similar others (as discussed in chapter 3) to facilitate automated provisioning and policy based networking of various types of existing and emerging IP services. As new services emerge, the types and contents of the repositories may be enhanced. Examples and formats of such repositories can be found in earlier chapters and also in [KBG00], [KB00b], [BGK01].

7.7.2 Central Dispatcher

Figure 7.16(a) shows the structure of the client-server process in the central dispatcher. There are several client-server processes in the central dispatcher and

the communication between any client and a server process is achieved via TCP sockets on UNIX platforms. The Central dispatcher's Master Main Loop (MML) has a *request listener* to process very high priority service requests for immediate processing and a *request poller* to periodically poll and serve pending requests that are not very time sensitive. Audit requests of previously configured services are also polled by the request poller. Figure 7.15(a) shows the processing life cycle of a service request. The request poller serves a set of N service request queues $Q_1, Q_2, ..., Q_N$ in a fixed cyclic order: $Q_1, Q_2, ..., Q_N, Q_1, Q_2, ..$ as shown in Figure 7.15(b). The queues are ordered based on priority of service requests and served under the exhaustive service policy, i.e. the poller serves each queue until the queue becomes empty. Regardless of how the MML receives requests, it starts the service activation (or re-configuration in the case of audit) process for a request by invoking the following:

- *Policy Extractor*: What the Central Dispatcher receives for a new service or audit request is a request ID. The Policy Extractor invokes the Central Repository System to extract raw data required to create a new or audit an existing service and then passes that to the Request Generator.
- *Request Generator*: The Request Generator formats the raw data of a service request in a way that is easily extractable by the parser of the intelligent agent.
- *Agent Communicator*: The Agent communicator sends the formatted request to the appropriate agent. The decision about which agent to communicate is based on simple load balancing policy.

7.7.3 Distributed Agent Processes

Figure 7.16(b) shows several intelligent agent processes running on clusters of workstations. One of the Agent listener daemon processes is a dedicated server that is the entry point from agent communicator client processes in the central dispatcher.

When the central dispatcher is invoked because of the arrival of a service request,the agent communicator tries to make a TCP connection with the agent listener. The Agent listener spawns (fork and exec) a new agent process to provision a device.

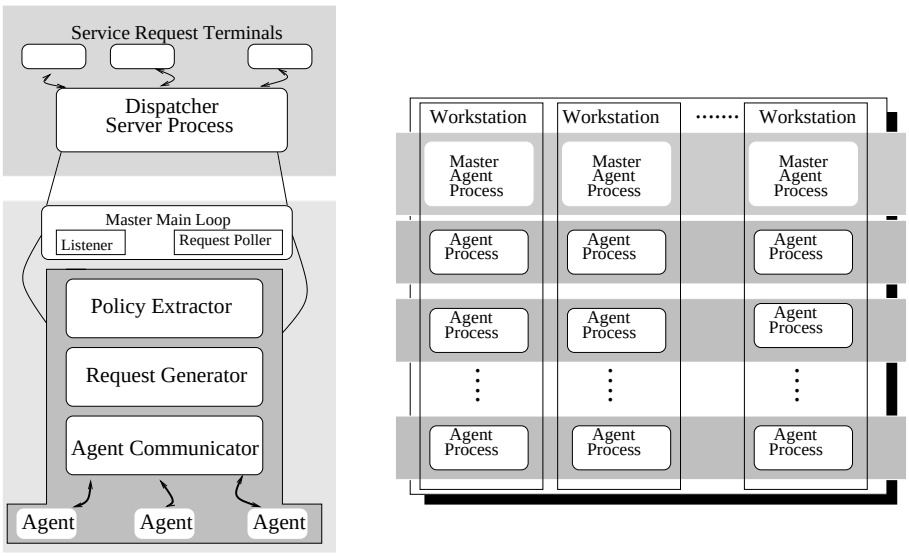


Figure 7.16: (a) Distributed Process Architecture of the Central Dispatcher (b) Distributed Process Architecture for Multi Agent Spawning

7.8 Conclusion

In recent years, intelligent mobile agents have proven to be indispensable tools for providing network management assistance. They will become even more indispensable as networks continue to expand and companies continue to minimize their personnel requirements. For ISPs and large enterprises, the capability to effectively and remotely provision a large number of network devices from a centralized or distributed locations becomes even more important. We have proposed a new distributed architecture where highly mobile intelligent agents can take the responsibilities of not only provisioning but also configuration audit management in a timely fashion. Although recent research have focused on using mobile agents for network monitoring or simple push-based device configuration in a distributed architecture, their ability have not been exploited in dynamic IP service provisioning. In this chapter, we have taken a new approach to configuration modeling that is device neutral and based on which any existing or emerging new IP services can be presented in an abstract form by encapsulating service semantics, including service-specific data. We have developed new intelligent provisioning and audit agent architectures that use the knowledge built upon configuration dependency modeling. Although we have presented a prototype implementation,

performance evaluation in terms of provisioning response time and effectiveness of load balancing remain to be future research issues. However, results in [LR92], [HBCM99] clearly approves our approach. We have not addressed the issue of secured configuration delivery by the agents. This can be easily achieved by establishing a secured management tunnel between the agent and the network device using IPSec or SSH protocols. However, if the devices do not support such protocols, a scalable new mechanism for secured configuration delivery will be an interesting research issue. Performance evaluation in terms of provisioning response time could be useful. Also, further investigation of advanced policy management [SL99] [Wie94] [KKR95] in large scale environment would be interesting.

Chapter 8

Summary and Conclusion

In this chapter we provide a summary of the thesis, and identify the contributions we have made. We also briefly discuss the limitations of our work and future work that others may pursue to enhance the works presented in this thesis.

8.1 Summary

It is estimated that almost forty percent of the total VPN budget in an organization is spent for deploying and management of VPN and network analysts advocate outsourcing the VPN services to the ISPs [VPN]. In an increasingly growing complex network the success of both ISPs and enterprises depend on how the IP services are created and managed. This thesis identifies the challenges that ISPs or corporate network managers face in diverse and rapidly growing large networks, and proposes solutions toward automation and management of emerging network services. Instead of manually configuring each network device via Command Line Interface (CLI) the policy-based QoS-VPN Management solutions proposed in this thesis feature a streamlined end-to-end activation process through interfaces that enables both network administrators and end users to apply settings to multiple devices with ease. Thus, programmability of network components to dynamically create services has been one of the goals of this thesis.

To address these challenges in this thesis, we proposed a policy based Service Broker (SB) architecture and demonstrated an implementation in chapter 3 that can automate QoS-VPN activation initiated by both network administrators and corporate customers in large network installations. As we developed better ideas

in terms QoS provisioning and signaling between Diffserv domains, advanced Service Brokers were also presented in this thesis. In chapter 3 we have described the implementation of a Service Broker that allows registered users having SLA with their ISPs to establish and terminate QoS enabled VPN tunnels dynamically. Unlike in traditional policy managed systems where only network administrators can activate services using a policy console, users of our system can also login from a web based front-end, verify themselves and initiate a VPN based on their predefined SLA and needs. This obviates the need of invoking help from a system administrator or an ISP and at any time they can disconnect the VPN service or check their current bills. As security and QoS requirements of one user might vary from others under various circumstances depending on the needs, the Service Broker interface presents various IPSec [KA98] tunneling and QoS policies as user selectable options (Figure 3.6) to allow a customer to choose the one that suits him best. To appreciate service differentiation we have introduced a new differential tunnel pricing mechanism by which pricing of a QoS-VPN tunnel is computed based on its network resource reservation and the load during the time tunnel is active. One useful feature of the architecture is that it allows to add new policy templates in the system as new device types are added to the network which gives the network managers the ability to operate in multi-vendor environment.

However, the implementation of Service Broker in chapter 3 supports a single ISP domain only. In practice, a Virtual Leased Line (VLL) may actually span over several domains. To overcome this limitation, we have extended the Service Broker in chapter 6 that uses a simple signaling mechanism to communicate with other cooperative brokers to enable customers to dynamically create VLLs over multiple DiffServ domains. We have presented a simple approach to make advance reservations in the absence of senders or receivers in a multi-domain scenario. Rather than using RSVP and COPS in inter-domain signaling to reserve capacity across domains, we used a novel method to identify domains, and hence Service Brokers that are responsible for maintaining them. Corresponding brokers are contacted along the path of a VLL from ingress to egress for a yes/no admission control decision. If every broker responds positively, a VLL is set up.

Yet another limitation of the Policy Server in chapter 3 is that it allows users only to specify a single quantitative value (i.e. 1 Mbps or 2 Mbps etc.) and based on this specification the edge routers establish VPN connections dynamically. However, such a simple QoS policy may not be suitable and it is apprehended that some users will be unable or unwilling to predict the load between VPN endpoints [DGG+99]. Also, from the provider's point of view, guaranteeing exact

quantitative service might be a difficult job at the beginning of VPN-DiffServ deployment [BBC+99]. In chapter 4, we have proposed a novel range-based SLA that allows customers to specify their requirements as a range of quantitative services. For example, users who want to establish VPN connections between two stub networks and are not sure whether 0.5 Mbps, 0.6 Mbps or 1 Mbps are needed, and only know the lower and upper bounds of their requirements approximately, can specify a range 0.5- 1 Mbps when they outsource their services to the ISPs. An ISP can offer such multiple options via a web site (Figure 6.6) to help customers to select any suitable option to activate services dynamically on the fly. However, to support such services, the underlying resource allocation mechanism of the Service Broker of chapter 3 was further enhanced in chapter 4 to logically partition the capacity at the edges to various service classes (or groups) of VPNs and manage them efficiently to allow resource sharing among the groups in a dynamic and fair manner.

One obvious advantage of the enhanced policy server supporting range-based SLA is the pricing gain. The price that customers have to pay is higher than one pays for the lower-bound capacity but lower than what is normally needed to be paid for upper-bound capacity. During low-load it is possible that users might enjoy the upper-bound rate without paying anything extra. Such pricing might be attractive to users and ISPs can take advantage of this to attract more customers.

We restricted chapter 4 to edge provisioning only considering the fact that most of the complexities lie at the boundaries of the network and is the main driving force for overall provisioning. However, the ISPs must provision the interior nodes in the network to meet the assurance offered at the boundaries of the network. In chapter 5, we proposed a core provisioning architecture and further enhancements of the Service Broker that works with the proposed edge resource allocation policies addressed earlier in chapter 4. We show how we can exploit range-based SLAs to simplify core provisioning, make multiplexing gain and guarantee at least lower bounds of bandwidth range even under heavy VPN demand conditions. Simulation results support our claims and analysis. One of the main motivation behind a novel core provisioning method was scalability. Dynamic and frequent configurations of an interior device driven by edge bandwidth modifications is not desired as this will lead to scalability problems and also defeats the purpose of the DiffServ architecture which suggests to drive all the complexities towards edges. We address this scalability issue by proposing a virtual core provisioning that only requires a capacity inventory of interior devices to be updated based on VPN

connection acceptance, termination or modification at the edges. In our virtual core provisioning architecture, an edge router selects an explicit route and signals the path through the network, as in a traditional application of MPLS. Router interfaces along these routes are pre-configured to serve certain amount of quantitative VPN traffic. A new VPN connection is subject to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers actual configuration of edge device, but only resource state updates of core routers interfaces in the Service Broker database.

Many of the dynamically created services at least partially depend on the current configuration states of the target devices. Using a simple push-based model to configure network services while ignoring dependencies among configurations elements may easily lead to configuration inconsistencies resulting in failure or inefficiencies. For example, in chapter 3, the Service Broker configuration process deployed such a simple push-based approach.In chapter 7, we have taken a new approach to configuration modeling that is device neutral and based on which any existing or emerging new IP services can be presented in abstract form by encapsulating service semantics, including service-specific data. We have proposed a new distributed architecture where highly mobile intelligent agents built upon the knowledge derived from configuration dependency modeling can take the responsibilities for not only provisioning but also for configuration audit management in a timely fashion.

8.2 Related Works

While we have introduced new ideas in terms of automated service activation, resource provisioning, configuration modeling, intelligent provisioning agents, and inter domain signaling between Service Brokers, similar works have been reported by numerous researchers. We describe them and compare with our works in the following:

- **Bandwidth Brokers and Automated Provisioning**: To take advantage of the new service differentiation technology, Bandwidth Brokers that can dynamically create VLL on demand have been proposed in [NJZ99], [Sch98] and refined in [TWOZ99], [Tea99]. The policy-based Service Broker architecture is derived from these earlier works. However, we introduced the idea of QoS and IPSec policy templates to translate user service requirements and high level policies into low level device specific configurations.

This allows to add new policy templates in the system as new device types are added to the network and gives the networks managers the ability to operate in multi-vendor environment. New Bandwidth Broker models based have been proposed in [MlCPT00], [PSV00], [ZDGH00], [FdCPPdR01], and several implementations have been reported in [KBG00], [KB00a], [BGK01], [TOT+99], [Tea99], [QBO00], [KB00b], [Sta02] but they mostly do not work in multi-vendor and multi-ISP environment. The enhanced implementation [KB01b] of the Service Broker presented in chapter 6 allows customers to dynamically create Virtual Leased Lines (VLLs) over multiple DiffServ domains. While most of the implementations rely on RSVP for both intra-domain and inter-domain signaling between brokers, we use a novel method to identify DiffServ domains, and hence the Service Brokers that are responsible for maintaining them. Identified brokers in a chain are signaled and positive responses from all the brokers leads to end-to-end VLL setup. Also, rather than using COPS for policy provisioning in network devices, we use agents such as configuration daemons to do the same job in a push-based fashion.

- **QoS Provisioning**: To support range-based SLA [KB02b] we proposed edge provisioning methods that can logically partition the capacity at the edges to various classes or groups of VPN connections and manage them efficiently to allow resource sharing among the groups in a dynamic and fair manner. Here, each group is identified from what it offers. For example, one group could represent the range 0.5- 1 Mbps, another 1-2 Mbps. One advantage of range-based SLA is that customers do not have to specify exact bandwidth between VPN endpoints, they only need to specify ranges. Hose-based models [DGG+99], [KRSY01], [RCK02] for edge resource allocation of VPN connections also do not require customers to specify exact resource requirements, only per VPN endpoint specification is considered sufficient. In hose models, the ingress bandwidth for an endpoint specifies the incoming traffic from all the other VPN endpoints into the endpoint, while the egress bandwidth is the amount of traffic the endpoint can send to the other VPN endpoints. While this has the ease of specification and flexibility, in such models service providers allocate resources to aggregated VPN tunnels, and the amount of bandwidth a corporate user may receive out of aggregated allocation is probabilistic. In contrast, using range-based SLA customers are guaranteed at least the lower-bound capacity of the range they specify.

  In our proposed virtual core provisioning, a new VPN connection is subject

to admission control at the edge as well as at the hops that the connection will traverse. An acceptance triggers the actual configuration of an edge device, but only resource state updates of core router interfaces in the Service Broker database. Therefore, unlike the traditional IntServ approach, which has the fundamental scalability limitations because of the responsibility to manage each traffic flow individually on each of its traversed routers, our virtual core provisioning approach does not suffer from the same problem since capacity reservation states are actually stored in a Service Broker based inventory and not in the core routers. Other works that propose guaranteed services without per flow provisioning at core are: [SZ99], [SSZ98], [CK00], [ZDGH00], [BV01], [LC01], [CB01]. However, all of them consider short-lived flows while VPN connections in our case are usually rate-controlled long-lived flows that are often provisioned for larger time-scale.

Some notable works [FBP+01], [JD02], [QK01], [KS99] on bandwidth allocation focus on achieving statistical multiplexing gain for bursty traffic sources. However, statistical characterization of traffic sources is not often reliable [BBLO00], [RRR98] and VPN connections are statistically independent and smoothed by deterministic regulators at the connections input to the network edge based on provisioning policies. In contrast to the approaches in [FBP+01], [JD02], [QK01], [KS99], we achieve multiplexing gain and revenue by exploiting range-based SLA where guaranteeing the lower-bound bandwidth of the range is considered sufficient.

- **Configuration Modeling and Distributed Management**:Automated IP service creation in network devices relies on the fundamental assumption of having an accurate and consistent abstraction of the underlying network, particularly a view of essential parts of configuration information stored in those devices spread across different device-specific repositories in different formats. Relationships between different configuration elements are implicit with repositories containing replicated and interdependent configuration information. Many of the dynamically created services at least partially depend on the current configuration states of the target devices. Using a simple push-based model to configure network services while ignoring dependencies among configuration elements may easily lead to configuration inconsistencies resulting in failure or inefficiencies. In this thesis, we have presented [KB02a] a generalized view of configuration information in a network device and modeled it as clusters of configuration elements.

A service is constituted from configuration elements spread across the clusters, and thus elements are linked to each other for a specific service. Using intelligent agents that derive knowledge from the proposed configuration modeling, configuration inconsistencies can be avoided.

The Commonly used SNMP protocol does not have the capability to capture dependency among configuration elements. Physical resources are represented by managed objects within these protocols. All managed objects in the SNMP environment are arranged in a hierarchical or tree structure. The leaf objects of the tree are the actual managed objects, each of which represent some resource, activity, or related information that is to be managed. The tree structure itself defines a grouping of objects into logically related sets called a Management Information Base (MIB). Associated with each type of object in a MIB is an identifier of the ASN.1 type OBJECT IDENTIFIER. The object identifier is a unique identifier for a particular object type and serves to name the object. COPS, which has Policy Information Bases (PIBs) analogous to MIBs, manages policy rules for network devices but also does not have built in capability to capture dependency among configuration elements and avoid configuration inconsistencies. Although the protocols can be modified or tools can be built on top of these protocols to achieve this ability, no reported works exist. This often neglected but important issue of configuration modeling was however addressed in [YKF00] with limited use in useful IP service creation.

Since existing network management architectures ([MFZH99]) deploying SNMP [Sta98] or CMIP [WBLH90] as the management protocols do not really fit well to suit the needs of automated provisioning in a large scale environment due to some well known scalability limitations of the deployed protocols, for the sake of decentralization of management tasks we proposed [KB02a] intelligent provisioning and audit agents that are built upon the knowledge derived from configuration modeling. Many architectures for network management with mobile and intelligent agents have been proposed in the literature [Sch97], [PT99], [KLS97], [BGP98], [BGP97], [GY98], [MEB+95], [CCLM99], [BPW98], [EDB99] to address automation tasks by dynamically delegating management functions to the agents. However, the vast majority of these agents have been mainly used in traffic analysis, fault management, network monitoring and performance management, and not intelligent service provisioning and configuration audit.

- **Configuration Audit Management**: There are several reasons why configuration problems may occur in network devices. Services configured by human administrators may lead to mis-configuration or cause conflict with the existing services activated via the automated system. Also, if network devices crash due to power failure or some other reasons, valuable configuration information stored into the devices might well be lost partially or fully. One cannot rule out the possibility of an inside or outside intruder changing some configuration settings. Intelligent audit management proposed in our work handles such problems to recover from any mis-configuration caused by any process other than the automated system. Audit management ensures that the configuration state in the policy repository is equivalent to actual device state.

  To the best of our knowledge very little works have been reported in this area. Some closely related works can be found on Configuration and Change Management systems [CDF+95], [Fos97] [ML88a] [BABR96] [PR99] [ML88b] that help customers to formalize the manner in which changes are evaluated, approved and moved into production, thus dramatically reducing disruptions and improving network service availability. However, these change management systems are considered to be preventive methods and unlike the proposed audit management system do not deal with cases when there are unauthorized network device configurations changes.

  Since the role of fault management is to detect, diagnose and correct the possible faults during network operations, such a system could be easily adapted to perform configuration audit tasks. However, most of the traditional fault management systems [CCLM99] [KLMB96], [EDB99], [LMB97], [OMK+97] consider the term *fault* the same as *failure* which may mean component malfunctions, network interface not responding, broken links etc. In case of mostly physical network problems, these fault management systems detect, isolate, and repair problems in the network by using error logs and tracing errors through log reports and interface data collected via SNMP.

8.3 Limitations and Future Works

Despite the contributions we have made by proposing and implementing Service Brokers, new resource allocation and dynamic provisioning and signaling mechanisms, the works presented in this thesis have some limitations and further works

may be useful. We describe those limitations and possible future works in the following areas:

- *Generic Service Platform*: Although our architecture supports creation of new services in new devices using policy templates, the service broker prototype in this thesis shows examples of QoS and VPN provisioning only. Also, despite the ability to add new services by adding templates, data modeling and logic of system flows in the service broker system still require enhancement to support such a facility. Network administrators may find it difficult to modify existing system to support unknown future services. Development of a toolkit that would help administrators to define new services without changing the underlying system engine would be very useful.

- *Centralized SB-based QoS Routing*: The centralized SB in its role as a global network manager maintains information about all the established real-time VPN tunnels and the network topology, and can thus select an appropriate route for each real-time connection request. If a pinned path or pre-selected alternate routes fail to reserve requested resources for a VPN connection, QoS routing can then be used efficiently to select an appropriate one that meets the requirements. Since the objective of any routing algorithm is to find a qualified path with minimal operational overheads, a centralized SB-based QoS routing might be very effective. This is an issue we have not addressed and can be a future research topic.

  Also, the centralized Service Broker is the right system to setup explicit path that is determined by using QoS routing. Some research papers [SWW01], [AKK+00] already suggest the use of MPLS to setup such paths, and similar methods can be used in the Service Broker for QoS enabled VPNs. Therefore, integration of traffic engineering capabilities in the SB automated system is an interesting topic.

- *Scalability of BB in Multi DiffServ Domains*: Further simulation work might be useful to examine the scalability and performance of our broker-to-broker signaling approach and is a topic of future research. Also, comparison of the proposed method with other interdomain signaling approaches [Gün01] might be interesting.

- *Secured Configuration Delivery* : We have not addressed the issue of secured delivery of configuration by the agents. This can be easily achieved by establishing a secured management tunnel between the agent and the network

device using IPSec or SSH protocols. However, if the devices do not support such protocols, a scalable new mechanism for secured configuration delivery will be an interesting research issue.

- *Performance Evaluation of Distributed Agents*: The prototype implementation of automated service provisioning system in chapter 7 was developed to address scalability issue of such systems to perform well in large scale networks. Service Brokers deploying only a handful of agents can easily become a bottleneck when there are many requests to serve. In chapter 7, the machines hosting a policy repository or a central dispatcher do not perform network device provisioning, and actions taken by them do not require much computing power. Provisioning tasks are performed by distributed agents running on separate machines. We did not carry out simulation experiments because results in [LR92], [HBCM99] clearly approves our approach. In a heterogenous environment different types of network devices take different amount of time to be provisioned. A machine hosting many provisioning or audit agents and serving many requests may become a bottleneck while other agents remain lightly loaded. Therefore, performance evaluation in terms of provisioning and audit service response time and effectiveness of load balancing by central dispatcher in a large scale heterogeneous environment require further investigation. For realistic simulation the use of emulated active network platform [Bau02] may be a good approach.

- *Integrated advanced policy Management*: Further investigation of advanced policy management [SL99], [Wie94], [KKR95], [Slo94], [KLMB96], [TFP02] in a large scale environment managing QoS enabled VPNs, MPLS, and traffic engineering services from a single integrated platform would be interesting.

Bibliography

[AKK+00] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter. RATES: A Server for MPLS Traffic Engineering. *IEEE Network Magazine*, 14(2):34 – 41, March/April 2000.

[Ash99] Gerald R. Ash. Routing Guidelines for Efficient Routing Methods. Internet Draft `draft-ash-itu-sg2-routing-guidelines-00.txt`, October 1999.

[BABR96] Luc Bellissard, Slim Ben Atallah, Fabienne Boyer, and Michel Riveill. Distributed Application Configuration. In *Proc. of 16th IEEE International Conference on Distributed Computing Systems*, pages 579–585, 1996.

[Bau02] Florian Baumgartner. *Quality of Service Support by Active Networks*. PhD thesis, IAM, University of Berne, February 2002.

[BBC+98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weis. An Architecture for Differentiated Services, December 1998. RFC 2475.

[BBC+99] Y. Bernet, J. Binder, M. Carlson, B. E. Carpenter, S. Keshav, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A Framework for Differentiated Services. Internet Draft `draft-ietf-diffserv-framework-02.txt`, February 1999.

[BBCF01] D. Black, S. Brim, B. Carpenter, and F. Le Faucheur. Per Hop Behavior Identification Codes, June 2001. RFC 3140.

[BBLO00] R. R. Boorstyn, A. Burchard, J. Liebeherr, and C. Oottamakorn. Statistical Service Assurances for Traffic Scheduling Algorithms.

IEEE Journal on Selected Areas in Communications, 18(12), December 2000.

[BCD+00] J. Boyle, R. Cohen, D. Durham, S. Herzog, R. Rajan, and A. Sastry. The Common Open Policy Service (COPS) Protocol, January 2000. RFC 2748.

[BCS00] Paolo Bellavista, Antonio Corradi, and Cesare Stefanelli. An Integrated Management Environment for Network Resources and services. *IEEE Journal on Selected Areas in Communications*, 18(5):686 – 701, May 2000.

[BGK01] Torsten Braun, M. Günter, and Ibrahim Khalil. Management of Quality of Service Enabled VPNs. *IEEE Communications Magazine*, 39(5):90–98, May 2001.

[BGP97] M. Baldi, S. Gai, and G. P. Picco. Exploiting Code Mobility in Decentralized and Flexible Network Management. *First Int'l Workshop on Mobile Agents Mobile Agents'97,Berlin, Germany*, pages 13 – 26, April 1997.

[BGP98] M. Baldi, S. Gai, and G. P. Picco. Enabling a Mobile Network Manager (MNM) Through Mobile Agents. *2nd International Workshop on Mobile Agents,Stuttgart, Germany*, pages 249 – 260, September 1998.

[BPW98] Andrzej Bieszczad, Bernard Pagurek, and Tony White. Mobile Agents for Network Management. *IEEE Communications Surveys*, 1(1):2 – 9, September 1998.

[BV01] Sudeept Bhatnagar and Brett J. Vickers. Providing Quality of Service Guarantees Using Only Edge Routers. In *Proceedings of IEEE Globecom*, San Antonio, USA, November 2001.

[BW02] U. Blumenthal and B. Wijnen. User-based security model (USM) for version 3 of the simple network management protocol (SNMPv3), December 2002. RFC 3414.

[CB01] B. Choi and R. Bettati. Endpoint Admission Control: Network Based Approach. In *IEEE Proceedings of ICDCS*, pages 227–235, April 2001.

[CCLM99] M. Cheikhrouhou, P. Conti, J. Labetoulle, and K. Marcus. Intelligent Agents for Network Management: Fault Detection Experiment. *Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99), Boston, USA*, pages 63–83, May 1999.

[CDF+95] S. Crane, N. Dulay, H. Fossa, J. Kramer, J. Magee, M. Sloman, and K. Twidle. Configuration Management for Distributed Software Systems. *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (ISINM 95),Santa Barbara, USA*, pages 29 – 42, May 1995.

[CK00] Coskun Cetinkaya and Edward W. Knightly. Egress Admission Control. *INFOCOM'2000*, March 26-30 2000.

[Cla99] David D. Clark. A Model for Cost Allocation and Pricing in the Internet. *Workshop on Internet Service Quality Economics, MIT*, Dec 2-3 1999.

[CN98] S. Chen and K. Nahrstedt. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. *IEEE Network Magazine*, November/December 1998.

[CNRS98] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet, August 1998. RFC 2386.

[Cru95] R. L. Cruz. Quality of Service Guarantees in Virtual Circuit Switched Networks. *IEEE Journal on Selected Areas in Communications*, 13(6):1048 – 1056, August 1995.

[DGG+99] N.G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K.K. Ramakrishnan, , and Jacobus E. Van der Merwe. A Flexible Model for Resource Management in Virtual Private Networks. *SIGCOMM'99 Conference*, August 1999.

[EDB99] M. El-Darieby and A. Bieszczad. Intelligent Mobile Agents: Towards Network Fault Management Automation. *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, pages 611–622, 1999.

[FBP+01] S. Ben Fredj, T. Bonald, A. Proutierea, G. Regnie, and J. Roberts. Statistical Bandwidth Sharing: A Study of Congestion at Flow Level. *SIGCOMM'01 Conference*, August 2001.

[FdCPPdR01] Marcial Porto Fernandez, Aloysio de Castro P. Pedroza, and Jos Ferreira de Rezende. QoS Provisioning across a DiffServ Domain using Policy-Based Management. In *Proceedings of IEEE Globecom*, San Antonio, USA, November 2001.

[FG99] B. Fox and B. Gleeson. Virtual Private Networks Identifier, September 1999. RFC 2685.

[FJ95] Sally Floyd and Van Jacobson. Link-Sharing and Resource Management Models for Packet Networks. *IEEE/ACM Transactions on Networking*, 3(4), August 1995.

[Fos97] Halldor Fossa. *Interactive Configuration Management for Distributed Systems*. PhD thesis, University of London,, April 1997.

[FWD+01] F. L. Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. MPLS Support of Differentiated Services. Internet Draft `draft-ietf-mpls-diff-ext-09.txt`, April 2001. work in progress.

[GBK99] M. Günter, T. Braun, and I. Khalil. An Architecture for Managing QoS-enabled VPNs over the Internet. In *Proceedings of the 24th Conference on Local Computer Networks LCN'99*, pages p.122–131. IEEE Computer Society, October 1999.

[GLH+99] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A Framework for IP Based Virtual Private Networks. Internet Draft `draft-gleeson-vpn-framework-03.txt`, 1999. work in progress.

[Gro] The IETF Policy Framework (Policy) Working Group. http://www.ietf.org/html.charters/policy-charter.html.

[Gün01] Manuel Günter. *Management of Multi-Provider Internet Services with Software Agents*. PhD thesis, IAM, University of Berne, June 2001.

[GY98] G Goldszmidt and Y. Yemini. Delegated Agents for Network Management. *IEEE Communications Surveys*, 36(3):66 – 70, March 1998.

[HB99] Alex L. G. Hayzelden and John Bigham. Agent Technology in Communications Systems: An Overview. *Knowledge Engineering Review*, 1999.

[HBCM99] Mor Harchol-Balter, Mark E. Crovella, and Cristina D. Murta. On Choosing a Task Assignment Policy for a Distributed Server system. *Journal of Parallel and Distributed Computing*, 59(2):204–228, November 1999.

[Jaf81] J.M. Jaffe. Bottleneck Flow Control. *IEEE Transactions on Communications*, 29(7):954 – 962, 1981.

[JD02] Manish Jain and Constantinos Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP throughput. *SIGCOMM'02 Conference*, August 2002.

[JNP99] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding phb, June 1999. RFC 2598.

[KA98] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol, November 1998. RFC 2401.

[KB00a] I. Khalil and T. Braun. Implementation of a Bandwidth Broker for Dynamic End-to-End Resource Reservation in Outsourced Virtual Private Networks. *The 25th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 511–519, November 9-10 2000.

[KB00b] Ibrahim Khalil and Torsten Braun. Edge Provisioning and Fairness in DiffServ-VPNs. *IEEE International Conference on Computer Communication and Network (I3CN)*, pages 424–431, Oct 16-18 2000.

[KB01a] I. Khalil and T. Braun. A Range-Based SLA and Edge Driven Virtual Core Provisioning in DiffServ-VPNs. *The 26th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 89–90, November 15-16 2001.

[KB01b] Ibrahim Khalil and Torsten Braun. Implementation of a Bandwidth Broker for Dynamic End-to-End Capacity Reservation over Multiple Diffserv Domains. *4th IFIP/IEEE International Conference on Management of Multimedia Networks and Services, MMNS 2001*, pages 160–174, Oct 29 - Nov 1 2001.

[KB02a] Ibrahim Khalil and Torsten Braun. Automated Service Provisioning in Heterogenous Large-Scale Environment. In *8th Network Operations and Management Symposium (NOMS 2002)*, pages 575–588, April 15-19 2002.

[KB02b] Ibrahim Khalil and Torsten Braun. Edge Provisioning and Fairness in vpn-diffserv Networks. *JOURNAL OF NETWORK AND SYSTEMS MANAGEMENT*, 10(1):11 – 38, March 2002.

[KBG00] Ibrahim Khalil, Torsten Braun, and M. Günter. Implementation of a Service Broker for Management of QoS enabled VPNs. In *IEEE Workshop on IP-oriented Operations & Management (IPOM'2000)*, pages 13–23, September 2000.

[KK95] Thomas Koch and Bernd Krmer. Towards a Comprehensive Distributed Systems Management. *Open Distributed Processing, IFIP*, pages 259 – 270, 1995.

[KKR95] T. Koch, B. Kramer, and G. Rohde. On a Rule Based Management Architecture. *Proceedings of the Second International Workshop on Services in Distributed and Networked Environment,Canada*, pages 68 – 75, June 1995.

[KLMB96] M. Katchabaw, H. Lutfiyya, A. Marshall, and M. Bauer. Policy-driven Fault Management in Distributed Systems. *Proceedings of the The Seventh International Symposium on Software Reliability Engineering (ISSRE)*, November 1996.

[KLS97] H. Ku, G. W. Luderer, and B. Subbiah. Intelligent Mobile Agent Framework for Distributed Network Management. *In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'97), Phoenix,USA*, November 1997.

[KRSY01] Amit Kumar, Rajeev Rastogi, Avi Silberschatz, and Bulent Yener. Algorithms for Provisioning Virtual Private Networks in the Hose Model. *SIGCOMM'01 Conference*, August 2001.

[KS99] E. Knightly and N. Shroff. Admission control for statistical qos: Theory and practice. *IEEE Network*, 13(2):20–29, 1999.

[LC01] Raymond R.-F. Liao and Andrew T. Campbell. Dynamic Core Provisioning for Quantitative Differentiated Service. *IWQoS 2000*, June 6 -8 2001.

[LMB97] Hanan L. Lutfiyya, Andrew D. Marshall, and Michael A. Bauer. Configuration Maintenance for Distributed Applications Management. *Proceedings of the CAS Conference (CASCON)*, pages 43–57, November 1997.

[LR92] H.-C. Lin and C.S. Raghavendra. A Dynamic Load-Balancing Policy with a Central Job Dispatcher (lbc). *IEEE Transactions on Software Engineering*, 18(2):148 – 158, February 1992.

[McK90] P E McKenny. Stochastic Fairness Queueing. *INFOCOM'90 Conference*, June 1990.

[MEB+95] K. Meyer, M. Erlinger, J. Betser, C. Sunshine, G. Goldszmidt, and Y. Yemini. Decentralizing Control and Intelligence in Network Management. *The Fourth International Symposium on Integrated Network Management*, pages 63–83, May 1995.

[MFZH99] J.P. Martin-Flatin, S. Znaty, and J.P. Hubaux. A Survey of Distributed Enterprise Network and Systems Management Paradigms. *Journal of Network and Systems Management*, 7(1), March 1999.

[MKM+01] K. Muthukrishnan, C. Kathirvelu, A. Malis, T. Walsh, F. Ammann, J. Sumimoto, and J. M. Xiao. Core MPLS IP VPN Architecture. Internet Draft `draft-ietf-ppvpn-rfc2917bis-00.txt`, July 2001. work in progress.

[ML88a] Axel Mahler and Andreas Lampen. shape - a software configuration management tool. In *System Configuration Management*, pages 228–243, 1988.

[ML88b] Axel Mahler and Andreas Lampen. A toolkit for software configuration management. In *In Proceedings of the EUUG Spring Conference*, pages 185–202, April 1988.

[MlCPT00] Hemann De Meer, Aurelio la Corte, Antonio Puliafito, and Orazio Tomarchio. Programmable Agents for Flexible QoS Management in IP Networks. *IEEE Journal on Selected Areas in Communications*, 18(2), February 2000.

[MPE] MPEGTV. Mpeg tv home page. http://www.mpegtv.com/.

[NBBB98] K. Nichols, S. Blake., F. Baker, and D. Black. Definition of the Differentiated Services Field (DS field) in the IPv4 and IPv6 Headers, December 1998. RFC 2474.

[NJZ99] K. Nichols, Van Jacobson, and L. Zhang. A Two-bit Differentiated Services Architecture for the Internet, July 1999. RFC 2638.

[OMK+97] Kohei Ohta, Takumi Mori, Nei Kato, Hideaki Sone, Glenn Mansfield, and Yoshiaki Nemoto. Divide and conquer technique for network fault management. In *Proceedings of ISINM97*, pages 675–687, May 1997.

[PG93] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case. *IEEE/ACM Transactions on Networking*, 1(3):344 – 357, June 1993.

[PG94] A. K. Parekh and R. G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node Case. *IEEE/ACM Transactions on Networking*, 2(2):137 – 150, April 1994.

[PR99] N. De Palma and B. Riveill. Dynamic Reconfiguration of Agent-based Applications. In *European Research Seminar on Advances in Distributed systems (ERSADS'99)*, April 1999.

[PSV00] George A. Politis, Petros Sampatakos, and Iakovos S. Venieris. Design of a multi-layer bandwidth broker architecture. In *INTERWORKING*, pages 316–325, 2000.

[PT99] A. Puliafito and O. Tomarchio. Advanced network management functionalities through the use of mobile software agents. *In Proceedings of the 3rd International Workshop on Intelligent Agents for Telecommunication Applications (IATA'99),Stockholm, Sweden*, August 1999.

[QBO00] QBONE. The Internet2 QBone Bandwidth Broker, 2000. http://www.internet2.edu/qos/qbone/QBBAC.shtml.

[QK01] Jingyu Qiu and Edward W. Knightly. Measurement-based Admission Control with Aggregate Traffic Envelopes. *IEEE/ACM Transactions on Networking*, 9(2):199–210, 2001.

[RCK02] Satish Raghunath, Kartikeya Chandrayana, and Shivkumar Kalyanaraman. Edge-based QoS Provisioning for Point-to-Set Assured Services. *ICC'2002 Conference*, April 2002.

[RRR98] M. Reisslein, K. W. Ross, and S. Rajagopal. Guaranteeing Statistical QoS to Regulated Traffic: The Multiple Node Case. *In Proceedings of 37th IEEE Conference on Decision and Control (CDC),Tampa*, December 1998.

[SA98] Morin C Sahai A. Towards Distributed and Dynamic Network Management. *Proceedings of the IEEE/IFIP Network Operation and Management Symposium (NOMS),New Orleans, USA*, February 1998.

[SBGP99] B. Stiller, T. Braun, M. Günter, and B. Plattner. Charging and accounting technology for the Internet. In *4th European Conference on Multimedia Applications, Services, and Techniques ECMAST'99*, LNCS 1629, pages 281–296. Springer-Verlag, May 1999.

[Sch97] J. Schonwalder. Network Management by Delegation - From Research Prototypes Towards Standards. *Computer Networks*, 29(15):1843–1852, November 1997.

[Sch98] Olov Schelen. *Quality of Service Agnets in the Internet.* PhD thesis, Lulea University of Technology, August 1998.

[Sit] Cisco Web Site. http://www.cisco.com.

[SL99] M. Sloman and E. Lupu. Policy Specification for Programmable Network. *First Int. Working Conference on Active Networks (IWAN99),Berlin*, June 1999.

[Slo94] Morris Sloman. Policy Driven Management for Distributed Systems. *Journal of Network and Systems Management*, 2(4):333–360, 1994.

[SQK00] J. Schonwalder, J. Quittek, and C. Kappler. Building Distributed Management Applications with the IETF script MIB. *IEEE Journal on Selected Areas in Communications*, 18(5):702 – 714, May 2000.

[SSZ98] Ion Stoica, Scott Shenker, and Hui Zhang. Core-Stateless Fair Queueing: A Scalable Architecture to Approximate Fair Bandwidth Allocations in High Speed Networks. *SIGCOMM'98 Conference*, 1998.

[Sta98] William Stallings. SNMP and SNMPv2: The Infrastructure for Network Management. *IEEE Communications Magazine*, 36(3), March 1998.

[Sta02] Günther Stattenberger. *Scalable Quality of Service Support for Mobile Users*. PhD thesis, IAM, University of Berne, December 2002.

[SV97] D. Stiliadis and A. Varma. A General Methodology for Desigining Efficient Traffic Scheduling and Shaping Algorithms. *Proc. IEEE INFOCOM'97*, 1997.

[SWI] SWITCH. The switchlan backbone. http://www.switch.ch/lan/national.html.

[SWW01] Subhash Suri, Marcel Waldvogel, and Priyank Ramesh Warkhede. Profile-based Routing: A New Framework for MPLS Traffic Engineering. In *Proceedings of QofIS 2001*, pages 138–157, 2001.

[SZ99] Ion Stoica and Hui Zhang. Providing Guaranteed Services Without Per Flow Management. *SIGCOMM'99 Conference*, August 1999.

[SZN97] Ion Stoica, Hui Zhang, and T S E Ng. A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real Time and Priority Queues. *SIGCOMM'97 Conference*, September 1997.

[Tea99] Benjamin Teitelbaum and et al. Internet2 QBone: Building a Testbed for Differentiated Services. *IEEE Network*, September/October 1999.

[TFP02] P. Trimintzios, P. Flegkas, and G. Pavlou. Policy-driven Traffic Engineering for Intra-domain Quality of Service Provisioning. *QofIS/ICQT'02*, October 2002.

[TOT+99] Andreas Terzis, Jun Ogawa, S. Tsui, Lan Wang, and Lixia Zhang. A Prototype Implementation of the Two-Tire Architecture for Differentiated Services. In *IEEE RTAS'99*, 1999.

[TWOZ99] Andreas Terzis, Lan Wang, Jun Ogawa, and Lixia Zhang. A Two-Tier Resource Management Model for the Internet. In *IEEE Global Internet'99*, December 1999.

[VPN] VPNcon. Vpncon spring 2000. http://www.vpncon.com/spring/springvpn.htm.

[WBLH90] U. Warrier, L. Besaw, L. LaBarre, and B. Handspicker. The Common Management Information Services and Protocols for the Inte rnet, October 1990. RFC 1189.

[Wd89] F. Wong and J.R.B. deMarca. Fairness in Window Flow Controlled Computer Networks. *IEEE Transactions on Communications*, 37(5):954 – 962, 1989.

[Wie94] Rene Wies. Policies in Network and Systems Management - Formal Definition and Architecture. *Journal of Network and Systems Management*, 2(1):63–83, March 1994.

[WSF82] J.W. Wong, J.P. Sauve, and J.A. Field. A Study of Fairness in Packet Switching Networks. *IEEE Transactions on Communications*, 30(2):346 – 353, 1982.

[YDP00] R. Yavatkar and R. Guerin D. Pendarakis. A Framework for Policy-based Admission Control, January 2000. RFC 2753.

[YKF00] Y. Yemini, A. V. Konstantinou, and D. Florissi. NESTOR: An Architecture for Self-Management and Organization. *IEEE Journal on Selected Areas in Communications*, 18(5):758 – 766, May 2000.

[ZC93] Moshe Zukermann and Sammy Chan. Fairness in ATM networks. *Computer Networks and ISDN Systems*, 26, 1993.

[ZDGH00] Zhi-Li Zhang, Zhenhai Duan, Lixin Gao, and Yiwei Thomas Hou. Decoupling qos control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. *ACM SIGCOMM 2000*, August 2000.

Author's Biography

IBRAHIM KHALIL has a M.S. degree in computer engineering from University Putra Malaysia and a B.S. in electrical engineering from BIT Rajshahi, Bangladesh. He was a graduate research assistant with the ATM research group in electronics and computer engineering, University Putra Malaysia between 1993 and 1996. Prior to joining the group of Prof. Braun in 1998 as research assistant and Ph.D. student, he briefly worked with the LTS and C3i groups of EPFL, Switzerland. His research interests are dynamic network provisioning, QoS issues of VPN, MPLS, and network pricing. He is currently working with Mountain View, California based Ponte Communications, USA.